A Versatile Interactive 3D Brush Model

William V. Baxter and Ming C. Lin
University of North Carolina at Chapel Hill
Department of Computer Science
Chapel Hill, NC, USA
{baxter,lin}@cs.unc.edu
http://gamma.cs.unc.edu/BRUSH

Abstract

We present a flexible modeling approach capable of realistically simulating many varieties of brushes commonly used in painting. Our geometric model of brush heads is a combination of subdivision surfaces and hundreds of individual bristles represented by thin polygonal strips. We exploit bristle-to-bristle coherence, simulating only a fraction of the bristles and using interpolation for the remainder. Our dynamic model incorporates realistic physically-based deformation, including anisotropic friction, brush plasticity, and tip spreading. We use an energy minimization framework with a novel geometric representation of the brush head to generate a wider variety of brushes. Finally, we have developed an improved haptic model that provides realistic force feedback, directly related to the results of the brush dynamic simulation.

Using this model, we are able to simulate a wide range of brush styles and create an excellent variety of strokes such as the crisp, curvy strokes of Western decorative painting, or rough scratchy strokes like certain Oriental calligraphy. We have also developed an exporter for a popular free 3D modeling package that makes it easier for non-programmers to create any desired style of brush, real or fanciful.

1. Introduction

"There is no item of greater importance to the successful execution of a painting than a sufficient quantity of the very-highest-grade brushes that it is possible to find." [7]. So begins Mayer's discussion of brushes in *The Artist's Handbook*. Having the proper brushes is critical to good painting. A good set of brushes can enable a competent artist to quickly create virtually any effect he or she can imagine, from the intricate detail of cresting waves, leafy trees and delicate flower petals, to wispy billowing clouds, and the subtly blended shifting hues in a sunset. Digital artists can benefit greatly from having this expressive power available in digital painting programs.

Figure 1. A painting created using our new brush models. Notice the subtle shading and shape of the petals which were each created with a single, complex stroke.

Though there are many types of brushes commonly used in painting (see Figure 2), they share certain properties that make them effective tools for applying paint to a surface in accordance with an artist's intentions. Artistic references such as [7] describe desirable attributes of high-quality brushes with terms such as "elasticity", "durability" and "ability to maintain a point". For a virtual brush, however, the greatest challenge is capturing the most basic physical attributes such as stability, passivity, interactivity, and proper reaction to friction, which can all be taken

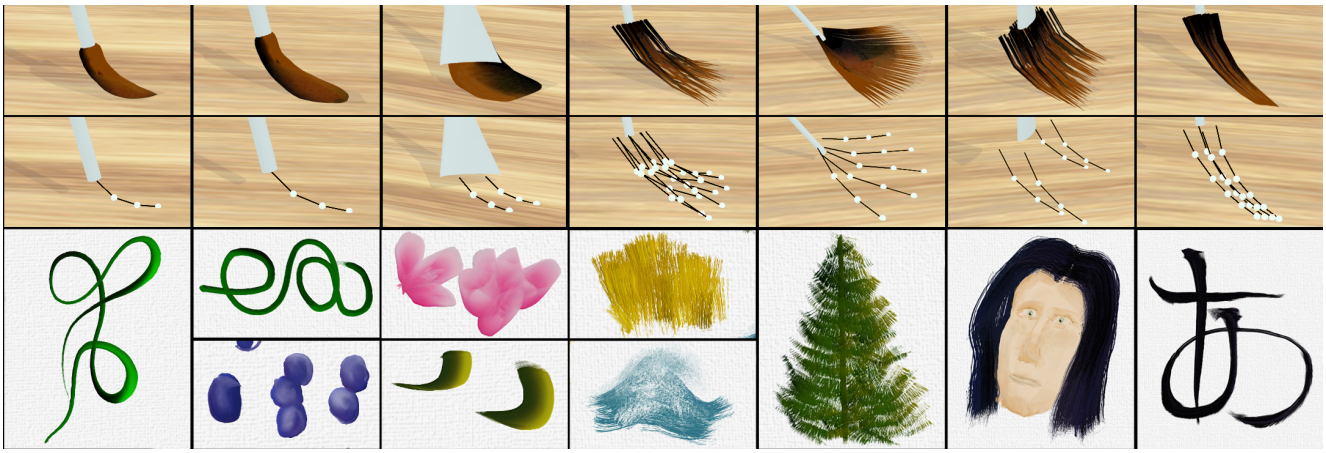


Figure 2. Some of our brushes, their spine models, and example strokes made with each.

for granted in the physical world.

