Appeared in Pacific Graphics 2002

An Efficient Brush Model for Physically-Based 3D Painting

Nelson S.-H. Chu Chiew-Lan Tai

Department of Computer Science
Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong
{cpegnel, taicl}@ust.hk

Abstract

This paper presents a novel 3D brush model consisting of a skeleton and a surface, which is deformed through constrained energy minimization. The main advantage of our model over existing ones is in its ability to mimic brush flattening and bristle spreading due to brush bending and lateral friction exerted by the paper surface during the painting process. The ability to recreate such deformations is essential to realistic 3D digital painting simulations, especially in the case of Chinese brush painting and calligraphy. To further increase realism, we also model the plasticity of wetted brushes and the resistance exerted by pores on the paper surface onto the brush tip. Our implementation runs on a consumer-level PC in real-time and produces very realistic results.

Figure 1: ***Calligraphy made with our system***

1. Introduction

Digital painting has been adopted by many artists due to its advantages over traditional media in terms of convenience, ease of experimentation, and the possibility of combining the effects of multiple traditional media in one digital painting. It is now possible to achieve the effects of various western painting media (e.g. oil or charcoal) realistically using commercial packages like Corel Painter [13]. In contrast, however good a 2D mark-making technique is, it cannot extend the expressiveness of Chinese brushes to the digital domain. In Chinese brushwork, a painter uses his brush in a manner similar to the way a musician uses his instrument to deliver harmonious rhythm – each brush stroke should be rendered in a continuous rhythmic movement so that the painted subject exhibits vigor and spirit [10, 18]. The artist's intent is always to capture the spirit of the painted subject, rather than the accuracy of its outward appearance, and this is often achieved by the grace of a few deft strokes [1]. In the case of calligraphy, the brush movement is analogous to dancing. This kind of expressive executions is achievable because of the soft-yet-resilient quality of the brush tuft.

Our aim is to simulate the deformation of Chinese brushes and the ink deposition during the painting process in real-time. It is expected that our 3D brush model, when combined with an ink-diffusion simulation method and a six degrees-of-freedom (DOF) input device, can be a tool for creating digital Chinese brushwork. The benefits of such a tool include the following:

- *Create electronic art more naturally and with spontaneity* – users paint and draw with a virtual brush rather than edit control points, allowing individual user style to be naturally embedded in the art pieces.
- *Render oriental font with aesthetic quality of real calligraphy* – high-resolution characters with more visually pleasing features can be generated.
- *Parameters can be modified to produce different effects* – artists can easily experience the effects of different types of brushes and paper.
- *Non-photorealistic rendering* – Chinese brush painting style can be applied to 3D object rendering.

Our model is empirical in that we attempt to model only physical properties that are necessary for producing realistic visual results. Our emphasis is on

[1] Chinese paintings are categorized into two main types: *meticulous* and *spontaneous* [10]. Throughout this paper, we refer to the spontaneous style.

reproducing features of Chinese brushes that are important in the artistic sense.

2. Previous Work

Earlier efforts in brushwork simulation focused on stroke rendering. Strassmann [19] swept a one-dimensional texture to obtain varying shades within one stroke. Strokes generated by this method look artificial because the natural spreading of the brush bristles is not modeled. In [14], Pham modeled the trajectory of a stroke as a planar cubic B-spline, and obtained the width of the stroke by offsetting the knots in the spline. It is difficult to produce natural looking strokes by knot specification, and the effects of bristles spreading and varying shades also look artificial. The skeletal stroke technique [7] deformed 2D strokes to produce amazing results and it worked well for making illustrations. However, for Chinese brushwork, or watercolor-like painting in general, this technique requires storing a large sample of stroke textures to avoid appearing repetitive. Strokes with self-intersection or high curvature are also not realistic because the stroke textures are not generated physically.

To produce more realistic brush strokes, later research efforts incorporated physics into the brush models. Wong et al. [22] modeled a calligraphy brush as an inverted cone, with the footprint controlled by user-adjustable parameters. Using the theory of elasticity, Lee [11] modeled a brush as a collection of rods with homogeneous elasticity along the entire length. Saito et al. [17] used a Bezier spine curve and a set of discs centered along the curve to model a brush. Unfortunately, all these models are too simplistic to produce the effects of some commonly used painting techniques. For example, the cone model ignores the brush tip when a brush is pressed down, and thus fails to produce *slanted-brush* strokes (in which the tip travels along one side of a stroke rather than staying in the middle) in painting and calligraphy [10]. The splitting of bristles is also not adequately simulated in these systems and thus cannot produce realistic brush footprints.

In the recent work of Baxter et al. [3], a western brush was modeled as a simple spring-mass system. Using an approximated implicit integration method, they were able to produce a real-time system for doing acrylic-like painting. Like other previous techniques, no attention is paid to brush spreading or splitting. Thus, their system cannot be applied to produce oriental brushwork since the brush spreading plays an important part in such painting process.

