5

Performance Enhanced Virtual Hairy Paintbrush System

5.1 Overview

This chapter presents a new approach to modeling a 3D physical paintbrush, based on which an interactive painting system has been developed. Compared with existent brush-based painting systems, our new system can accurately and stably simulate the complex painting functionality of a running brush using a modest amount of system resources. The detailed modeling empowers the user to create high-quality digital paintings with delicate aesthetic details that can rival real artwork. With the amount of details to be modeled, we have to rely on a hierarchical modeling approach, dividing the modeling tasks to on-line and off-line computations, and a powerful pigment model that is fully integrated into the brush model. These optimizations and special components make the system operable in real time, fully interactive, and easy to manipulate.

5.2 Introduction

A long cherished dream of both scientists and artists since the birth of digital computers is to be able to use the computer to produce beautiful art. With the ever increasing power of modern computers, the computer as a serious artistic tool is now within reach for many. The more powerful the computer would becomes, the more that can be done in perfecting the artistic effects achievable through advanced computing techniques. The recent hot pursuit of Non-Photorealistic Rendering (NPR) is one major effort to capitalize on the power and flexibility of today's computers for art creation that can surpass what is possible by human artists with conventional means. One major research area in the NPR field is emulation by pen-and-ink [SABS94] which has wide applicability and could lead to powerful expression in many situations. In the genre of Chinese art forms, the brush takes the place of the pen, and the computerization of a paintbrush (and the ink) for interactive painting or calligraphy is a very attractive goal. A brush is many times more complex than a pen, and hence the problem presents a huge challenge for

the interested computer scientists. We describe our solution to the problem in this chapter.

Smith [Smi01] has written a good survey on the early painting systems. More recently, several researchers have successfully implemented a virtual brush (or "e-brush") that can mimic a real 3D brush for painting and calligraphy through physically-based modeling [WI00, BSLM01, CT02]. Compared with many existing 2D virtual brushes, which often require the user to specify the contour's control points and the texture of painted strokes (e.g. [Str86]), these 3D brushes are more natural to use, especially for non-computer specialists. A guiding principle for the design of an e-brush is that it must present itself as a familiar tool to the traditional human artist, because "presenting a system that requires the designer to adapt, distract, or place attention outside of the actual task will hinder the creative process to which the designer can come up with ideas" [HSS02]. With the e-brush we have implemented, the user will not be required to specify any control point or to adjust any parameter for the texture, but can operate the brush in more or less the same way as a physical brush.

In a good virtual brush design, each step of the painting process should be simulated in high realism. Here simulation refers to both the production of strokes on the virtual canvas and the continuous visual display of the running brush. The latter is necessary if the user is to be able to feel the presence of a brush in order to have full control over it and to manipulate the brush as in real life. An even more perfect virtual brush should also provide a haptic interface, like what is done in [BSLM01]. Apart from research on virtual brushes for 2D painting, designing e-brushes for 3D painting appears to be a promising area [HH90, ABL95, Pix00, KMM+02].

In this chapter we describe a new e-painting system with improved system design aiming at better performance. We particularly focus on the modeling aspects—the geometry and the dynamics of the paintbrush. Our modeling of the paintbrush stands out among existing approaches because of its unique ability to capture the highly complex geometry of a physical brush as well as its dynamic behavior during painting with great accuracy. Our modeling approach pays special attention to many very fine details that are called for by the creation of high quality digital paintings. Because of these very fine details, the system presents a "realistic" simulation of the physical brush to the user who would operate it like a real brush and expect the output to be as good as a real one.

Compared with the brush based painting system design that we've studied in Chapter 4, this chapter delves further into the underlying detailed modeling of various essential aspects of the realistic paintbrush. The level of details being addressed here is useful not only to those who want to implement a similar system, but also others who develop painting systems of different kinds, as well as those working in the more general area of NPR and physically-based modeling. Besides the low-level details, we also provide in-depth analyses and discussion to justify several high-level design decisions, and experimental results to demonstrate their effectiveness.

The chapter is organized as follows. Sect. 5.3 and Sect. 5.4 present our realistic modeling of the geometry and the dynamics of the brush respectively. Sect. 5.5 gives an overview of the new e-painting system we developed based on our modeling. Sect. 5.6 compares our system with other e-brush based painting systems. Sect. 5.7 concludes the chapter and discusses possible future tasks.

5.3 Modeling the Paintbrush's Geometry

One of the earliest paintbrush models was by Strassmann [Str86], in which brush strokes are created by sweeping a 1D brush bristle over a skeleton, and the color, width and wetness of the brush can be varied. Wong and Ip's virtual brush [WI00] is modeled as an inverse cone which can produce an elliptic drawing mark. In the DAB project [BSLM01], a subdivision surface is wrapped around a spring-mass particle system skeleton to emulate the brush surface. Most recently, Chu and Tai [CT02] use a geometrically-based approach to model an un-split brush tip and an alpha map to model a split brush tip. None of these models however is powerful enough to model a physical brush to a high degree of likeness to the physical brush's real geometry. In particular, the common situation in which a brush splits into a large number of hair bundles appears to be out of the reach of all these existing models. According to many practising artists, a feature for modeling such a high degree of splitting is very desirable. The novel e-brush modeling method we propose here can effectively deal with this and other difficult problems using little memory and CPU resources.

To model a paintbrush with the granularity of a single hair strand which is done in Wong and Ip's system [WI00] is inefficient because a typical real brush may consist of thousands,or even tens of thousands of individual hair strands. To overcome this inefficiency, in Chapter 4 we have introduced our first effort at increasing the brush system's modeling capability with careful consideration of the consumption of system resources by modeling a brush as clusters of brush hair. This approach however can only handle the case of a brush splitting into a small number of hair clusters but not one with heavy splitting.

In this chapter we propose a hierarchical brush geometric model. At a lower level of the hierarchy, hair strands whose position and geometry in 3D space are close to each other are gathered together and modeled as one *hair macro*. At the upper level, disjoint hair macros whose geometries are similar are classified into the same *cluster of hair macros*. The motivation behind having multiple levels is to eliminate as much as possible the redundancy in representing and simulating the brush hair. In real actions, a brush can easily split into thousands of disjoint clusters of hair threads. Cluster-based modeling alone,as introduced in Chapter 4, is not sufficient to deal with the highly chaotic geometry of the brush. It can be easily observed that even in such a situation there exist only a few sharply distinctive geometries among all the geometries of hair clusters. We call these distinct geometries *primitive*

geometries. With these primitive geometries, the geometries of all the hair clusters can be approximately derived via simple affine transformations. Fig. 5.1 shows some complex brush geometries of our virtual brush. Based on this two-level hierarchical representation, our virtual brush model can efficiently represent the complex geometry as well as simulate the dynamic behavior of real paintbrushes having thousands of disjoint hair clusters.

Fig. 5.1. Some complex brush geometries of our virtual brush

5.3.1 Three-layer Hierarchical Modeling

Our realistic modeling of the paintbrush geometry is organized as a three layered hierarchy: the lowest layer consists of *hair macros*, which are clusters of hair threads; the intermediate layer consists of clusters of hair macros; the highest layer is the whole brush tip bundle.