Main Results: Brushes are generally very stiff dynamical systems, due to a high stress-to-mass ratio. It is difficult to simulate them with stability and accuracy using time-stepping integration techniques. We simulate the dynamics of brushes using an optimization-based framework, similar to [6, 9]. However, we incorporate a versatile brush construction methodology and introduce individual bristle dynamics to achieve the detailed bristle effects. The underlying dynamic model for a brush is created from multiple optimization-based spine primitives. The geometry of the brush head is then created out of a combination of subdivision surfaces and thin polygonal strips. The deformation of the spines determines both the motion of the subdivision surfaces as well as the strips. With this approach, brushes composed of a subdivision surface and hundreds of strips can be simulated at interactive rates. The key characteristics of our brush model include:

- A complete set of high-quality virtual 3D brushes for the digital studio, including models for rounds, flats, brights, filberts, badger blenders, and a fan brush, capturing behavior of both Oriental and Western-style brushes;
- A geometric representation empowering easy creation of fine bristle features and allowing for both smooth, clean strokes and rough, scratchy strokes;
- A constrained dynamics framework capable of handling extreme deformations like bristle splaying and modeling anisotropic friction and brush plasticity;
- A novel force model for generating haptic feedback directly from the brush deformation;
- Versatile modeling tools for enabling non-programmers to produce almost any other shape or type of brush desired.

Several amateur artists have used our new realistic virtual brush models to create paintings exhibiting complex brush strokes and fine bristle detail.

Organization The remainder of the paper is organized as follows. We first review related work, then describe our dynamic model for brushes, then our geometric model, and finally the haptic model. We conclude with demonstrations of strokes and paintings created with our new brush model.

2. Related Work

Recently several researchers have proposed physically-based 3D brush models for use in interactive image creation. With the exception of [4], the focus of each has been one particular type of brush, the round brush used in Chinese calligraphy and Oriental ink paintings.

The first fully physically-based 3D brush model was the calligraphy brush proposed by Saito [9, 10]. Saito used energy optimization on a single brush spine to determine the brush deformation. The geometric model for the brush surface was defined by sweeping circles of diminishing radius along the spine. His model accounted for spring stiffness, friction, and kinetic energy.

The “Virtual Chinese Brush” of Chu and Tai [6] delivers a very convincing model for Chinese calligraphy that includes factors such as plasticity, tip spreading, and “pore resistance” (the tendency of the bristles to get stuck on the rough surface of the paper). Like Saito, Chu and Tai also used optimization for the brush dynamics, but with a more elaborate internal structural model with lateral springs to model spreading. Their surface model is a swept surface of ellipses of diminishing radius.

The Chinese calligraphy brush model in [11] represents the brush head as a collection of individual NURBS tufts, which can dynamically and recursively split into smaller

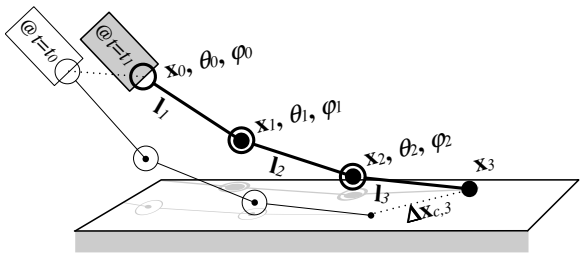


Figure 3. A single brush spine structure shown at two time steps, t_0 and t_1.

NURBS tufts based on brush deformation. The dynamic model consists of heuristics for modifying surface control parameters based on brush position history. Realistic stroke results are achieved by a technique which seems to be more image-based than physically-based, and which requires that the computer be trained using a number of real example strokes in order to generate a particular type of mark.

Baxter, *et al.* [4] proposed a flexible modeling approach for a wide variety of brushes using a semi-implicit dynamics model and subdivision surfaces. They mention, however, that their simulation technique required making some undesirable approximations to the physics, which has the side effect of making the actual forces from the simulation unusable for haptics.

Our brush dynamics model is based on optimization techniques like some previous work, but we focus on generating the wider variety of brushes that are found in Western painting with a versatile multi-spine modeling approach. We also have developed a brush model that uses both surface and strip representations for the brush head geometry. This approach allows us to capture both the smooth, neat strokes in which individual bristles play a minor role, as well as the random, scratchy strokes which depend completely on the effects of individual bristles. Finally, with the exception of [4], which used a very simple linear spring model computed independent of the brush dynamics, none of the previous work has proposed a haptic model for use in brush simulation.

3. Brush Dynamics

The transitory behavior of a typical brush subjected to an impulse force is very brief. Stated another way, given an externally applied force system, a brush will reach equilibrium very rapidly. Thus we can approximate the dynamics by solving a static equilibrium problem every time step.

Our basic dynamic model is similar to that in [6] or [9]. We start with a spine model composed of a number of segments as in Fig. 3. The bending of each joint i in the kinematic chain is described by two angle parameters, θ_i and ϕ_i.

E	Energy of system
E_f	Energy lost to friction
E_s	Spring potential energy
θ_i, ϕ_i	Joint angles
$\mathbf{\Theta}, \mathbf{\Phi}$	Vectors of all joint angles
μ	Coefficient of friction
$\mathbf{x}_i$	Cartesian coordinates of joint i
$\mathbf{l}_i$	Local coordinates of the end of segment i
$\Delta\mathbf{x}_{c,i}$	Change in contact point i
$F_{n,i}$	Normal force at contact point i
K_i	Stiffness of spring i
D_i	Damping constant of spring i
$\beta(\theta_i, \phi_i)$	Total bending angle of spring i
$\mathbf{x}_p$	Point on planar constraint surface (canvas)
$\hat{\mathbf{n}}_p$	Normal of planar constraint surface (canvas)

Table 1. Summary of mathematical notation.