3. Introduction to Chinese Brushes

To design an effective brush model, we must first understand thoroughly the properties that Chinese brushes exhibit. Chinese brushes are made from animal hairs. The anatomy of a typical brush [18] is shown in Figure 2. A layer of shorter hairs called *mantle* is placed inside the brush body, and the empty space created by the shorten hairs serve as a reservoir for ink. Some special-purpose brushes may have different hair layer arrangements made with different types of hairs. According to the type of hair used, Chinese brushes can be generally classified into three main types, namely, *hard*, *soft* and *combination*, each of which has different stiffness and degree of absorbent. For example, a combination brush has a hard kernel, but softer and more absorbent hairs in the outer layer. The kernel can also be waxed to give extra stiffness. Since the type of brush can affect the output considerably, a good brush model should be flexible enough to accommodate these variants of brush characteristics.

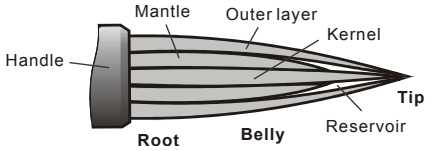


Figure 2: ***Anatomy of a typical brush***

A good brush is said to be *elastic* – it bends when some external force is applied to it and restores to its original shape when the force is removed. However, practically all brushes, especially soft ones, are inelastic to a certain degree when moisten. The tip of a brush is less stiff than its root for two reasons: natural animal hair is thinner at its tip, and less hairs reach the tip due to their different lengths. Therefore, we should also model variable stiffness along the length of a hair tuft.

A brush forms a single tuft and runs into a fine tip when it is moisten because the attraction force of water molecules is large enough to pull all the bristles together. It becomes bushy only when it is dry or worn away after long use. The same brush can be used to draw both fine and bold lines by applying different pressures and holding at different angles. As the brush gradually dries up while ink is being deposited onto the paper, the attraction force of water molecules reduces and the tip of the tuft starts to split into two or more tips. This characteristic, together with lessen ink loading, provides the condition for producing the *flying-white* effect [10], in which white areas appear in the stroke. In addition, whenever the artist desires, the tip can be deliberately split to produce multiple lines in a single stroke. We observe that a small number (at most five) of tufts suffice for most painting process; treating the brush as more tufts is necessary only when applying special painting techniques, such as the *split-brush* technique [10] for painting hair-like objects.

4. Brush Model

The ultimate way to producing realistic dynamics of a brush tuft is to physically simulate each and every bristle, which is simply not practical. So, the challenge we face is how to collectively simulate the bristles so that the simulation can be done in real-time, and yet the model is flexible enough to yield the effects expected by

artists. The idea of collective modeling has also been proposed for human hair modeling [8, 16, 20].

With the properties described earlier in mind, we design a brush model consisting of three components: *brush geometry*, *brush dynamics,* and *ink loading and depositing*. The details are presented in the following sub-sections.

4.1 Brush Geometry

The representation of the brush geometry is closely related to how we model the brush dynamics. Like some previous models [3, 4], we employ a layered structure**.** The geometry consists of two layers, namely the *skeleton* and the *surface*.

4.1.1 Brush Skeleton. The skeleton consists of a *spine* and some *lateral nodes*. Figure 3 shows our geometric model for a brush tuft. We represent the spine as a connected sequence of line segments of decreasing lengths towards the tip. Since the brush root is usually much stiffer than the tip, it bends much less; in fact, usually only the tip and the belly are used to paint. Therefore, for modeling efficiency, progressively shorter segments are used towards the brush tip so as to dedicate higher resolution to the tip. The highest node attached directly to the brush handle is called the *root node*. Each joint between two adjacent spine segments has two DOF's in the polar coordinate system (see Figure 4). Suppose the spine has n + 1 nodes, $\boldsymbol{N}_0,\boldsymbol{N}_1,...,\boldsymbol{N}_n$, with $\boldsymbol{N}_0$ as the root node. We denote the positions of the nodes by $\boldsymbol{O} = (O_0, O_1, ..., O_n)$. The differential changes in the orientation of the segments are denoted by $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_n)$ and $\boldsymbol{\phi} = (\phi_1, \phi_2, ..., \phi_n)$. We refer to θ as the *bend-angles*, and ϕ the *turn-angles*, since θ_i is the angle between the ith spine segment and its previous segment, and ϕ_i gives the angle of rotation of the ith segment about its previous segment. We refer to $(\boldsymbol{\theta}, \boldsymbol{\phi})$ as the *state* of the spine. A local Cartesian frame is defined for each node, by rotating the frame of the previous node by the turn- and bend-angles. The spine is responsible for the general motion of the whole tuft.