Hair strands whose positions in the 3D space are close to each other and whose geometries are similar, are modeled together as one *hair macro* which is a single, aggregative representation of both the geometry and dynamics of these hair threads. A hair macro is the smallest granularity in our modeling. Hence. the overall modeling capability of our virtual brush derives from the modeling power of a hair macro. A formal definition for our geometric model of a hair macro is given as Eq. (5.1). We construct the model of a hair macro $\mathbf{H}$ through the general sweeping operation in CAD, $GeneralSweeping(\cdot)$, by moving a variable ellipse $E(t)$ along a 3D trajectory, $K(t)$. We call the trajectory "the skeleton of $\mathbf{H}$", which is represented as a 3D B-spline. The generated skinning surface is the surface of $\mathbf{H}$ while the swept volume is the interior volume of $\mathbf{H}$. During the sweeping operation we ensure that the sweeping ellipse $E(t)$ always lies on the normal plane of the sweeping trajectory $K(t)$, i.e. $E(t)|_{t=t_0} \perp K(t)|_{t=t_0}$. See Fig. 5.2 for the illustration of a hair macro in its initial geometry, and a deformed hair macro.

$$\begin{cases} \mathbf{H} \triangleq GeneralSweeping\Big(E(t), K(t)\Big)\ (0 \leqslant t \leqslant 1) \\ E(t) \triangleq \{(x,y) | x = L(t)\nu\cos\big(w+\theta(t)\big) \\ \qquad\quad y = S(t)\nu\sin\big(w+\theta(t)\big) (0 \leqslant \nu \leqslant 1, 0 \leqslant w < 2\pi)\} \end{cases}, \tag{5.1}$$

$$\mathbf{H} \triangleq Modeling\big(K(t),\ S(t),\ L(t),\ \theta(t)\ \big)\ . \tag{5.2}$$

$E(t)$ is a variable ellipse in that the lengths of its major axis $L(t)$, minor axis $S(t)$ and its orientation $\theta(t)$ (on the normal plane of its sweeping trajectory) can all be varied during sweeping. Thus, given three B-splines,

$S(t)$, $L(t)$ and $\theta(t)$, we can uniquely determine an $E(t)$ $(0 \leqslant t \leqslant 1)$. We can also rewrite Eq. (5.1) as Eq. (5.2), which means that by four B-splines, $\big(K(t)$, $S(t)$, $L(t)$, $\theta(t)$ $\big)$, the geometry of a hair macro, $\mathbf{H}$, can be presented parametrically. This representation makes it easy and intuitive for end users to tailor the geometry of a hair macro and further customize their virtual brush if the users so choose. Fig. 5.2 (a) shows the graphical user interface to be used for this purpose.

The general sweeping operation we adopted is a suitably powerful modeling metaphor, which meets the demand for modeling long furry objects. Our representation is simple to generate and capable of capturing all kinds of geometries that a paintbrush may have, as is demonstrated in Fig. 5.2(a). In comparison, although the subdivision surface mesh employed in [BSLM01] is a more powerful modeling metaphor in general, unlike our general sweeping operation, it is not specifically customized for the modeling of brush hair. Therefore, in spite of its being more sophisticated than our method, it cannot deal with some of the more extreme cases, such as heavily deformed brush hair.

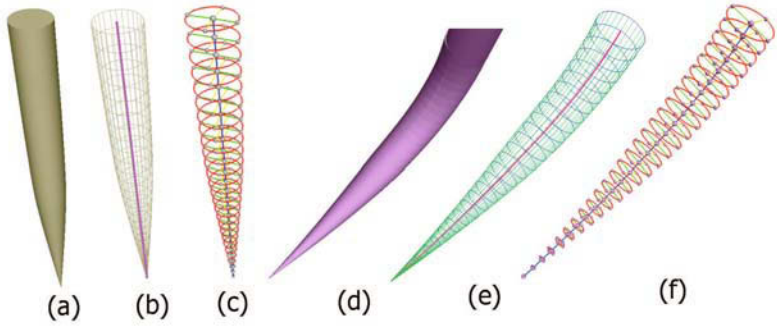


Fig. 5.2. (a to c)An initial hair macro: (a) when shaded; (b) as a wireframe with the sweeping trajectory highlighted; (c) as a series of profiles of the sweeping ellipse, which are in red. (d to f)A deformed hair macro: (d) when shaded; (e) wireframe; (f) profiles of the sweeping ellipse

General sweeping, however, is a time-consuming operation. Moreover, the internal solid models of all the hair macros together could take up a substantial amount of space, especially in situations when the brush is split into many small hair bundles. Consideration of system response time and memory resources calls for a strategy to eliminate as much as possible the redundancy in representing and simulating the brush hair. We introduce the idea of *hair macro cluster* to group disjoint hair macros,whose geometries are similar into one single,modeling unit. This is a reasonable move because it can be easily observed in real life when a brush is split into numerous disjoint hair strands or clusters, there will only be a limited number of sharply distinctive geometries among the clusters. Note that the grouping considers only the geometries but not the physical positions of the hair macros in the 3D space. Given a macro hair cluster and the geometry of any individual hair

macro in the cluster, the geometries of all the remaining hair macros of the cluster can be easily derived via simple affine transformations. This design gives rise to a brush geometry model that is compact in memory, and enables fast simulation of the brush's actions.

Fig. 5.3 shows two examples of complex brush geometry modeled using our three-layer hierarchy. Fig. 5.4 shows the brush modeling when one, two, or three layers in the hierarchy are in effect.

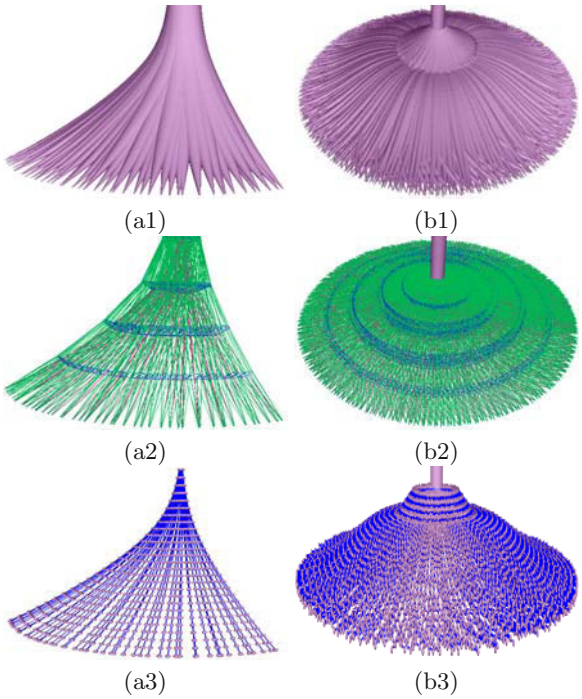


Fig. 5.3. (a1 to a3) A split virtual brush when shaded, in wireframes, and in terms of profiles of the sweeping ellipses respectively; (b1 to b3) a brush that is heavily splitting

5.3.2 Real-time Visual Display of the Brush

Visual feedback is important for any interactive system. In a painting session the user needs to feel the physical presence of the brush in order to manipulate it at will. To provide a good visual feedback could require a huge amount of computation because of the highly complex geometry of a realistic virtual brush. Our hierarchical modeling provides a solution to efficiently tackle the problem. As discussed previously, we only need to explicitly model the geometry of one hair macro for each hair macro cluster, from which all other geometries belonging to the same cluster can be derived. Thus all the hair macros in a cluster share the same data structure, and one tessellation process

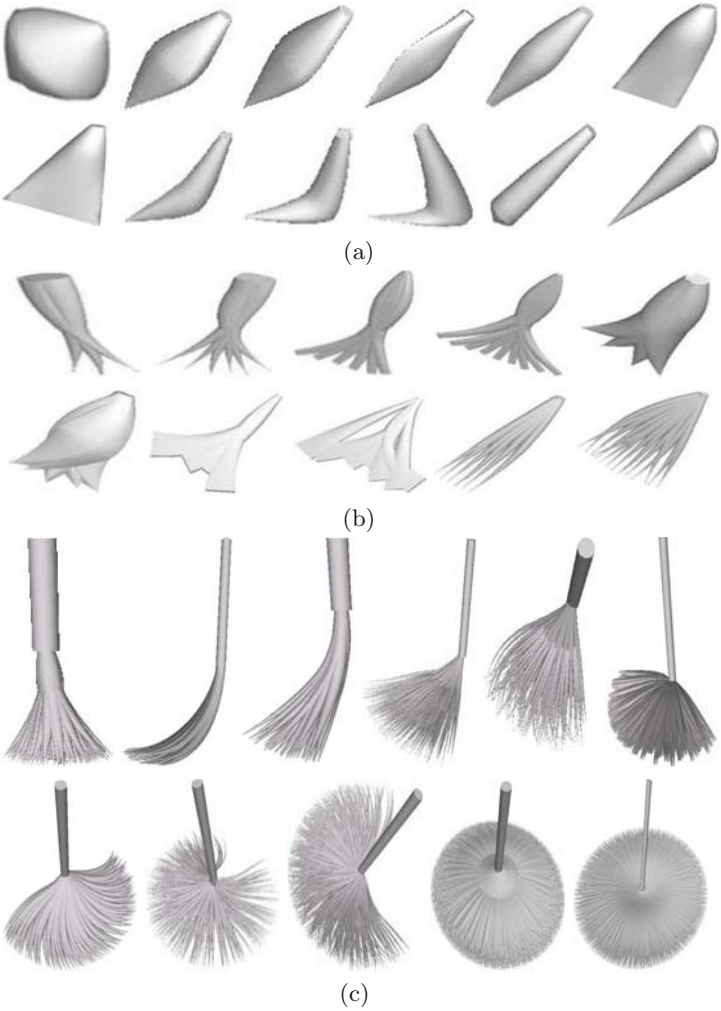


Fig. 5.4. Brush geometry modeling on three levels: (a) using only the top level—the whole brush tip bundle; (b) using two levels—the top level and the "hair macro" level; (c) using all three levels

being applied to the geometry of one hair macro is sufficient for tessellating all the hair macros in the cluster (by applying the same affine transformation as mentioned before).