At every step we minimize the total energy function:

$$E(\mathbf{\Theta}, \mathbf{\Phi}) = E_s + E_f + E_d \tag{1}$$

where

$$E_s(\mathbf{\Theta}, \mathbf{\Phi}) = \sum_i K_i \beta(\theta_i, \phi_i)^2/2 \tag{2}$$

$$E_f(\mathbf{\Theta}, \mathbf{\Phi}) = \sum_i \mu |F_{n,i}| \|\Delta\mathbf{x}_{c,i}\| \tag{3}$$

$$E_d(\mathbf{\Theta}, \mathbf{\Phi}) = \sum_i D_i |\Delta\beta_i| \tag{4}$$

subject to

$$(\mathbf{x}_i - \mathbf{x}_p)\hat{\mathbf{n}}_p \geq 0 \tag{5}$$

Please refer to Table 1 and Figure 3 for a summary of the mathematical notation used in this paper.

3.1. Virtual Work and Optimization

The principle of virtual work can be used to solve for the static equilibrium of a system via minimization. The virtual work done by all external active forces on a mechanical system in equilibrium equals the corresponding change in the total potential energy of the system for any and all virtual displacements consistent with the constraints [8].

$$\delta E_{\text{potential}} = \delta W_{\text{external}} \tag{6}$$

or

$$\delta(E_{\text{potential}} - W_{\text{external}}) = 0 \tag{7}$$

Which is to say the variation of the total energy with respect to an infinitesimal change in configuration is zero. By integrating these equations and analyzing the derivatives in the

neighborhood of the critical points, one arrives at the conclusion that stable equilibria coincide with energy minima.

For a constrained system in which we cannot easily express the "consistent displacements" in terms of a minimal number of degrees of freedom, we can express the constraints using Lagrange multipliers and scalar constraint functions of the form $C(\mathbf{q}) = 0$. The augmented objective function to minimize is then

$$L(\mathbf{q}) = E(\mathbf{q}) + \sum_{i=1}^{n} \lambda_i C_i(\mathbf{q}) \tag{8}$$

with

$$C_i(\mathbf{q}) = 0, \quad 1 \leq i \leq n. \tag{9}$$

Given an optimal solution, $\mathbf{q}^*$, The unknown constraint forces can then be recovered with the expression $\sum_{i=1}^{n} \lambda_i \nabla C_i(\mathbf{q}^*)$. The constraint forces need not be included directly in the energy function because they do no work, and hence do not add or remove energy from the system.

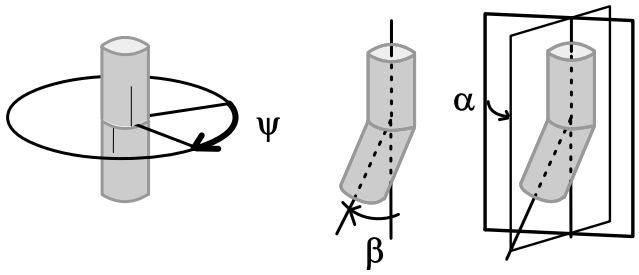
Our solution procedure involves the following steps:

1. Move the brush handle from the initial position $\mathbf{b}_0$ to its new position $\mathbf{b}_1$, according to user input from the input device.
2. Move all bristle spines rigidly with handle, not changing any joint angles.
3. Solve optimization problem for each spine, enforcing constraints at this new position.
4. Repeat from Step 1.

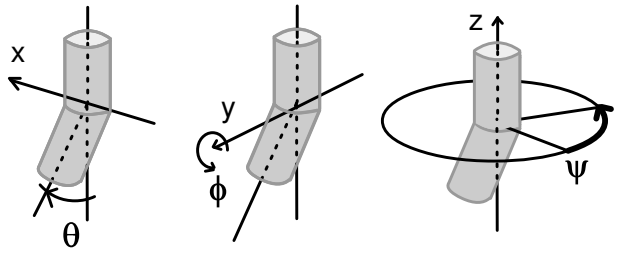
Since we first rigidly translate the bristles with the handle, the optimization essentially can be seen as a backward search for the closest configuration consistent with the constraints and in which spring forces were equal and opposite to friction forces.

3.2. Spine Kinematics

Our kinematic definition of a brush spine differs from that of [6] in that we use angles from the Euler XYZ angle set (Fig. 4(b)) rather than the ZYZ set (Fig. 4(a)). Since the amount of twisting which occurs in real brushes is limited, our parameterization uses only the first two angles, θ and ϕ. All Euler angle parameterizations for rotations have singularities, but where these singularities occur differs. With ZYZ angles a singularity occurs at the point where the angles are all zero, which is the rest configuration of the brush. Singularities are a problem for the optimizer because gradients evaluated at the singularity are essentially noise. With XYZ angles the singularity is on the horizon at 90°, much less likely to interfere.