In addition to the brush spine, we introduce some lateral nodes to further model the tuft deformation. Each spine node has two lateral nodes attached to it. To

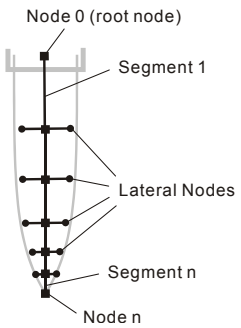


Figure 3: ***Geometric model of a brush tuft***

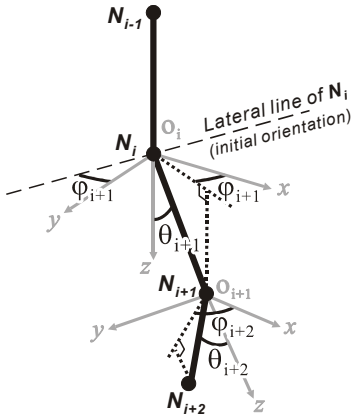


Figure 4: ***Notations for our geometric model***

constrain the movement of a lateral node associated to a spine node $\boldsymbol{N}_i$, we define the *joint-bisecting plane* of $\boldsymbol{N}_i$ as the plane passing through O_i, bisecting the angle between the two adjacent spine segments and perpendicular to the plane spanned by the segments. The lateral node is then constrained to move along a line passing through O_i on the joint-bisecting plane. We call this line the *lateral line* of $\boldsymbol{N}_i$. Initially, the lateral line is set to be normal to the two adjacent spine segments; during simulation it has one rotational DOF on the joint-bisecting plane.

The two lateral nodes attached to a spine node represent two groups of bristles on both sides of the spine. We observe that this configuration can effectively capture the essence of tuft deformation for the following reasons. Since the brush interacts with only a planar painting surface, the brush footprint is largely determined by tuft flattening, controlled by bending and lateral drag (e.g., when doing *slanted-brush* strokes). With the lateral lines having one DOF, the non-penetration constraints in our dynamic model tend to keep the lateral lines of those spine nodes that touch the paper to be parallel to the paper. Consequently, when the brush is pressed against the paper, the loci of the lateral nodes would lie on the painting surface, and thus effectively model horizontal deformation and the lateral spreading of the bristles.

4.1.2 Brush Surface. The brush surface is obtained by sweeping a varying elliptic cross section along the spine. For moistened and unbent brushes, the cross sections are assumed to be circles throughout the tuft. We pre-define these circle radii for various types of brushes and call them the *minimum tuft radii*. In general, the cross section is composed of two half ellipses having a common minor radius, but possibly different major radii, as shown in Figure 5. This simple representation is computationally efficient and does not differ much from the observed reality. A cross section $\boldsymbol{\Omega}_i$ at a spine node $\boldsymbol{N}_i$ lies on the joint-bisecting plane, and its major axis coincides with the lateral line of $\boldsymbol{N}_i$. To generate the brush surface, we derive the cross sections between spine nodes by interpolating O_i, the frame axes and radii of $\boldsymbol{\Omega}_i$ using cubic spline.

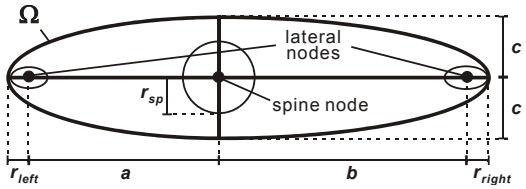
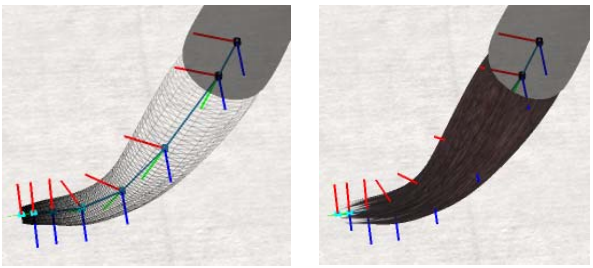
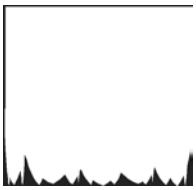


Figure 5: ***Tuft cross section***



(a) Surface mesh **(b) Textured**



(c) Alpha map

Figure 6: ***Bristle splitting for a single tuft***

The radii of $\boldsymbol{\Omega}_i$ at a spine node N_i are determined by the positions and the *effective radii* of its two lateral nodes. For intersection checking during dynamics simulation, the spine and lateral nodes are assigned effective radii to model the thickness of the bristle groups they represent. For a spine node, since it is sufficient to assume that the bristle group has circular cross section (elliptic shape would be taken care of by the lateral nodes), it is assigned an effective radius r_{sp}; for a lateral node, to better model the flattening, we assume that the bristle group forms an ellipse and assign to it the major and minor effective radii (see Figure 5). All these effective radii are set as fractions of the minor radius of $\boldsymbol{\Omega}_i$ from the previous time frame. Let a and b be the distances between the lateral nodes and the spine node as shown in Figure 5. The major radii for $\boldsymbol{\Omega}_i$ are then taken as $(a + r_{left})$ and $(b + r_{right})$, where r_{left} and r_{right} are the major effective radii of the lateral nodes. By the conservation of cross section area, we compute the minor radius, c, as follows:

$$c = \frac{2r^2}{a + b + r_{left} + r_{right}}$$

where r is the *minimal tuft radius* at the spine node .

With a single tuft, it is not possible to model bristle splitting geometrically. To achieve brush footprints with bristle-splitting effect, we use an alpha map to make part of the brush surface transparent, as shown in Figure 6(b). The alpha map for the tuft surface can be generated dynamically by patching white tuft-like shapes of various lengths and widths onto a black image. Currently, our implementation uses a static map.