We also make sure that modeling satisfies the preconditions necessary for taking advantage of the hardware acceleration facility through the "Display Lists" feature in OpenGL. The time-consuming commands for rendering the geometries of the hair macros would therefore be optimized by the driver as the affine transformations can be pre-computed. With the hierarchical modeling approach and hardware acceleration, our virtual brush can achieve

real-time visual feedback of the complex geometry of the brush using a reasonably small amount of memory and CPU time.

5.4 Modeling the Paintbrush's Dynamic Behavior

Simulating the dynamics of a real paintbrush, which includes the brush's deformation due to outer force, recovery from deformation when the force vanishes, splitting due to inner stress, etc., is a non-trivial problem because of the complexity of the brush's geometry and the physical principles that underlie the brush's behavior. In the DAB project [BSLM01], the motion of the brush geometry is simulated through a pair of first-order differential equations. The dynamics of Chu and Tai's brush [CT02] are modeled as springs whose deformation is via constrained energy minimization. In spite of all these efforts, what would be a good model of brush dynamics that supports realistic, efficient, and stable brush motion simulation using a reasonable amount of system resources remains not completely answered.

Our design of the virtual brush system offers a highly detailed dynamic modeling of the behavior of a physical paintbrush. The modeling is divided into two phases. The first phase consists of on-line computation of the computationally inexpensive and input-sensitive physical processes, such as brush deformation due to brush pressing. We adopt the approach of using a "phenomenal model" which simulates the dynamics of a changing object based on observations, instead of by the highly complex underlying physical laws that govern the changes. It proves to be a highly economical approach when we have to model a large number of brush features. The result is fast simulation of a sufficiently detailed model of the brush. With this observational modeling approach, however, we compromise some degree of modeling accuracy. And so in the second phase, off-line data are used to calibrate and refine the on-line simulated results. These data come from a simulation error calibration database constructed from off-line acquired ground truth about simulation errors. Our design represents a balance between a complete on-line based approach and one that is at the other extreme. The former demands a huge amount of runtime resources in order to achieve real-time response; the latter could result in a database which is too large to manage. This "observation model plus calibration database" approach achieves high realism for the brush dynamics being simulated as well as interactivity with little incurred computational cost. Fig. 5.5 shows the dynamic deformation of a primitive geometry. Fig. 5.6 shows a series of simulated dynamic deformations of our virtual brush.

During painting, the geometry of the paintbrush is deformed due to the outer force arising from friction between the brush and the paper. Brush deformation is modeled in the first phase of the two-phase modeling, in two different parts: deforming of hair macro due to brush-paper collision and deforming of hair macro due to the brush's inner stress.

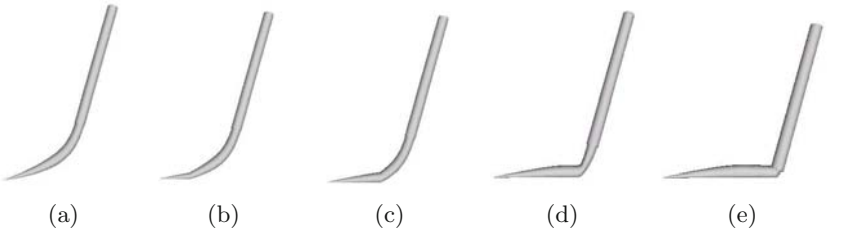


Fig. 5.5. Dynamic deformation of a primitive geometry during painting

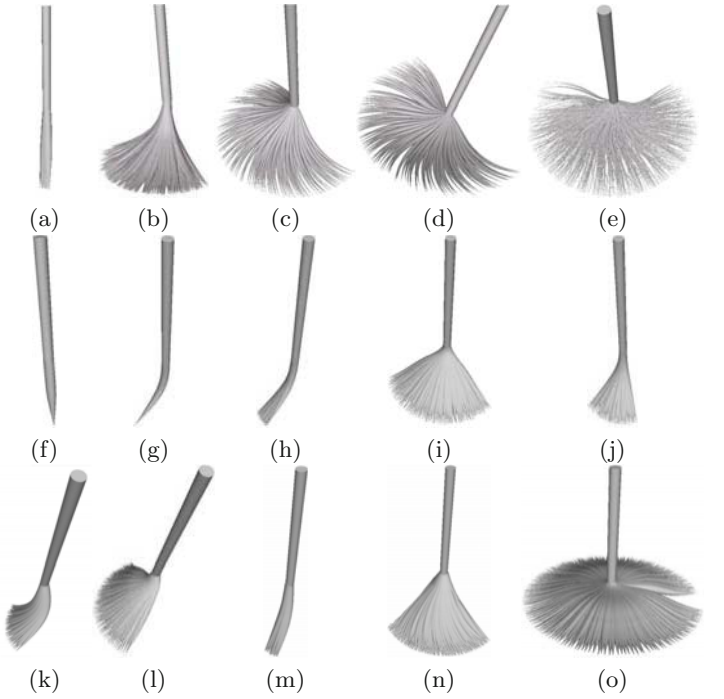


Fig. 5.6. (a) to (e)—Simple dynamic deformation of the virtual brush: (a) the initial geometry, (b) pressed down, (c) rotated, (d) tilted, (e) further pressed down and rotated. (f to o) A more sophisticated virtual brush deformation process: (f) the initial brush; (g) pressed down before splitting is simulated; (h) after splitting is simulated; (i) rotated and further pressed down; (j) lifted a bit with some split brush strands merging; (k) tilted; (l) further rotated and pressed down; (m) further lifted to the initial brush position; (n) pressed down again; (o) completely pressed down to the virtual paper with some rotation

5.4.1 Deformation due to Brush-paper Collision

5.4.1.1 Deformation of skeleton

The main constraint to observe here is that the brush's skeleton cannot penetrate the virtual paper. During brush painting or writing, if some part of a skeleton $K(t)$ penetrates the virtual paper ψ at an intersecting point $K_{\mathrm{o}}(t_{\mathrm{p}})$ between the original skeleton $K_{\mathrm{o}}(t)$ and the virtual paper ψ, we would deform $K_{\mathrm{o}}(t)$ by replacing it with a new curve $K_{\mathrm{n}}(t)$ to satisfy the above constraint. The deformation scheme is expressed mathematically as Eq. (5.3) and illustrated by Fig. 5.7.