(a) Euler ZYZ Bristle Angles



(b) Euler XYZ Bristle Angles

Figure 4. Angle parameterizations. We use the θ, ϕ of the XYZ angles because they are singularity-free in the rest configuration ($\theta = \phi = 0$).

Given this parameterization, the rotation matrix for a segment $i+1$ with respect to its parent i is given by

$${}^{i}\mathbf{R}_{i+1} = \begin{pmatrix} \mathrm{c}\phi & \mathrm{s}\phi\mathrm{s}\theta & \mathrm{s}\phi\mathrm{c}\theta \\ 0 & \mathrm{c}\theta & -\mathrm{s}\theta \\ -\mathrm{s}\phi & \mathrm{c}\phi\mathrm{s}\theta & \mathrm{c}\phi\mathrm{c}\theta \end{pmatrix} \tag{10}$$

where c and s are used as abbreviations for sine and cosine of angles. And the full expression for a point $\mathbf{p}$ in terms of its parent frame is

$${}^{i}\mathbf{p} = {}^{i+1}\mathbf{p} + {}^{i}\mathbf{R}_{i+1}\mathbf{l}_{i+1}, \tag{11}$$

By composing such transforms recursively we can compute the Cartesian positions $\mathbf{x}_i$ of each joint in world space. We also must take derivatives of these expressions to hand to the optimizer. For more details on the derivations necessary for analytical gradient calculations in the optimizer, please see [2].

3.3. Spring Energy

The simplest term in the energy function comes from the potential stored in the springs. The energy is a function of

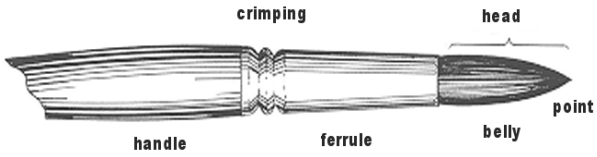


Figure 5. The nomenclature for parts of a typical brush.

the total deflection from the vertical. The deflection angle can be computed as

$$\beta(\theta, \phi) = \cos^{-1}(\mathrm{c}\theta\mathrm{c}\phi). \tag{12}$$

Several factors make real brushes stiffer near the ferrule than the point (see Figure 5). First, the individual hairs in a brush head are thinner at the tip than at the base, making them naturally less stiff near the point. Second, the tight packing of the hairs within the ferrule stiffens the bundle near the base. Finally, in some brushes, not all the hairs extend all the way to the tip of the brush, also leading to less stiffness at the point. To account for this variable stiffness, we set the K_i values higher near the base.

3.4. Friction Energy

For the frictional model, we use a modified Coulomb law. First, we consider the frictional force to act only on joints that are in contact with the canvas. The force has magnitude $\mu|F_n|$, and this is approximated as a constant over the course of one optimization. We also simplify the problem by assuming that the motion of the joint over the surface can be approximated as a straight line. Given a change in surface contact position $\Delta\mathbf{x}_{c,i}$, then the work done by friction is as shown in Eq. 3.

In analyzing the work done in a virtual displacement, the work of friction is considered negative. Ignoring E_d for a moment, plugging in $-\delta E_f$ for the external work in Eqn. 7 and E_s for the potential energy we get $\delta(E_s + E_f) = 0$, as the equation of equilibrium. In order for this to be a *stable* equilibrium it should be a minimum rather than a maximum of the energy function.

Technically the frictional function does not meet the requirements for most numerical minimization techniques, since they rely on the differentiability of the objective function, but E_f is not differentiable at the point where $\Delta\mathbf{x}_{c,i} = 0$. This characteristic can cause difficulty for some optimization techniques; however, an SQP solver that approximates the second derivative of E with finite differences (a BFGS Hessian approximation) works well enough for our purposes. When evaluating gradients, we report 0 as the friction gradient at the apex of the cone. The BFGS finite difference approximation of the Hessian tends to smooth out the second derivative locally, which at worst allows the contact point to slip slightly when friction would otherwise cause it to stick. The fix is to use set-valued derivatives (*subdifferentials*) instead of ordinary differentials, but this introduces high overhead.

3.4.1. Stiction The friction model above does not account for the frequently observed stiction effect in which the coefficient of static friction, μ_s is greater than kinetic or sliding friction μ_k. In order to incorporate this effect, we have implemented a simple solution. When in the sticking state, first solve the optimization problem using μ_s. If the solution indicates $\Delta\mathbf{x}_{c,i} > 0$, then switch to the sliding state and rerun the minimizer using μ_k. When in the sliding state, if $\Delta\mathbf{x}_{c,i} < \epsilon$ then we switch to the static state for the start of the next optimization. Hysteresis can also be used in determining the transition thresholds.