4.2 Brush Dynamics

One way to model the brush dynamics is to use bend and stretch springs at the spine joints, assign point masses to the nodes, and use Newtonian physics to setup the motion equations for the nodes. However, since bristles have very little inertia relative to the forces applied onto them, by the Newton's second law, large accelerations could result. Exerted by its own internal spring forces, a brush tuft also behaves as a highly damped system. Given the stiff nature of the dynamic system, it is difficult to produce a tractable real-time system for the brushes by solving second-order differential equations derived from Newton's second law. Baxter et al. [3] cope with the instability difficulty by adding large damping and employing an approximated implicit integrator [6], which is reported to be more stable and faster, but much less accurate, than the large step integrator presented in [2]. However, the approximated integrator has the drawback of having to model the brush internal forces using only stretch springs rather than bend springs, which model non-stretchable bristles more naturally.

An alternative to modeling the brush dynamics is to use energy minimization to determine the steady state of the brush at a given time step. An energy function is set up for the system and its steady state is determined by finding a local energy minimum numerically. In previous cloth simulation work, energy-based methods have generally been used to simulate static scenes [12]. However, since brushes are in equilibrium almost all the time during the painting process, energy minimization is also a viable approach for simulating brush dynamics; it was employed by Saito et al.[17]. Therefore, to avoid solving stiff differential equations [21], we employ the energy minimization approach for our brush simulation.

4.2.1 Energy Minimization Problem. We formulate the brush dynamics system as a *static constrained minimization problem* [1]. The energy function takes into account the frictional and deformation energies. We use Sequential Quadratic Programming (SQP) to solve the constrained energy minimization problem because of its fast convergence. We first describe our energy minimization algorithm, and then give the details of our energy function formulation in Section 4.2.2.

In general, the initial estimates are crucial for solving minimization problems. Fortunately, for our simulation, the state of the previous frame serves as a very good initial estimate. Suppose the positions of the spine nodes are $\boldsymbol{O_i} = (O_{i0}, O_{i1}, \ldots, O_{in})$ and the spine state is $(\boldsymbol{\theta_i}, \boldsymbol{\phi_i})$ for the current frame. Further suppose that the new position tracked by the input device is p_1 for the next frame. We determine the new node positions $\boldsymbol{O_f} = (O_{f0}, O_{f1}, \ldots, O_{fn})$ and the new spine state $(\boldsymbol{\theta_f}, \boldsymbol{\phi_f})$ in the

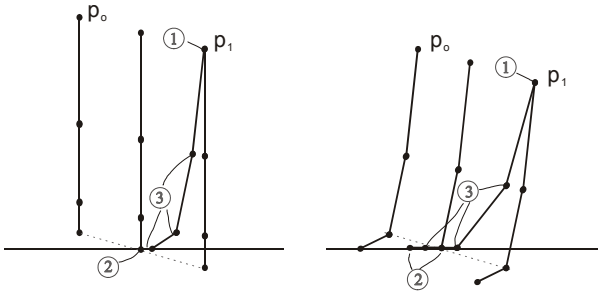


Figure 7: ***Two possible scenarios for determining the positions of the spine nodes after moving the brush from p_0 to p_1***

following steps (circled numbers in Figure 7 correspond to these steps):

1. Set $O_{f0} = p_1$ and initialize $(\theta_f, \phi_f) = (\theta_i, \phi_i)$.
2. Determine if any spine node penetrates the paper, and set minimization constraints for such nodes to be above the paper. Optionally, obtain a better initial estimate by updating (θ_f, ϕ_f) so that no nodes penetrate the paper.
3. Solve the constrained energy minimization problem for the state (θ_f, ϕ_f) and update the node positions $\boldsymbol{O}_f$ accordingly.

The lateral nodes are to be dragged with friction against the paper and their positional deviations from their rest positions contribute to the deformation energy of the tuft. For both the spine nodes and the lateral nodes, the frictional work is calculated by assuming that the nodes constrained to be above the paper (i.e. those penetrating the paper initially) are dragged from their contact positions to their final positions, with the contact positions taken as the positions where the nodes first touch the paper (determined by interpolation, shown as dotted line in Figure 7). We refer to this displacement as the *dragging vector* of the node. When checking if a node penetrates the paper, we consider a spine node as non-penetrating only if it is at least r_{sp} above the paper since the node represents a tuft with thickness; a lateral node is non-penetrating only if the ellipse representing the bristle group is above the paper. Thus, part of the generated brush surface will actually penetrate the paper plane and it determines the brush footprint for ink depositing. For efficiency, only the lateral nodes of alternating spine nodes are fed into the dynamics simulation; the positions of the other lateral nodes are determined by interpolation.

4.2.2 Energy Functions. In our energy function formulation, we include only those components that have significant effects on the dynamic behavior. We exclude the potential energy of the tufts due to gravity since the brush mass is small. The twisting energy of the bristles and the kinetic energy of the tuft are also neglected due to their small contribution. We define the energy function as $E = E_{deform} + E_{frict}$, where E is the total energy, E_{deform} the tuft deformation energy and E_{frict} the frictional work done by dragging the brush against the paper surface.