$$K_{\mathrm{n}}(t) \triangleq K_{\mathrm{pro}}(t) \times (1 - rfc) + K_{\mathrm{ro}}(t) \times rfc, \tag{5.3a}$$

$$K_{\mathrm{pro}}(t) \triangleq \begin{cases} Pro\big(K_{\mathrm{o}}(t), \psi\big) & \text{when } K_{\mathrm{o}}(t) \text{ is below } \psi \\ K_{\mathrm{o}}(t) & \text{otherwise} \end{cases}, \tag{5.3b}$$

$$t_{\mathrm{p}} = \max\{t | K_{\mathrm{o}}(t) \text{ is below } \psi\}, \tag{5.3c}$$

$$K_{\mathrm{ro}}(t) \triangleq \begin{cases} K_{\mathrm{o}}(t) & t_{\mathrm{p}} \leqslant t < 1 \\ Rot\big(K_{\mathrm{o}}(t), K_{\mathrm{o}}(t_{\mathrm{p}}), \phi_{\min}\big) & 0 \leqslant t < t_{\mathrm{p}} \end{cases}, \tag{5.3d}$$

$$\phi_{\min} \triangleq \arg\min_{\phi} \Big(\forall P \in \{Rot\big(K_{\mathrm{o}}(t), K_{\mathrm{o}}(t_{\mathrm{p}}), \phi\big) | \; 0 \leqslant t < t_{\mathrm{p}}\} \Rightarrow P \text{ is above } \psi\Big), \tag{5.3e}$$

$$K_{\mathrm{o}}(t) \leftarrow Smooth(K_{\mathrm{n}}(t)). \tag{5.3f}$$

$K_{\mathrm{n}}(t)$ is the linear interpolation between $K_{\mathrm{o}}(t)$'s projected version K_{pro} and its rotated version K_{ro} [Eq. (5.3a)]. The weight rfc used in the above interpolation is essentially a coefficient representing the degree of rigidity-flexibility of the material used for the brush hair [Cal99]. K_{pro} is computed by simply projecting any point on $K_{\mathrm{o}}(t)$ that is below the virtual paper ψ onto the plane of ψ [Eq. (5.3b)]. To compute $K_{\mathrm{ro}}(t)$, we need to first detect the intersecting point $K_{\mathrm{o}}(t_{\mathrm{p}})$ between $K_{\mathrm{o}}(t)$ and ψ. In Eq. (5.3c), we define this point to be the closest point below the virtual paper ψ to $K_{\mathrm{o}}(1)$, if the point exists. Here $K_{\mathrm{o}}(1)$ is the bottom of the skeleton, i.e. the tip of the penholder. Then the rotated version K_{ro} can be computed by rotating $K_{\mathrm{o}}(\mathrm{t})|_{0 \leqslant \mathrm{t} < \mathrm{t_p}}$ around $K_{\mathrm{o}}(t_{\mathrm{p}})$ through a minimum angle $\phi_{\min}$ to make all the points on the skeleton come on top of the virtual paper ψ [Eqs. (5.3d) and (5.3e)]. At last, before replacing $K_{\mathrm{o}}(t)$ with the new skeleton $K_{\mathrm{n}}(t)$, we first smooth $K_{\mathrm{n}}(t)$ by applying a B-spline curve fitting to ensure a smooth transition of the curvature of $K_{\mathrm{n}}(t)$ [Eq. (5.3f)].

We also model the kinking up of a hair macro due to large friction induced by high stress which is according to classic Newton force. The larger the friction, the slower the hair macro will move. This non-uniform displacement will cause local prolongation and compression of the hair macro's skeleton, which when it becomes too severe will lead to kinking up of the skeleton, as shown in Fig. 5.8. To simulate this, we introduce additional curvature to the skeleton so that the arc length of the skeleton is the skeleton's original length and the chordal length is its squeezed length.

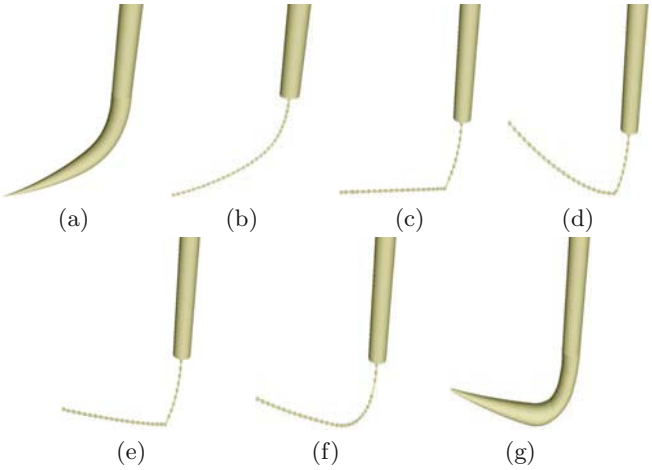


Fig. 5.7. Skeleton deformation of the virtual brush: (a) a hair macro penetrating the virtual paper; (b) its corresponding skeleton; (c) the projected skeleton, (d) the rotated skeleton; (e) the interpolated version of the skeleton; (f) the new skeleton after smoothing, which is completely above the virtual paper; (g) the deformed hair macro with the new skeleton

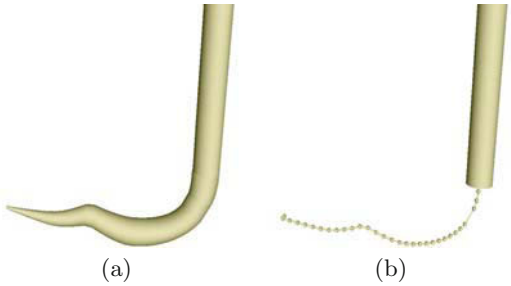


Fig. 5.8. (a) A kinked-up hair macro; (b) its corresponding skeleton

5.4.1.2 Deformation of sweeping ellipse

Recall that the geometry model of a hair macro is constructed by sweeping a variable ellipse along a trajectory. When the brush is deformed, the profiles of the variable ellipse touching the paper will also be deformed. We apply minimization to the areas of the parts of the deformed profiles that are under the virtual paper plane, with the hard constraint that the profiles' areas cannot change. These areas could change at a later stage in the modeling when we take into account the inner stress. We minimize the areas concerned for the simple reason that the physical brush cannot go under the canvas; minimizing gives us the best approximation to what would happen in reality. Eq. (5.4) is a brief mathematic representation of this deformation.

$$\begin{cases} \min \sum_i Area\Big(UnderPaper(Profile_i)\Big) \\ \text{subject to :} \\ \forall\, Profile_i :\ Area(Profile_i) \equiv \text{Constant} \end{cases} . \tag{5.4}$$

Here $Profile_i$ is the i-th profile of the sweeping ellipse of a certain hair macro. $UnderPaper(Profile_i)$ computes the part of the profile $Profile_i$ that is under the virtual paper plane. Operator $Area(\cdot)$ computes the area of a planar shape.

5.4.2 Deformation due to Inner Stress

Once the hair macro is deformed due to collision between the brush and the paper, inner stress will develop inside the hair macro. Because we model the geometry of a hair macro explicitly, its volume is computable. Based on the volumes of the initial and the deformed geometry of a hair macro, we can estimate the inner stress of the macro. This inner stress will then give rise to further deformation of the geometry of the macro, in the form of distension and splitting of the hair macro.

5.4.2.1 Estimating the inner stress

From Eq. (5.1) we notice any point in a certain profile $E(t)|_{t=t_0}$ of the variable ellipse $E(t)$ can be determined by the pair (ν, w) under a given $L(t_0)$, $S(t_0)$ and $\theta(t_0)$. Accordingly, we set up a local planar elliptical polar coordinate system for $E(t_0)$ by taking the major and minor axes of $E(t_0)$ as two axes of the coordinate system, and the center of $E(t_0)$ as its origin. We can then establish a point to point correspondence between the un-deformed profile of $E(t_0)$ and its deformed counterpart in the new local coordinate system. We use the subscripts "u" and "d" to refer to the un-deformed and deformed items respectively during the brush deformation. Then the correspondence rule we adopted can be stated as: two points $(\nu_\mathrm{u}, w_\mathrm{u})$ in $E_\mathrm{u}(t_0)$ and $(\nu_\mathrm{d}, w_\mathrm{d})$ in $E_\mathrm{d}(t_0)$ are correlated if and only if $\nu_\mathrm{u} = \nu_\mathrm{d}$ and $w_u = w_\mathrm{d} + \tau(t_0)$, where τ is a parameter to be determined later. By this correspondence, we can then estimate the local inner stress $\rho(\nu_\mathrm{d}, w_\mathrm{d})$ around point $(\nu_\mathrm{d}, w_\mathrm{d})$ on $E_\mathrm{d}(t_0)$ with Eq. (5.5).

$$\rho(\nu_\mathrm{d}, w_\mathrm{d}, \tau(t_0)) \triangleq \begin{cases} \tan\left(\max\left(k_1(1-\frac{\nu_\mathrm{u}}{\nu_\mathrm{d}}), -\frac{\pi}{2}\right)\right) & \text{when} \nu_\mathrm{d} \leqslant \nu_\mathrm{u}; \\ \tan\left(\min\left(k_2(1-\frac{\nu_\mathrm{u}}{\nu_\mathrm{d}}), \frac{\pi}{2}\right)\right) & \text{otherwise.} \end{cases} \tag{5.5}$$