3.4.2. Anisotropy The tips of bristle hairs are more likely to get caught in the tooth of the painting surface when pushed as opposed pulled over the paper. We can incorporate this effect simply and efficiently by incorporating an anisotropy term in the calculation of the friction. We have devised a simple anisotropy term that can be combined with E_f. The advantages of this function are that it has $C1$ continuity and we can compute its derivatives analytically. We will use $\hat{\mathbf{x}}$ to represent the unit vector $\mathbf{x}/\|\mathbf{x}\|$ in what follows. Our modified anisotropic energy function is given by:

$$E_f = (1 - \eta)\mu|F_n|\|\Delta\mathbf{x}_{c,i}\| \tag{13}$$

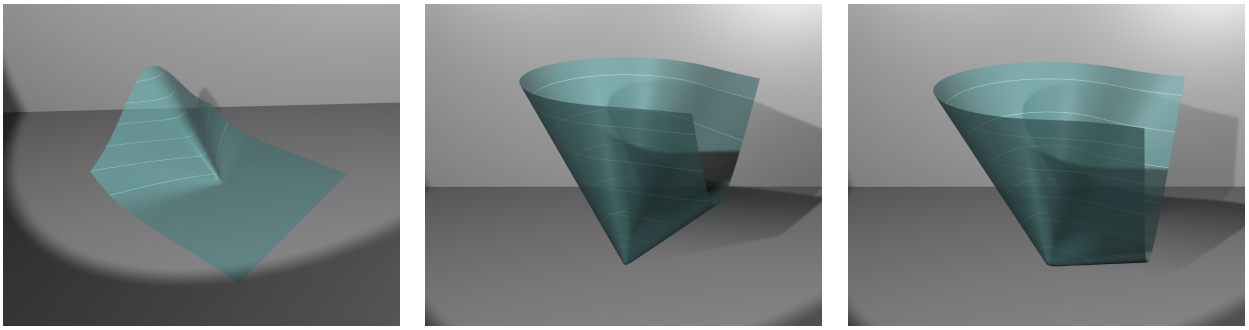
where

$$\eta = C_\eta \max\left(0, \mathbf{d}_p \cdot \frac{\boldsymbol{\Delta}\mathbf{x}_{\mathbf{c,i}}}{\|\boldsymbol{\Delta}\mathbf{x}_{\mathbf{c,i}}\|}\right)^k,\ C_\eta \in [0, 1]. \tag{14}$$

$\mathbf{d}_p$ is the preferred direction, i.e. the direction of minimal resistance (the "pull" direction) for the bristle. Most readers will recognize the resemblance of this expression to that of the intensity of a Blinn-Phong of a specular highlight. It turns out this function works quite well for anisotropic friction as well. Examples of the overall Coulomb friction energy for different values of C_η are shown in Fig. 6. The C_η and k constants give us an intuitive way to control the anisotropy. $C_\eta = 0$ recovers the isotropic case, and $C_\eta = 1$ removes all friction in the preferred direction. The k determines how sharply focused anisotropy will be, just as it does in computing specular highlights.

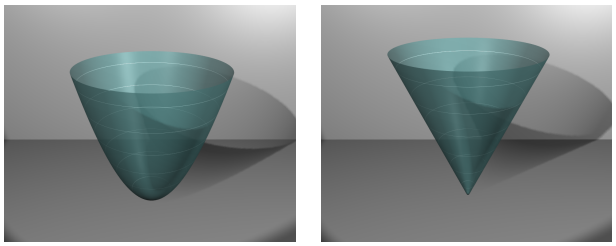
3.5. Damping and Plasticity

Another type of friction which affects some brushes significantly is internal friction and drag between wet bristles.



(a) The anisotropic term by itself. (b) $C_\eta = 0.5$ (c) $C_\eta = 0.8$

Figure 6. Anisotropic Coulomb frictional energy function. The function (a) is subtracted from the function in Fig. 7(b) to get (b). (Note, orientation of the the preferred direction is reversed in (a) to better display the geometry of the anisotropy).



(a) Spring potential, E_s (b) Friction work, E_f

Figure 7. Some energy terms that make up the objective function. Note that the horizontal axes of (a) are joint angles, while in (b) they are X and Y components of $\Delta \mathrm{x}_{c,i}$ for one joint.

We can model this as a constant resistance in joint space:

$$\mathbf{F}_{d,i} = -D_i \mathrm{sgn}(\Delta\beta_i) \tag{15}$$

and incorporate it as part of the overall energy as shown in Equation 4.

The effect of adding this internal friction term is that a deformed brush does not return all the way to its starting point, since the friction forces are greater than the spring bending forces for small angles. In effect, this works as a simple model for brush plasticity. Plasticity is achieved by other means in [6], but using joint friction is both simple to implement and perhaps more closely related to the actual physical mechanism behind brush plasticity.

3.6. Constraints

To prevent the joints of the brush spine from penetrating the canvas, we subject each to an inequality constraint that expresses that the Cartesian joint position must be outside the plane of the canvas, as given in Eq. 5. Note that $\mathbf{x}_i$ is a nonlinear function of θ_j and ϕ_j for $1 \leq j \leq i$, given by the forward kinematic Equations 10–11.