Deformation Energy. The tuft deformation energy E_{deform} has two components, E_{spine} and $E_{lateral}$, which account for the bending of the tuft spine and the lateral deformation of the tuft respectively. In our brush model, each spine joint has two DOF's, namely the bend-angle and the turn-angle. We impose a bend spring at a spine joint for each DOF to model the bending force of the tuft. The energy stored in a bend spring is expressed as

$$\text{BendEnergy}(\theta) = \kappa_{bend}|\theta|^m$$

where κ_{bend} is the spring coefficient, $m \geq 2$ to account for the non-linearity of real bristles. Setting $m = 3$ works well in our implementation.

When the brush is wet, the attractive force between the water and bristle molecules holds the bristles together. When the brush is bended by an external force, work is done against the friction caused by the molecular attractive force. Some energy is transformed to the internal (heat) energy of the water and bristle molecules. To account for this phenomenon, we also introduce an energy term on the *change* of bend- and turn-angles in the same form as that for bend springs. The final spine deformation energy, E_{spine}, is defined as a weighted sum of bend spring energies for the bend- and turn-angles and energies on the angle changes.

The lateral nodes, apart from being dragged by frictional force against the paper, are subject to two sources of spring forces: stretch springs along the lateral lines, and bend springs connecting consecutive lateral nodes along the tuft. We define $E_{lateral}$ as a weighted sum of the energies of these stretch and bend springs. The stretch springs account for the attraction force of water molecules that pulls the lateral nodes towards the spine and thus the spring coefficient is a function of the wetness. The energy function for a stretch spring is in the form:

$$E_{stretch}(d,\theta) = \kappa_{stretch}|d - (r + s(\theta))|^m$$

where $\kappa_{stretch}$ is the spring coefficient, d is the distance of the lateral node from its spine node, r is the minimal tuft radius of the spine node, and s is a linear function of the bend-angle θ at the associated spine node. Empirically, $\kappa_{stretch}$ can be a simple linear function of the current wetness.

The bend springs are added to the lateral nodes to account for the bending of the bristle groups represented by the nodes. The energy of such a bend spring is a function of the angle between two line segments, each connecting the associated lateral node to one of its two adjacent lateral nodes (shown as β_i's in Figure 8). For modeling stiffer brushes, this energy term would have a larger weighting with respect to that of the stretch springs in its contribution to $E_{lateral}$.

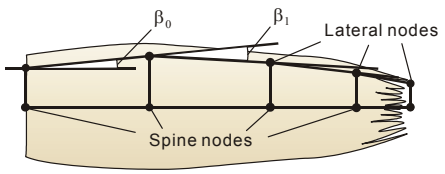


Figure 8: ***Bend springs for lateral nodes***

Frictional Energy. We formulate the frictional energy of a brush dragged against a painting surface as follows:

$$E_{fricct} = \mu \sum_{\substack{\text{contacting} \\ \text{nodes}}} F \cdot \|\Delta \mathbf{x}\|$$

where μ is the frictional coefficient, F is the normal force that a contacting node exerts on the paper, and $\Delta \boldsymbol{x}$ is the dragging vector of the node defined in Section 4.2.1. A node is said to be a *contacting node* if it corresponds to an active constraint or if its normal force F in the previous frame is non-zero. Suppose $\boldsymbol{N}_j$ is the contacting node with the smallest index. The normal force F for $\boldsymbol{N}_j$ can be taken as the sum of the vertical components of the spring forces at all the nodes $\boldsymbol{N}_i$'s for $i < j$. For the rest of the contacting nodes, F is taken as the vertical component of the spring force at its previous node. However, since the tuft spine is represented as discrete nodes, the normal force function would not be smooth when a node switches between contacting and non-contacting. In order to avoid drastic discontinuity in the frictional energy function so that the optimization would converge, we distribute part of F at $\boldsymbol{N}_j$ to its upper neighbor $\boldsymbol{N}_{j-1}$ according to the paper-touching proportion of the tuft segment between $\boldsymbol{N}_j$ and $\boldsymbol{N}_{j-1}$. For a lateral node, we set its F as a fraction of the normal force of the associated spine node.

Since the bristles forming the tuft are generally aligned, the tuft surface appears corrugated. This makes the tuft experiences a larger friction when it is dragged sideway. To account for this anisotropic resistance, we also modulate the frictional energy with the direction of the dragging:

$$E_{fricct} = \mu \sum_{\substack{\text{contacting} \\ \text{nodes}}} F \cdot \left(\kappa_f \cdot \|\Delta \mathbf{x}_{par}\| + (1 - \kappa_f) \cdot \|\Delta \mathbf{x}_{perp}\| \right)$$

where $\Delta \boldsymbol{x}_{par}$ and $\Delta \boldsymbol{x}_{perp}$ are the components of $\Delta \boldsymbol{x}$ parallel and perpendicular to the tip spine projected onto the paper, respectively, and $\kappa_f \in [0,1]$ is the weighting value for $\Delta \boldsymbol{x}_{par}$ and $\Delta \boldsymbol{x}_{perp}$.