Based on this point-wise stress estimation, we can derive the formulae to evaluate the average inner stress $\rho(E_\mathrm{d}(t))$ of the ellipse $E_\mathrm{d}(t)$ as Eq. (5.6). Here by our design, $\tau(t)$ is supposed to be a number between $[0, 2\pi)$ in order to minimize the absolute value of $\rho(E_\mathrm{d}(t))$.

$$\rho(E_\mathrm{d}(t)) \triangleq \min_{\tau(t)\in[0,2\pi)} \left(\frac{\int_{E_\mathrm{d}(t)} \rho(\nu_\mathrm{d}, w_\mathrm{d}, \tau(t)) \mathrm{d}\nu_\mathrm{d}\, \mathrm{d}w_\mathrm{d}}{\int_{E_\mathrm{d}(t)} \mathrm{d}\nu_\mathrm{d}\, \mathrm{d}w_\mathrm{d}}\right). \tag{5.6}$$

Integrating the point-wise stress throughout the volume of the hair macro can further lead to the estimated average inner stress $\rho(\mathbf{H}_\mathrm{d})$ of the deformed hair macro $\mathbf{H}_\mathrm{d}$ as given by Eq. (5.7).

$$\rho(\mathbf{H}_{\mathrm{d}}) \triangleq \int_0^1 \rho(E_{\mathrm{d}}(t))\mathrm{d}t. \tag{5.7}$$

A positive ρ suggests that the stress is inward, which will compress the geometry of the hair macro; whereas the negative ρ suggests that the stress is outward and will dilate the hair macro; zero ρ means the hair macro is in its steady geometry. If only $\rho \neq 0$, stress-driven brush deformation tends to reduce the current inner stress of the virtual brush through brush geometry deformation.

5.4.2.2 Distension of hair macro

Although the volume of a hair macro is determined mainly by the number of hair strands in the macro, there is another factor, the *phenomenal distensibility* of the hair macro, which will affect the volume. A high degree of phenomenal distensibility suggests that the hair strands are not very densely packed inside the hair macro. Obviously, distensibility is a function of the inner stress of a hair macro. For a hair macro, if its phenomenal distensibility increases (resp. reduces), all of the sweeping ellipse profiles of this hair macro will increase (resp. reduce) in size. For simplicity we model the degree of distensibility β as a function of the stress in the hair macro $\mathbf{H}$ as follows:

$$\beta(\mathbf{H}) \triangleq \frac{1}{2} - \frac{1}{\pi}\arctan(k_3 \times \rho(\mathbf{H})). \tag{5.8}$$

Since $\rho(\mathbf{H})$'s range is the whole real number axis $(-\infty, +\infty)$, the range of $\beta(\mathbf{H})$ is $(0, 1)$. Here k_3 is a positive coefficient controlling the sensitivity of the distensibility to the inner stress. For each hair macro $\mathbf{H} = \mathit{Modeling}\big(K(t), S(t), L(t), \theta(t)\ \big)$, if its phenomenal distensibility grows from $\beta_1(\mathbf{H})$ to $\beta_2(\mathbf{H})$, $\mathbf{H}$'s parametric representation will change to $\mathbf{H}'$, as defined in Eq. (5.9).

$$\begin{cases} \mathbf{H}' \triangleq \mathit{Modeling}\big(K(t), S'(t), L'(t), \theta(t)\big) \\ S'(t) \triangleq (\beta_2(\mathbf{H})/\beta_1(\mathbf{H}))S(t) \\ L'(t) \triangleq (\beta_2(\mathbf{H})/\beta_1(\mathbf{H}))L(t) \end{cases}. \tag{5.9}$$

5.4.2.3 Splitting of hair macro

If there is a part inside the volume of a hair macro, whose stress is above the threshold of a maximum tolerable inner stress, the hair macro will split. Splitting of the hair macro takes place in a stress-descending order if multiple parts of the virtual brush meet the splitting criterion simultaneously. Split hair macros could merge again if their inner stress starts to come down. In the current version of our modeling, merging however is controlled through user interaction, mimicking the way the user caresses a physical brush to merge some split hair. Fig. 5.9 shows some brush motion simulation results with and without brush splitting.

More formally, if there exists a point in a hair macro, whose stress is above the threshold of a maximum tolerable inner stress, the hair macro $\mathbf{H}$ will split.

According to the way we compute the stress in **H**, discussed in Sect. 5.4.2.1, the distribution of the stress within **H** is continuous. This means if there exists one such point in **H** whose stress is greater than the splitting threshold, it is likely that the neighbouring points are also above the threshold. Therefore our splitting procedure works on groups of points. For one specific sweeping ellipse we need first of all to separate those points, whose stress is above the threshold, from the other points. If the number of connected points whose stress is above the threshold is very small, there will be no splitting. This is the effect of physical attraction between neighbouring hair threads. After the small group elimination, real splitting operation comes into the picture. The separated point groups will become two ellipses, the centers of which are the centroids of each group. To determine the lengths of major and minor axes of the ellipse, we impose two constraints: the area of the ellipse should be equal to the area of the group before splitting, and the ratio of the major axis to minor axis of the ellipse is equal to the ratio of the length to width of the bounding box of the point group. The orientation of the major axis of the new ellipse is determined by the principal axis of the group. Splitting the hair macro takes place in a stress-descending order if multiple point groups meet the splitting criterion. After splitting **H** at the largest average stress point group, the stress inside **H** is recomputed. Splitting continues as long as the splitting condition holds.

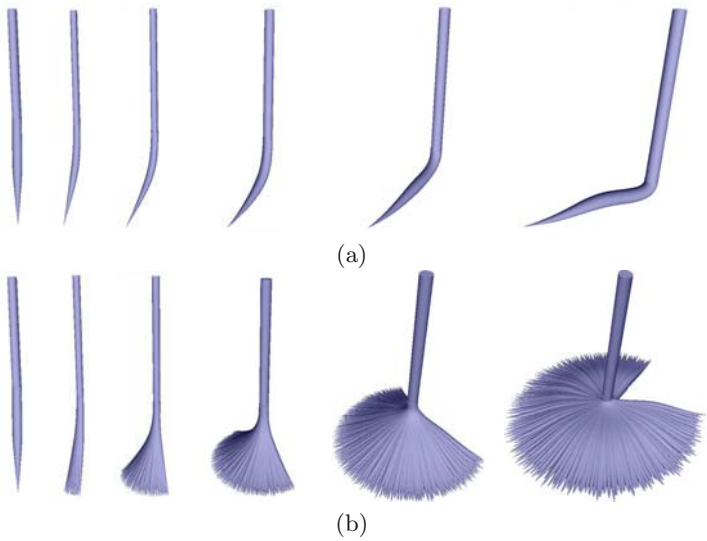


Fig. 5.9. A brush deforming when being pressed down continuously: (a) with no splitting of hair; (b) with splitting

5.4.2.4 Recovery from deformation

To simulate recovery from deformation, we model the material of the hair macro as a kind of hybrid elastic-rigid material. According to studies in material science [Cal99], the deformed hair macro will recover to a certain degree, which ranges from full recovery in the case that the material is purely flexible to no recovery at all if the material is completely rigid. In our computation model we determine the recovery extent based on the amount of inner stress of the hair macro and the hair macro's inherent threshold of elasticity and threshold of rigidity. Using the estimated recovery extent as the weight, we can then simulate the brush's recovery by interpolating its original undeformed geometry and its current deformed geometry. The above process is described in Eq. (5.10).

$$\begin{aligned} &\mathbf{H}_{\mathrm{r}} \leftarrow \big(\mathbf{H}_{\mathrm{d}} \times (1 - extent) + \mathbf{H}_{\mathrm{u}} \times extent\big), \\ &extent = \begin{cases} 1 & Stress(\mathbf{H}_{\mathrm{d}}) < thre^{\mathrm{ela}}; \\ 0 & Stress(\mathbf{H}_{\mathrm{d}}) > thre^{\mathrm{rig}}; \\ \frac{thre^{\mathrm{rig}} - Stress(\mathbf{H}_{\mathrm{d}})}{thre^{\mathrm{rig}} - thre^{\mathrm{ela}}} & \text{otherwise.} \end{cases} \end{aligned} \tag{5.10}$$

Here $\mathbf{H}_{\mathrm{r}}$ is the geometry that a hair macro will recover to; $\mathbf{H}_{\mathrm{d}}$ is its deformed geometry before recovery; $\mathbf{H}_{\mathrm{u}}$ is the hair macro's initial (un-deformed) geometry; *extent* is the recovery extent, which is determined by the current inner stress, $Stress(\mathbf{H}_{\mathrm{d}})$, of $\mathbf{H}_{\mathrm{d}}$ as well as the hair macro's inherent thresholds of elasticity, $thre^{\mathrm{ela}}$, and rigidity, $thre^{\mathrm{rig}}$.