Our SQP minimizer uses an active set approach, in which inequality constraints are treated as equality constraints while active, and simply ignored when not active [1]. This approach is efficient in that the dimensionality of the augmented objective function is reduced when few constraints are active, leading to smaller matrix equations to solve.

4. Geometric Modeling of Brushes

Our geometric model for brushes can take advantage of both an explicit surface representation as in previous work [4, 6, 9] but also a strip based method for representing the brush as individual hairs. A brush can use either one of these or both. The surface-based representation has the advantage of giving a solid, consistent footprint; however, it is difficult to capture bristle spreading effects and the stippling effects created by individual bristles with just a surface representation.

4.1. Subdivision Surface

Our surface representation is a subdivision surface similar to that used in [4]; however, we use an approximating Catmull-Clark surface instead of the interpolating Butterfly surface used there. As was noted in that paper, interpolating subdivision surfaces can develop unnatural kinks which

are undesirable. The challenge with approximating surfaces is that it is more difficult to place control vertices to achieve a desired shape. We have dealt with this issue by creating an export tool for a popular, free 3D modeling package, Blender. This package enables users to visually create brush surfaces interactively of any shape and with any properties desired.

The control vertices that determine the shape of the subdivision surface are placed using standard matrix skinning techniques. The skinning weights and brush spines are also set up from within Blender using its built-in skeletal animation tools.

4.2. Bristle Strips

A typical brush head can be composed of as many as a few thousand hairs. The most accurate model of a brush would involve simulating each one of these individually including full hair-to-hair collision response, which is not currently feasible in real time. Nevertheless, the effects of individual bristles are important to capture. Fortunately the motions of individual bristles show a large amount of coherence – neighboring bristles tend to move more or less in the same direction – so we take advantage of this by simulating only a few brush spines (generally less than ten), and interpolating as many as hundreds of other bristles from the motion of those spines.

Geometrically each hair is most like a thin, tapered tube; however, tubes require many vertices to specify. Instead we use strips of quadrilaterals as the primitive since they require just two vertices per joint and are quick to render in modern hardware. An alternative would be to use line strip primitives, which would only require one vertex per joint, but line widths on graphics hardware can only be specified in image space as an integral number of pixels. Thus the density of a set of line strips changes depending on the resolution of the canvas being painted upon. If hardware allowed for true lines with geometric widths, this would be the best representation.

4.2.1. Paint Transfer: We determine the paint transfer for the surface representation just as in [4]; however, a few modifications are necessary to handle strips properly. The basic process is the same: the paint and attributes are texture-mapped onto the brush geometry (we use one row of a texture map per strip) and then a contact footprint is determined by rendering the texture-mapped brush with an orthographic projection from the canvas's point of view. A clipping plane near the surface of the canvas culls out the portions of the brush head that are not in contact.

The first issue with strips is that when rendering the brush footprint, the flat side should always face the canvas. In this respect the strips are like viewer-oriented impostors or billboards, with the "viewer" being the canvas. We perform this step on the CPU every paint transfer step. A small vertex program on the GPU could also be used to perform this computation.

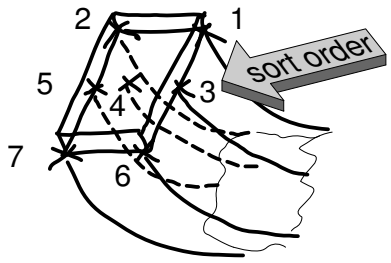


Figure 8. Determination of rendering order for paint transfer. The average bend direction of the spines determines the order in which bristles are sorted and rendered. In the example above the seven bristles shown would be rendered according to the numbering shown. This ordering ensures the brush will maintain consistent separation between front and back sides of the brush during paint transfers.

The second issue is rendering order. In a real brush, hair-to-hair collision interactions prevent hairs on one side from collapsing through to the other side of the brush. In order to avoid the cost of explicitly calculating this $O(n^2)$ interaction, we instead determine an approximate back-to-front rendering order with respect to the canvas using a simple heuristic. We first calculate a principal bend direction for the brush head as the average of all the brush spine tip deflection vectors: $\sum_{i=1}^{N} \Delta\mathbf{x}_{i,\text{tip}}/N$ (see Fig. 8). We then sort the strips by their root positions along the principle bend direction projected onto the brush cross-sectional plane. In the difficult case when all the hairs are bent in the same direction, and many are overlapping, this approach performs quite well. When the spines are splayed out in different directions the ordering may not be consistent with actual depth, but in that case there is little hair-to-hair collision to begin with, so rendering order is not important.

4.2.2. Bristle Dynamics Interpolation: In creating a new brush model, positioning 100 or more bristle strips would be tedious, so we have devised an algorithm for automatic random placement. First, we compute an oriented bounding box (OBB) around the root points of the brush spines. From the length of the edges of the OBB we determine whether sampling should be performed over a 1D or 2D space. If 1D (i.e. the spine roots are all contained in a very narrow bounding box) then we parameterize the best fit line and place strip roots randomly

along that line. If 2D, we compute a Delaunay triangulation of the brush spine roots and then place strip roots randomly within those triangles.