4.2.3 Plasticity. As mentioned in Section 4.2.2, when a wet brush is deformed, work has to be done against the molecular friction. When the brush is released, the restoring spring force has to overcome the resistance of molecular friction in order to revert the brush into its original shape. Failing to revert to its original shape makes the brush appears plastic. We used a simple but effective *zero-shifting* method to model this plasticity. Suppose the bending energy for a spine segment with bending angle θ is $k|\theta|^3$ and α is a user-adjustable parameter controlling the tuft plasticity (larger α corresponds to more plasticity). The intuitive idea of our method is to shift the minimum-energy angle from zero to a value determined by the plasticity and the bending angle from the last time frame; that is, if θ' is the bending angle from the previous time frame, the new energy function becomes

$$\text{BendEnergy}(\theta) = k|\theta - \rho|^3, \qquad where\ \rho = min(\theta', \alpha)$$

The plasticity value α may be automatically adjusted according to the current wetness of the brush. We observe that this simple method improves the realism of the brush significantly over previous brush models, giving the plasticity that users of real brushes expect. Brush plasticity affects the rhythmic movement artists make and is reflected on the ink traces. Figure 9 shows the ink traces of the same brush movement with different values of α (in radian).

In real-life Chinese painting, re-shaping the brush tip to a sharp point is often necessary before drawing a new stroke. If desired, the plasticity can be set to zero so that tip re-shaping is eliminated altogether. Clearly, we can include a feature in our implementation such that pressing a key reverts the brush to its original shape.

Figure 9: ***Ink traces of the same brush motion with different brush plasticity values***

4.2.4 Pore Resistance. Most types of painting paper are full of pores. When we slant the brush and bring its tip into contact with the paper, the pores act like a fence impeding sliding. If the brush is then pushed against the paper in the direction towards where the tip is pointing, these pores continue to exert large resistance. Setting up a frictional energy function to account for this behavior would give rise to a steep function, making the optimization harder to converge. Thus, we model this resistance by adding one extra constraint in the minimization problem as a *moving blocking plane*. The blocking plane is normal to the projected spine segment of the brush tip onto the paper surface. The distance between the plane and the brush tip is a function of the height of the tip above the paper and the angle the tip makes with the paper. When the tip is in contact with the paper, the distance between the blocking plane and the tip is very small, and thus the constraint prevents the tip from sliding. An additional lead space between the blocking plane and the brush tip is also introduced as a user-defined parameter to adjust the blocking effect.

In Chinese calligraphy, an artist using a soft brush sometime uses the pore resistance to straighten the tuft and make its tip blade-like. Mimicking pore resistance in our simulation allows the reproduction of this kind of deformation, which is expected by artists. In addition, the simulated pore resistance also helps to produce the effect of the *pushed-stroke* technique employed in painting [10], in which the brush is slanted and pushed

against the paper in the direction it is pointing, giving a 'rough' look to the stroke (Figure 12).

4.3 Ink Loading and Depositing

We store the ink and moisture information at each tuft node. A color gradient can be loaded into the tuft by interpolating color values at different tuft nodes. For ink depositing from the brush onto the painting surface, as in [3], we allow the brush surface to intersect with the paper plane and consider the orthogonal projection of the penetrating portion onto the paper plane as the brush footprint. The footprint is obtained by rendering the brush surface clipped by the paper plane in orthogonal view using OpenGL. In this way, we can utilize the hardware-accelerated polygon clipping and rasterization and leave the CPU computation power for the physics simulation. After the footprint is obtained, ink is deposited simply by transferring ink/water values from the penetrating tuft surface area onto the footprint area on the painting surface. We can either subtract the ink values from the tuft or just maintain the ink level to allow continuous painting without reloading. As the ink level gradually lowers, the tuft alpha map is modified to reflect the consumption of ink, mimicking the effect of a dry brush. Currently, simple alpha blending using OpenGL is used for applying diluted/transparent color onto the painting surface.

5. Implementation Results

We have implemented a painting system prototype based on our brush model. Our system is written in Object Pascal using Borland Delphi 6. It currently runs in real-time at 25 frames per second on a 1GHz Pentium-III PC with an NVIDIA GeForce2 Pro graphics card. The number of SQP iterations required for a typical time frame is below 10. In our experience, 98% of the time frames require less than 20 iterations.

5.1 System Architecture

Figure 10 shows the design of our painting system. There are three main modules: *user input*, *simulation*, and *brush and stroke rendering*. The user input module reads the position and orientation of the input device, and the user-defined parameters such as ink loading, stiffness, and brush size. The simulation module is the main component of the system and it consists of three sub-modules. The *brush-dynamics* sub-module simulates the behavior of the bristles, i.e., the bending and spreading of bristles due to external forces; the *ink-flow-within-brush* sub-module simulates the phenomenon of ink flowing from the more saturated parts to the less saturated parts in real brushes; the *ink-flow-on-paper* sub-module simulates the ink diffusion in the paper fibres. We have yet to incorporate ink diffusion simulation in our current prototype, and thus the system simulates brush painting as if it is done on *sized paper* (i.e., paper treated with alum). An ink-

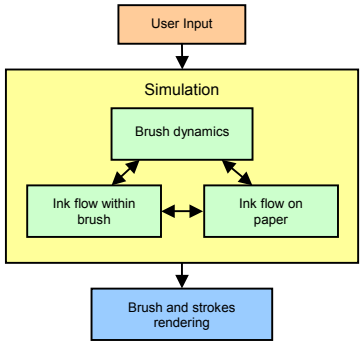


Figure 10: ***Block diagram of our proposed painting system***

depositing method links the brush-dynamics sub-module and the ink-flow-on-paper sub-module. The brush and stroke renderings module renders the appearance of the strokes and draws the brush on the screen to provide feedback to the user regarding its current state (position, orientation, bending, etc).