5.4.3 Calibrating the On-line Results

To ensure the accuracy of our simulation of the brush dynamics with respect to a physical brush, we calibrate the fast on-line simulation results in the second phase of the modeling. The procedure relies on data from a "simulation error database", whose records came from sampling using a real brush. Since the deformation of our virtual brush's geometry is according to the six degrees of freedom (DOFs) of the input, the database is indexed by the differential of the virtual brush's six DOFs between two consecutive simulation time slices and the current brush geometry. The content of each record in the database is the corresponding transformation of the virtual brush's current geometry. Our brush deformation database operates at the level of a hair macro in our three-level hierarchy. This reduces substantially the number of different cases that need to be separately sampled and stored in the database. To further reduce the size of the calibration database, we assume the effects that all the six DOFs have on the brush geometry are independent. As a result, the whole calibration database is divided into many small sections, each of which being responsible for calibrating the part of the deformation caused by variation of only one specific DOF. It turned out that a modest number of records (40) is enough to perform a satisfactory calibration for improving the on-line simulation results in our system.

Compared with traditional physically-based approaches to model the brush dynamics through numerical computation, there are two important

advantages of our on-line plus off-line modeling approach. First, we do not have to model those extremely complicated physical processes of a brush's dynamics. Second, we can avoid the very time-consuming and probably unstable numerical computation for solving differential equations on the fly if some truly powerful but complex equations can be established to model all the underlying detailed dynamics governing the brush's behavior.

More formally, since the deforming of a hair macro's geometry is according to the six DOFs of the input, the database is indexed by: 1) the differential of the virtual brush system's six DOFs between the current and the previous simulated time slice, which is denoted as $\partial\mathbf{D}$; 2) the current geometry of the hair macro $\mathbf{H} \triangleq \mathit{Modeling}\big(K(t),\, S(t),\, L(t),\, \theta(t)\ \big)$ [Eq. (5.2)]. The content of each record in the database is the corresponding transformation $\mathbf{T}$ on $\mathbf{H}$'s current geometry, namely $K(t), S(t), L(t), \theta(t)$. When collecting records to establish the calibration database, we choose the samples in such a way that no two similar brush dynamic deformation processes are sampled and stored.

The above keywords to index the database have many dimensions. We construct a manageable database by taking advantage of the inter-relationships among the fields of the keywords. The product of $L(t_0)$ and $S(t_0)$ reflects the area of the ellipse $E(t_0)$, and hence the number of hair strands in the hair macro under a certain distensibility. If the hair macro will not split, $S(t_0)$ can be computed from $L(t_0)$. Thus we can use four control points to represent the geometry deformation on $L(t)$ and four control points for $K(t)$. We also assume there is no inter-relationship between the dynamic deformations on $K(t)$ and $L(t)$, i.e. changing either one of the two splines will not have any direct effect on the other spline. This assumption comes from the observation that there is only a weak relationship between deformations on $L(t)$ and $K(t)$ during a real brush's deformation. As a result, the whole database can be divided into two sub-databases: **DL** for calibrating the simulated dynamic deformation on $L(t)$, and **DK** for calibrating the simulated deformation on $K(t)$. For simplicity we ignore the possible but minute deformation on the brush geometry introduced by the x, y displacement of the virtual brush. So only four of the six DOFs will affect the brush's geometry deformation process. We also assume that the effects these four DOFs have on the brush geometry are separable. Thus **DL** and **DK** are further divided into four sections, each of which is used to calibrate the error in the simulation of the geometry's dynamic deformation under the change of one DOF. For each of the sections in **DL** and **DK**, the index words are of low dimensions: one dimension for the varied DOF and the remaining dimensions for the control points of $L(t)$ in **DL** or $K(t)$ in **DK**.

During database retrieval we find several records in the database whose distances to the input index are within a certain threshold. The contents of the retrieved records are interpolated using the distances between the indices of the retrieved records and the input index as the weights. For database retrieval, the distance between two splines is defined to be the sum of the Euclidian distances of their corresponding four control points.

With the way we organize the database as described above, the database in our prototype implementation contains a modest number of records (40).

This number of records is already sufficient for performing a satisfactory calibration to correct/improve the online simulation results.

5.5 E-painting System based on Realistic Virtual Brush Modeling

We have built a complete working e-brush system based on the modeling strategies just described. Compared with other virtual brushes, this new system is designed to present a realistic brush in the sense that the system accurately and stably simulates the complex painting functionality of a running brush, and therefore is capable of creating high-quality digital paintings with minute aesthetic details that can rival the real artwork.

5.5.1 Additional Components of Our New Painting System

Other than brush modeling, the system has also incorporated a novel pigment model and a user manipulability improving component. As these two components are not the focus of this chapter, we will only give an overview in the following.

5.5.1.1 A novel pigment model

During painting, the contour of the current ink mark left by virtual brush on virtual paper can be computed by intersecting the geometrical models of these two virtual objects. To produce a good texture for the ink mark, a pigment model is required. A number of pigment models for digital painting have been proposed [Sma91, Lee99, ZST+99, Coc91], but they are either not well embedded into a 3D virtual paintbrush, or were developed earlier and therefore could only produce some very coarse painting results. A recent pigment model is contributed by [CAS+97]. It is however too slow to be used in interactive painting because of the need to solve heavy differential equations on the fly. Other pigment models that can be found in existent virtual brush systems are too simplistic: they either simply transfer ink values from the penetrating brush tip onto the ink mark area [CT02], or apply alpha blending in the above ink transformation to simulate glazing effects [BSLM01].

For our system, we devise a new pigment model that is best for expressive Oriental painting. The prominent feature of this new pigment model is that it is completely and seamlessly integrated into our realistic virtual brush model: we store both the local ink color and the wetness at each control point of the geometry model of our virtual brush. When generating the ink mark, each point in the ink mark is painted using a color which is a linearly interpolated result between the ink mark's original color on the paper and the color of the point on the brush surface contacting this ink mark. The interpolation weight is a random number, whose distribution is controlled by the current inner stress of the hair macro, the local wetness around the painted

pixel, as well as the quality parameters (explained in the next section) of the virtual brush. This probability-based pigment model allows us to simulate the dry brush effect, the running style effect and the ink saturation effect. These are important aesthetic effects that can contribute significantly to the expressiveness of the painting system.

5.5.1.2 User manipulability improving component

As with any real brush, it can happen that the user would feel unsatisfied with their creations using the virtual brush. Instead of training the user, our virtual brush system has the unique feature of training the brush. This is done through an intelligent *user manipulability improving component*. This component applies an additional transformation to the user's input before the system commits to the final painting result.

The idea behind the design of this component is as follows. Our virtual brush carries with it a collection of beautiful strokes and the input for creating these strokes. The user can choose among these "known" brush strokes not for his/her own strokes, but use them as training samples. For each selected sample, the user would then use our virtual brush to produce a stroke as close to the sample as possible. The system then applies a numerical analysis to compute a transformation from the user's input to that of the sample stroke. The derived transformation is the "personal habitual bias" of the user. Later, when the user paints using our virtual brush, his/her input will firstly go through the transformation of his personal habitual bias. Thus, it is the transformed input rather than his original input that drives the virtual brush to paint.