At run-time the geometry of an interpolated strip is computed by a convex weighted sum of spine positions. Each spine is parameterized as $\mathbf{p}_i(s), 0 \leq s \leq 1$ such that $\mathbf{p}_i(0)$ is the root of the ith spine and $\mathbf{p}_i(1)$ is the tip. Then an interpolated bristle is given by:

$$\mathbf{p}_{\text{interp}}(s) = \sum_i w_i \mathbf{p}_i(s) \tag{16}$$

where $\sum_i w_i = 1$ and $w_i \geq 0$.

Given a randomly chosen root location $p_{\text{interp}}(0)$, we need to determine an appropriate set of non-negative weights w_i, which sum to unity and result in the desired position. Since we are already using a constrained SQP solver, it is convenient to use minimization to compute this set of weights as well. We can encode the preference that nearby spines should have higher weights than distant spines in the objective function.

We find that among the several objective functions we tried, that minimizing the following weighted least-squares function for w_i yields good results:

$$\sum_i \|\mathbf{p}_{\text{interp}}(0) - \mathbf{p}_i(0)\|^2 w_i^2 \tag{17}$$

subject to

$$\sum_i w_i = 1 \tag{18}$$

$$w_i \geq 0 \tag{19}$$

$$\mathbf{p}_{\text{interp}}(s) = \sum_i w_i \mathbf{p}_i(s) \tag{20}$$

The coefficients on the w_i are small for spines *near* the interpolated bristle and large for spines *far away*, thereby leading to the opposite trend in the w_i values themselves. That is, an interpolated strip is influenced most by the spine closest to it.

5. Haptic Feedback

We use the SensAble Technologies Phantom™ haptic input device for controlling the 6DOF position and rotation of the brush. Other researchers have obtained 6DOF input via tracking devices attached to real brushes [6], which has certain advantages. But the two major disadvantages are that a) the simulation can get out of phase with the actual deformation of the brush leading to haptic feedback that is completely wrong for the visual model being shown, and b) the properties of a physical brush are fixed and cannot easily be changed to match the properties of the virtual brush, such as the length or stiffness of the brush hairs.

The difficulty when developing a haptic model to display the results of a 30-60Hz simulation is that the haptic thread needs to run at 1KHz. In [4], to get around this problem, the haptic feedback was based on a fast spring model, independent of simulation results. But this approach makes limited use of the haptic feedback device's ability to recreate an accurate haptic sensation of the virtual brush. Another approach is to take force samples from the brush simulation and filter them, either via virtual coupling or a traditional signal processing approach. Our approach is as follows: instead of sending a particular force sample to the haptic system, we send a higher-order representation of the local force landscape, which can be evaluated quickly by the haptic thread to generate forces at 1KHz.

The local force model is essentially a Taylor expansion of the force near a particular point. We evaluate one such force model for each brush spine and use the average as the force for haptic feedback. The basic force expansion is

$$\mathbf{F}(\mathbf{p}_0 + \Delta\mathbf{p}) = \mathbf{F}_0 + \tilde{\mathbf{F}}(\Delta\mathbf{p}), \tag{21}$$

where F_0 is the base force determined directly from the current bend in the springs. The $\tilde{F}$ term incorporates the sum of two main components. The first is a compression term, which is strongest in the direction in which the spine is bent, and the second comes from force change due to the bending of the springs:

$$\tilde{\mathbf{F}}(\Delta\mathbf{p}) = \tilde{\mathbf{F}}_{\text{compress}}(\Delta\mathbf{p}) + \tilde{\mathbf{F}}_{\text{bend}}(\Delta\mathbf{p}) \tag{22}$$

Both components are computed using linear approximations based on the current configuration of the spine, the compression term using a significantly larger coefficient than the bending term.

Note that since we are using actual force values from the brush simulation for haptic display, it is necessary for our brush model parameters to have physically meaningful scaling. We estimated the magnitude of reasonable spring stiffness by considering the that a brush under 1-2 Newtons of normal force should bend nearly 90°. From there we can calculate how much torque would be necessary to generate that force over a lever arm the length of a brush head. The result is that the stiffness constants, K_s, should be on the order of 5-10 N·mm/rad. With these stiffnesses, reasonable surface friction coefficients, μ, are about 0.3-0.5, and internal friction constants, D_i, are a maximum of about 0.01. Using these values, the forces generated for haptic feedback are also reasonable without need for additional scaling.

6. Implementation and Results

We have implemented our brush model on a 1GHz Pentium IV™ desktop, and and incorporated our brush model into an existing painting system which features several paint models [3, 4, 5]. We implemented an object-oriented C++

SQP optimizer based on the C code accompanying [1]. We have found 5 iterations of the optimizer to be sufficient for a 3-joint brush spine. This takes about 100 microseconds, allowing us to simulate up to approximately ten of these spines interactively. The cost of the interpolation bristles is very small, allowing us to interactively use as many as 200 bristle strips without difficulty.