5.2 User Interface

Visual feedback of the brush shape is important during the painting process. A bended brush would have different footprints when held at different angles. In our system, the main user window shows a perspective view of the 3D painting scene; the camera position and field-of-view are adjustable by the user. An alternative orthogonal view of the painting canvas, with the viewing direction perpendicular to the paper surface and every pixel corresponding to a fixed-size area of the paper model, is also provided. The brush is rendered with lighting and shadows to aid visualization. Shadows provide a natural aid for the user to sense how high the brush is above the paper. Our current system simply renders the shadows as two line segments. During painting, the brush itself can be set to transparent so as not to obstruct the user's view of the painting surface.

Figure 11 shows the physical setup of our current system. To drive the virtual brush, a six-DOF input device is needed. The PHANTOM haptic device [15] not only provides six-DOF data but also the force haptic feedback, and thus is ideal as an input device. A more affordable option is to build a six-DOF device from some 3-DOF devices and sensors. Our current setup makes use of an ultrasonic device and miniature gyroscopes to sense the brush position and orientation respectively. These sensors are attached to a real brush or a brush-like object, which is manipulated to drive the virtual brush in real-time. To provide a natural interface, our input device can be calibrated to map a real supporting surface to the virtual one, so that the real surface also gives some tangible feeling to the user when the brush is pressed down.

As an alternative, our system also supports pressure and tilt sensitive graphics tablets. The sensed pressure is used to control the height of the brush above paper while the tilts the orientation of the brush. Although the brush height is not controlled as intuitively as using a

true 3D positional device, the support for graphics tablet input makes our system more accessible to existing digital artists due to hardware availability. The graphics tablet is also more convenient to use since a supporting ground is already present without calibration.

5.3 Sample Results

Figures 1 and 12 to 17 show some sample painting and calligraphic results obtained using our system. The character *'dragon'* shown in Figure 1 was written using the *slanted-brush* technique. Figure 12 shows some sample strokes used in Chinese painting or calligraphy. Figure 13 shows a sample calligraphy done in contemporary style, with some of the strokes exhibiting the *flying-white* effect. Figure 14 shows some rocks painted with traditional texturing technique. The orchid painting shown in Figure 15 was done in 17 strokes, with the character *'orchid'* also done with our system. Figure 16 shows a flower painting done by loading color gradients onto the brush. Figure 17 shows a calligraphic work with more characters. For video demos and additional color images, please visit the web page: http://www.cs.ust.hk/~cpegnel/VCB/.

6. Conclusion and Future Work

We have presented an efficient model for simulating the deformation of Chinese brushes. Our model is able to produce more realistic tuft deformation, such as bristle spreading and plasticity, which is important for 3D digital brush painting. With some modification, it is expected that our model can also mimic western watercolor or oil painting brushes.

We are currently adding ink diffusion [5, 9, 23] to our prototype to make it a complete system for producing Chinese brushwork. Taking into account the paper texture would also make the dry-brush effect more realistic. It would also be desirable to incorporate haptic input device and stereo display into our system. We are also interested in further investigating vectorial dynamics, including the use of implicit integrator [2], on the speed and accuracy of the simulation. Faster brush dynamics simulation would allow higher brush modeling resolution and more realistic ink/water simulation.

Acknowledgement

The authors would like to thank Kwan-Wah Ng for his generous help in building the input device hardware used in our prototype system, and Rui-Duo Yang for his assistance in hardware interfacing. Thanks also go to the authors of various free software components used in building our prototype, especially Eric Grange and Mike Lischke for the OpenGL library *GLScene*, Mattias Andersson for his *TTablet* component and Dejan Crnila for his *TComPort* library.

References

[1] S. K. Agrawal and B. C. Fabien. *Optimization of Dynamic Systems.* Klumer Academic Publishers, 1999.

[2] D. Baraff and A. Witkin. Large steps in cloth simulation. *SIGGRAPH'98 Proceedings*, pp. 43-54, 1998.

[3] B. Baxter, V. Scheib, M. Lin and D. Manocha. DAB: Interactive haptic painting with 3D virtual brushes, *SIGGRAPH 2001 Proceedings*, August 2001.

[4] J. E. Chawisk, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. *SIGGRAPH'89 Proceedings*, pp. 243-252, July 1989.

[5] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. Computer-generated watercolor, *SIGGRAPH'97 Proceedings*, pp. 421-430, August 1997.

[6] M. Desbrun, P. Schroder, and A. Barr. Interactive animation of structured deformable objects, *Proc. of Graphics Interface '99,* 1999.

[7] S. C. Hsu and I. H. H. Lee. Drawing and animation using skeletal strokes, *SIGGRAPH '94 Proceedings*, Vol. 28, Annual Conference Series, pp. 109-118, July 1994.