The system input used to paint one stroke by our virtual brush is essentially a six-dimensional curve, where each dimension is the profile of one DOF for the geometry of the virtual brush with respect to a certain time during painting. This is denoted as $D(t) = \{D_1(t), D_2(t), D_3(t), D_4(t), D_5(t), D_6(t)\}$. We also denote the standard input to create the brush stroke as $I(t) = \{I_1(t), I_2(t), I_3(t), I_4(t), I_5(t), I_6(t)\}$. We then derive a third six-dimensional curve to capture the user painting bias, i.e. $B(t) = \{B_1(t), B_2(t), \cdots, B_6(t)\}$, which comes from dividing the value of each point in $I(t)$ against the corresponding point in $D(t)$, namely $B_i(t_j) \triangleq I_i(t_j)/D_i(t_j)\,(i = 1, 2, \cdots, 6)$ and t_j is one simulation time epoch. For $B_i(t)$'s we apply piecewise degree-3 Bézier curve fitting. Suppose $B_i(t)$ is fitted using n segments of a Bézier curve, $F_i(t) = \{F_i^1(t), F_i^2(t), \cdots, F_i^n(t)\}$. We also use the segmentation result to segment the input profile of $D_i(t)$ into $D_i(t) = \{D_i^1(t), D_i^2(t), \cdots, D_i^n(t)\}$. A set of mapping relationships M_i can thus be collected, where $M_i \triangleq \{(D_i^j(t), F_i^j(t))|j = 1, 2, \cdots, n\}$, which is the extracted i-th component of the user's personal habitual bias.

Applying this extracted painting bias is simple. With a new user input $D_i^{\text{new}}(t)$, we search the collected mapping relationships M_i to find the closest matching record, say $D_i^{\text{close}}(t)$ with the complete mapping relationship pair being $(D_i^{\text{close}}(t), F_i^{\text{close}}(t))$. Then, $D_i^{\text{new}}(t)$ is multiplied by $F_i^{\text{close}}(t)$ to achieve the user painting bias correction. To better strengthen the functionality of

this user manipulability improving component, we retrieve several closest matching pairs and then use linear interpolation to derive the specific painting correction pair by taking the discrete curve similarities of the first fields of the retrieved pairs to $D_i^{\text{new}}(t)$ as the interpolation weights. The definition of curve similarity is taken from [HOCS02].

5.5.2 The Running System

Figs. 5.10 (a to b) show two screen shots of the GUI of the running system. (a) was taken when the user was customizing the geometry of the hair macro—the skeleton as well as the profiles of the sweeping ellipse. (b) shows a painting in the making, with some closeup views as (c to d).

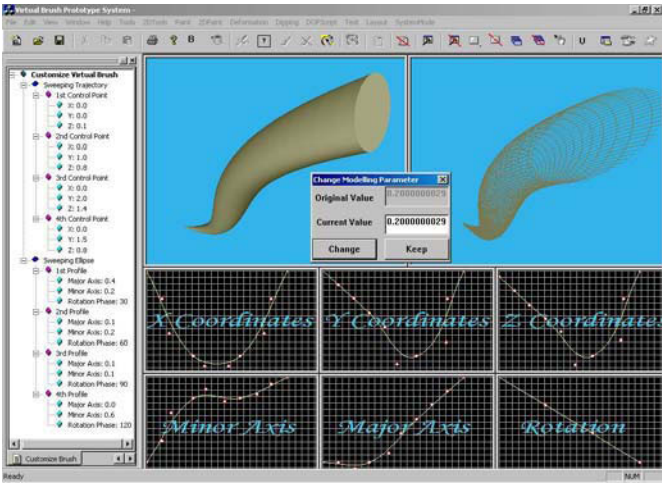
Figs. 5.11 and 5.12 give two examples of e-paintings created using our improved virtual brush system design, which are imitations of real paintings. Although the simulation attends to many fine details, our virtual brush running on a PC with 256 M memory and an AMD Duron 1.2 GHz processor can respond interactively to user commands. Currently, we use a WACOM pen on a tablet to get the position, the pressure (used as the brush's vertical displacement), and the tilt of the virtual brush; and keyboard input to get the remaining two DOFs. The rational behind using the vertical displacement of the brush as the pressure term is because the more displacement a brush experiences, the more inner pressure it will develop. A better input device in the future should provide some degree of haptic feedback.

5.6 Related Work

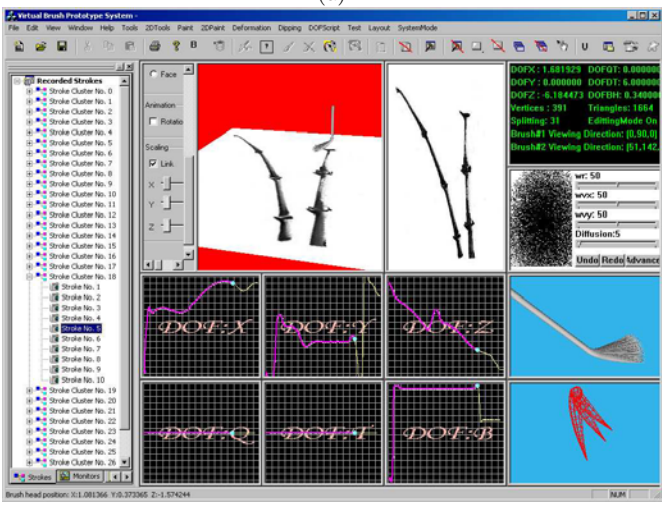
5.6.1 Wong & Ip's System

Wong and Ip's system [WI00] simulates the painting functionality of a real brush with granularity of a single hair thread. As a typical paintbrush can have hundreds or thousands of hair strands, the responses of their system tend to be too slow for interactive painting. Moreover, their system relies on an intricate set of parameters to control the shape, density and opacity of the brush's ink mark, all of which need to be manually specified through user interaction. Thus it is at best a semi-interactive system, and not natural to use for the human artists. And the ink mark generated by their system is always an elliptical blot.

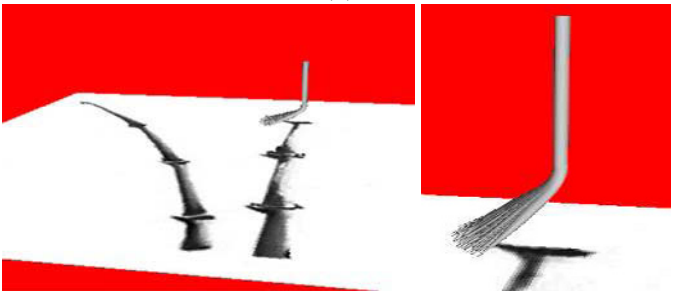
In comparison, we use writing primitives instead of individual hair strands to serve as the basic working units in the virtual hairy brush. In general, a single primitive is probably enough for regular scripts, official scripts and most running scripts and perhaps a dozen or so for cursive scripts. Hence our software needs to handle only a small number of writing primitives most of the time. As a result, our system operates comfortably in real-time mode because of the hierarchical modeling of the paintbrush geometry and hardware acceleration we employ. Our system is natural to use and is a genuine



(a)



(b)



(c) (d)

Fig. 5.10. GUI of the running system: (a) the running system—user customizing the virtual brush; (b) the running system—bamboos being painted; (c & d) closeup views of (b)

Fig. 5.11. "Spring garden" by the virtual brush

Fig. 5.12. "Summer water lily" by the virtual brush

interactive system in which the determination of both the painted stroke contour and its texture is fully automatic. The ink mark rendered by our virtual brush can be varied and of irregular shape, in order not to limit the creativity of the artist.

5.6.2 The DAB System

The brush geometry model in the DAB system [BSLM01] can represent some of the most common types of brushes typically used in oil painting, but not all the possible types. Their system is weak in modeling the splitting of a paintbrush. They use two separate models to model the brush geometry and the brush dynamics: a particle system skeleton to simulate the basic brush motion, and a deformable mesh to model the actual shape of the brush head surface. Synchronizing the running of the two models is a non-trivial problem. Their brush dynamics modeling is based on a pair of differential equations, and to solve these equations involves some tradeoff between system stability, computation efficiency and simulation accuracy.

In our system, the brush geometric model we use is very powerful, which can capture literally almost all the possible brush geometries with little mem-

ory consumption, and the rendering of the model is fast. Our brush's geometry model dynamics model and the pigment model are designed to be closely knitted together, and hence there is no synchronization problem between these models. Because of the division of work into on-line and off-line computations in modeling the brush's dynamics, our system performs the simulation accurately, operates stably, and requires a modest amount of system resources.