As a test of the effectiveness of the brush model we have attempted to recreate stroke samples included as training materials for a popular decorative painting technique called One-Stroke™ (http://www.unctv.org/onestrokepainting/). The One-Stroke techniques emphasize using the flexibility of the brush and its complex loading to create organic shapes such as flowers and birds quickly and with a high degree of realism using very few strokes.

In Figure 9, we compare the results of recreating the strokes with our new brush model versus with the model presented in [4]. As can be seen in Figures 10–12 the new brush model is capable of creating detailed bristle marks that cannot be captured by a surface representation alone.

For a complete interactive demonstration, please see the accompanying video. For further details on the implementation and derivations of various equations used, please see our technical report [2].

7. Conclusions

We have presented a new versatile model for virtual brushes that can reproduce the wide range of effects and styles needed by the digital painter. The combination of subdivision surfaces and strips allows for both smooth, detailed strokes, as well as rough scratchy brush work in which individual bristle details can be seen. We have demonstrated here and in the accompanying video that linear interpolation of strips based on a small number of spines is sufficient to yield a realistic overall deformation of the brush head, and that brushes with hundreds of hairs can easily be simulated interactively with these techniques.

8. Acknowledgments

The authors would like to thank all the artists for their paintings, and the members of the Blender community for their excellent free 3D modeling program and helpful suggestions on the Blender forums.

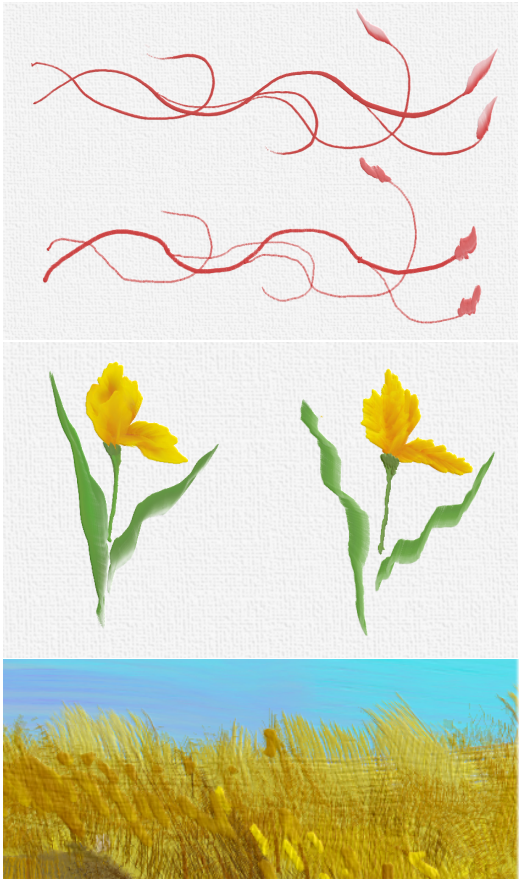


Figure 9. *(Top and middle)* A comparison of strokes made with the optimization-based brush model versus our previous model. Note the smoother path and width variations in the new strokes (top row and left side, respectively). *(Bottom)* These strokes are impossible to achieve with the old brush model.

References

[1] S. K. Agrawal and B. C. Fabien. *Optimization of Dynamic Systems*. Kluwer Press, 1999.

[2] W. V. Baxter. Notes on brush simulation with optimization. Technical report, University of North Carolina at Chapel Hill, 2004.

[3] W. V. Baxter, Y. Liu, and M. C. Lin. A viscous paint model for interactive applications. Technical Report TR04-006, University of North Carolina at Chapel Hill, 2004.

[4] W. V. Baxter, V. Scheib, and M. C. Lin. dAb: Interactive haptic painting with 3D virtual brushes. In E. Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 461–468. ACM Press / ACM SIGGRAPH, 2001.

[5] W. V. Baxter, J. Wendt, and M. C. Lin. IMPaSTo: A realistic model for paint. *To appear in Proceedings of NPAR 04*, 2004.

[6] N. S. Chu and C. L. Tai. An efficient brush model for physically-based 3D painting. *Proc. of Pacific Graphics*, Oct 2002.

[7] R. Mayer. *The Artist's Handbook of Materials and Techniques*. Penguin Books, Ltd., 4th edition, 1985.

Figure 10. A painting created using both surface and strip based brushes.

Figure 11. A painting created with our new strip brush models. Notice the detailed, scratchy bristle marks in the clouds and grass.

Figure 12. A painting created using our new brush models.

[8] J. L. Meriam and L. G. Kraige. *Engineering Mechanics: Statics*. John Wiley & Sons,Inc., 3rd edition, 1992.

[9] S. Saito and M. Nakajima. 3D physically based brush model for painting. *SIGGRAPH99 Conference Abstracts and Applications*, page 226, 1999.

[10] S. Saito and M. Nakajima. Physically based 3D brush model for interactive painting. *Jyouhou-Shori Gakkai Ronbunshi (A Japanese Journal)*, 41(3):608–615, 2000.

[11] S. Xu, F. Lau, F. Tang, and Y. Pan. Advanced design for a realistic virtual brush. *Computer Graphics Forum*, 22:533–542, September 2003.