[8] C. K. Koh, Z. Huang. Real-time human hair modeling and animation, *SIGGRAPH 2000 Conference Abstracts and Applications*, p.248, 2000.

[9] T. L. Kunii, G. V. Nosovskij, and T. Hayashi. A diffision model for computer animation of diffuse ink painting, *Computer Animation,* 1995.

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

[11] J. Lee. Simulating oriental black-ink painting, *IEEE Computer Graphics and Applications*, 19(3), pp. 74-81, May/June 1999.

[12] H. N. Ng, and GR. L. Grimsdale. Computer graphics techniques for modeling cloth. *IEEE Computer Graphics and Applications*, 16:28-41, 1996.

[13] Painter. Software package by Corel Corporation. http://www.corel.com.

[14] B. Pham. Expression brush strokes, *CVGIP: Graphical Models and Image Processing*, vol. 53, No. 1, 1991.

[15] PHANTOM. Six-DOF input device by SensAble Technologies, Inc. http://www.sensable.com/

[16] E. Plante, M.-P. Cani and P. Poulin. A layered wisps model for simulating interactions inside long hair, *Computer Animation and Simulation 2001*.

[17] S. Saito and M. Nakajima. 3D physics-based brush model for painting, *SIGGRAPH 99 Sketches,* Conference Abstracts and Applications, pp. 226-226, 1999.

[18] J. Silbergeld. *Chinese Painting Style: Media, Methods, and Principles of Form.* University of Washington Press, Seattle and London, 1982.

[19] S. Strassmann. Hairy Brushes, *SIGGRAPH '86 Proceedings*, 20(4), pp. 225-232, August 1986.

[20] Y. Watanabe and Y. Suenaga. A trigonal prism-based method for hair image generation. *IEEE Computer Graphics and Applications*, 12(1):47-53, January 1992.

[21] A. Witkin and D. Baraff. *Physically Based Modeling: Principles and Practice. SIGGRAPH Course Notes,* 1997.

[22] H. T. F. Wong and H. H. S. Ip. Virtual brush: a model-based synthesis of Chinese calligraphy, *Computers and Graphics*, 24(1), pp. 99-113, February 2000.

[23] Q. Zhang, Y. Sato, J. Takahashi, K. Muraoka and N. Chiba. simple cellular automation-based simulation of ink behavior and its application to Suibokuga-like 3D rendering of trees, *Journal of Visualization and Computer Animation*, 1999.

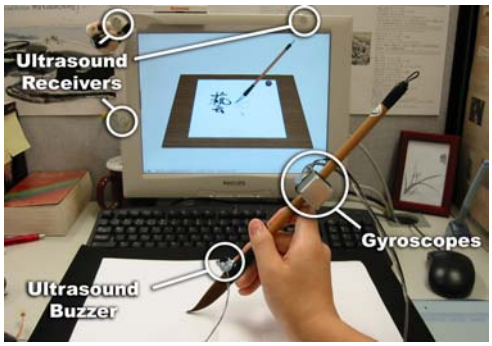


Figure 11: ***Physical setup of our system***

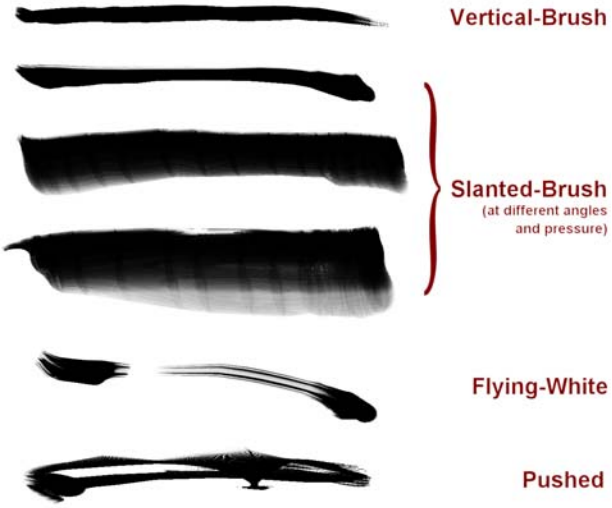


Figure 12: ***Sample strokes***

Figure 13: ***Calligraphy in contemporary style***

Figure 14: ***Rocks painted with texturing strokes***

Figure 15: ***Sample orchid painting***

Figure 16: ***Flowers painted with color gradients***

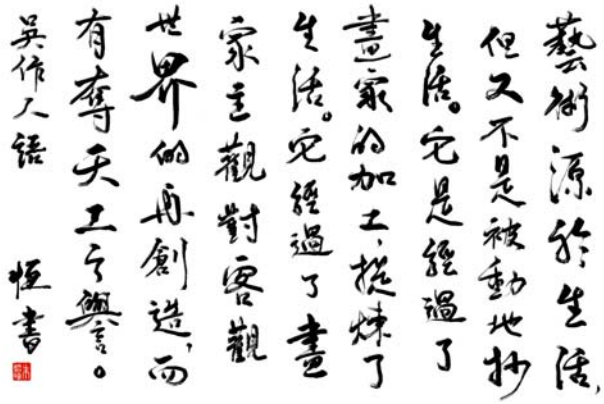


Figure 17: ***Calligraphy in Action Script***