5.6.3 Chu & Tai's System

Chu and Tai [CT02] use an explicit geometrical modeling approach for the un-split brush tip, which is mathematically equivalent to the clustered modeling strategy as reported in one of our previous publications [XTLP02]. To model the split brush, unlike our fully geometrically-based approach, they use an alpha map. This results in an over-simplification which limits the expressiveness and the amount of fine details that the brush is able to produce. For the brush dynamics, their modeling through brush energy minimization can simulate small-scale deformation of the brush geometry but not large-scale bending or stretching due to the restriction of constrained energy minimization. In fact, large-scale deformation of brush geometry is a frequent phenomenon in watercolor painting and Oriental painting.

Chu and Tai's system gives a somewhat awkward support for mimicking the dry brush effect and simulating brush splitting. The alpha map is central to their production of 2D expressive painting results. It is used to control which parts of the brush tuft are dry, and to specify which parts of the brush tip are split. Computing the alpha map automatically is challenging which Chu and Tai did not attack. Loading a pre-stored alpha map from a texture database will miss the power of user control on the texture of painted strokes. The idea however was published more than ten years ago [HLW93, HL94] and perfectly implemented in a commercial paint system [Cre]. The other possible choice would be to rely on the user to control the alpha map for the desired texture of the strokes to be painted. In any case, Chu and Tai's system lacks the needed support for the very important dry brush and brush splitting effects that occur frequently in Oriental painting.

In comparison, our system offers an explicit modeling of the highly complex paintbrush geometry to support creation of even the minutest details in a generated e-painting. Our hybrid observational-model-calibration-database modeling of the brush dynamics provides an accurate simulation of the paintbrush in motion, which includes splitting, stretching, bending to any extent. A variety of paintbrush effects are supported owing to the realistic modeling of the virtual brush as well as the seamlessly embedded pigment model. As a result no mandatory user interaction is required for generating aesthetic textures for the painted strokes.

5.7 Conclusion and Future Work

We have presented the design of a powerful painting system based on realistic modeling of the paintbrush. The amount of details being modeled necessitates the many time or space optimizations that we have introduced into the design. The result is a high degree of realism in every simulation step. Here is a summary of the unique features of our modeling approach: 1) Clustered and hierarchical modeling is used to minimize redundant representations and computations, and together with hardware acceleration, the model is easily renderable in real-time; 2) Division of the modeling into on-line tasks and off-line calibration makes possible an accurate and stable simulation of the brush's motion using little computational resources; 3) Our virtual brush can automatically determine both its geometric contour and the texture of its ink mark on the virtual paper without any human intervention; 4) Other features such as the pigment model and the user manipulability adaptation component make the virtual brush system a powerful one and natural to use for creating high-quality e-artwork.

There are still many interesting problems to be addressed on further enhancing the components in this virtual brush-based painting system design. One of them is that of choosing the appropriate samples for the brush motion calibration database that can enumerate all the possible motions a painting brush could experience without repetitive sampling. For the currently employed pigment model, only water-soluble pigment is simulated. An obvious future task would be to extend the model to cover oil painting and maybe other kinds of painting by following Small's [Sma91] and Cockshott's [Coc91] pioneering approaches. For the user manipulability improving component, finding the features that distinguish good input leading to visually pleasing brush strokes from bad input could help establish a mechanism to perform auto-beautification.

Although we have gone after a detailed modeling of the paintbrush, a real brush operates in a fashion that is orders of magnitude more complex than what is currently simulated. Features that can be added in the future versions include repeated dipping effects and more user's control of the brush during painting. Other features requiring longer-term effort include support for 3D painting (such as oil painting), and vectorization of painting results. The latter could lead to many interesting applications such as animation of e-paintings.

References

[ABL95] Maneesh Agrawala, Andrew C. Beers, and Marc Levoy. 3D painting on scanned surfaces. In *SI3D '95: Proceedings of the 1995 Symposium on Interactive 3D Graphics*, Monterey, CA, USA: ACM Press, pages 145–150 & 215, 1995.

[BSLM01] Bill Baxter, Vincent Scheib, Ming C. Lin, and Dinesh Manocha. DAB: Interactive haptic painting with 3D virtual brushes. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA: ACM Press, pages 461–468, 2001.

[Cal99] Jr William D. Callister. *Materials Science and Engineering: An Introduction*. John Wiley & Sons, Inc., 1999.

[CAS+97] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-generated watercolor. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA: ACM Press/Addison-Wesley Publishing Co., pages 421–430, 1997.

[Coc91] Malcolm Tundle Cockshott. *Wet and sticky: a novel model for computer-based painting*. PhD thesis, Computing Science Department (Research Report 91/R20), University of Glasgow, 1991.

[Cre] Creature House Ltd. Expression (software). First released in October 1999. http://www.creaturehouse.com/, Hong Kong.

[CT02] Nelson S.H. Chu and Chiew-Lan Tai. An efficient brush model for physically-based 3D painting. In *Proceedings of Pacific Graphics (PG '02)*, Beijing, China: IEEE Computer Society, pages 413–421, 2002.

[HH90] Pat Hanrahan and Paul Haeberli. Direct wysiwyg painting and texturing on 3D shapes. In *SIGGRAPH '90: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, Dallas, TX, USA: ACM Press, pages 215–223, 1990.

[HL94] Siu-Chi Hsu and Irene H.H. Lee. Drawing and animation using skeletal strokes. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, Orlando, FL, USA: ACM Press, pages 109–118, 1994.

[HLW93] Siu-Chi Hsu, Irene H.H. Lee, and Neil E. Wiseman. Skeletal strokes. In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, Atlanta, GA, USA: ACM Press, pages 197–206, 1993.

[HOCS02] Aaron Hertzmann, Nuria Oliver, Brian Curless, and Steven M. Seitz. Curve analogies. In *EGRW '02: Proceedings of the 13th Eurographics Workshop on Rendering*, Aire-la-Ville, Switzerland: Eurographics Association, pages 233–246, 2002.

[HSS02] Nick Halper, Stefan Schlechtweg, and Thomas Strothotte. Creating non-photorealistic images the designer's way. In *NPAR '02: Proceedings of the 2nd International Symposium on Non-*

photorealistic Animation and Rendering, Annecy, France: ACM Press, pages 97–104, 2002.

[Inc] Adobe Systems Incorporated. Adobe photoshop (software).

[KMM+02] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. WYSIWYG NPR: drawing strokes directly on 3D models. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, San Antonio, TX, USA: ACM Press, pages 755–762, 2002.

[Lee99] Jintae Lee. Simulating Oriental black-ink painting. *IEEE Computer Graphics and Applications*, 19(3):74–81, 1999.

[Pix00] Pixologic. Z-brush, http://pixologic.com, 2000.

[SABS94] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. Interactive pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, Orlando, FL, USA: ACM Press, pages 101–108, 1994.

[Sma91] David Small. Simulating watercolor by modeling diffusion, pigment, and paper fibers. In *Proc. of SPIE '91*, San Diego, CA, USA: SPIE Press, 1991.

[Smi01] Alvy R. Smith. Digital paint systems: an anecdotal and historical overview. *IEEE Annals of the History of Computing*, 23(2):4–30, 2001.

[Str86] Steve Strassmann. Hairy brushes. In *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, Dallas, TX, USA: ACM Press, pages 225–232, 1986.

[WI00] Helena T.F. Wong and Horace H.S. Ip. Virtual brush: a model-based synthesis of Chinese calligraphy. *Computers and Graphics*, 24(1):99–113, 2000.

[XTLP02] Songhua Xu, Min Tang, Francis C.M. Lau, and Yunhe Pan. A solid model-based virtual hairy brush. *Computer Graphics Forum (Proceedings of Eurographics '02)*, 21(3):299–308 & 625, 2002.

[ZST+99] Qing Zhang, Youetsu Sato, Junya Takahashi, Kazunobu Muraoka, and Norishige Chiba. Simple cellular automaton-based simulation of ink behavior and its application to Suibokuga-like 3D rendering of trees. *Journal of Visualization and Computer Animation*, 10(1):27–37, 1999.