iR2s: Interactive Real photo to Sumi-e

Ning Xie*
Computer Science Dept.
Tokyo Inst.of Technology

Hamid Laga†
Global Edge Institute
Tokyo Inst. of Technology

Sugru Saito‡
Computer Science Dept.
Tokyo Inst. of Technology

Masayuki Nakajima§
Computer Science Dept.
Tokyo Inst. of Technology

Figure 1: *Shapes interactively rendered with Oriental brush strokes. The user draws contours on an empty canvas or on real images and the system converts them into Sumi-e paintings. The brush trajectory is estimated by minimizing an energy function and the painting styles are generated by direct texture mapping.*

Abstract

We propose an interactive sketch-based system for rendering oriental brush strokes on complex shapes. We introduce a contour-driven approach; the user inputs contours to represent complex shapes, the system estimates automatically the optimal trajectory of the brush, and then renders them into oriental ink paintings. Unlike previous work where the brush trajectory is explicitly provided as input, we automatically estimate this trajectory from the outline of the shapes to paint using a three-stages algorithm; first complex shapes are decomposed into elementary shapes that can be rendered with a single brush stroke. Second, we formulate the optimal brush trajectory estimation as the minimization of an energy function that measures the quality of the trajectory constrained by the variation along a stroke of the painting process parameters, such as the footprint position, size, orientation, and angular velocity. Finally, the estimated trajectories are rendered into brush strokes by mapping footprint textures scanned from real images. We combine the proposed framework with an interactive segmentation in order to convert real images into Oriental ink paintings. Experiments on complex shapes show that the proposed contour-based approach produces a large variety of strokes compared to trajectory-based approaches. It is particularly suitable for converting real images into Oriental ink paintings with minimum interaction.

Keywords: Sketch-based interface, shape decomposition, water dispersion

1 Introduction

In Oriental ink painting, called *Sumi-e*, complex scene information are abstracted into few relatively independent strokes. Unlike Western styles, such as water color and oil painting, Sumi-e painting uses few strokes to represent an object. Each stroke conveys through its shape and texture significant information about the scene. Its appearance depends on the shape of the object to paint, the brush trajectory, and the distribution of the pigment in the brush. Previous work on interactive simulation of Oriental brush strokes require from the user to specify the trajectory of the brush which is then rendered into Sumi-e. The stroke shapes that can be generated with this simple interaction are very limited. On the other hand, adding more complex interactions, such as brush pressure [Strassmann 1986; Way and Shih 2001], requires a certain level of painting expertise from the user and is not suitable for automatically converting real drawings into Oriental painting strokes.

We propose in this paper an interactive sketch-based system for rendering Oriental brush strokes on complex shapes. We introduce a contour-driven approach where the user inputs contours to represent complex shapes, the system estimates automatically the optimal trajectory of the brush, and then renders them into Oriental ink painting. Unlike previous work where the brush trajectory is explicitly specified as input, we automatically estimate this trajectory from the outline of the shape to paint. This approach is inspired from the popular tracing exercises used for teaching Chinese character writing. In this exercise, people who are not particularly talented in art are given shape outlines and paint strokes inside them creating beautiful paintings with minimum skills.

The main advantage of our approach, compared to trajectory based techniques, is that it gives the user better control over the shape of the final stroke without using specific input devices equipped with pressure sensors; the interactions in our system are limited to those required when sketching a drawing on a paper, i.e., contours of the objects to paint. It requires no painting expertise compared to digital painting systems, such as Adobe brush packages, ArtRage, and Corel Painter, and offers better flexibility and control compared to artistic filters implemented in commercial software. Figure 1 shows some results generated by our algorithm.

The remaining parts of the paper are organized as follows. After reviewing the related work (Section 1.1), we give an overview of our algorithm and outline the major contributions (Section 1.2). Section 2 describes our framewok for optimal trajectory estimation

*e-mail:sya_nei@img.cs.titech.ac.jp
†e-mail:hamid@img.cs.titech.ac.jp
‡e-mail:suguru@img.cs.titech.ac.jp
§e-mail:nakajima@img.cs.titech.ac.jp

NPAR 2010, Annecy, France, June 7 – 10, 2010.
© 2010 ACM 978-1-4503-0124-4/10/0006 $10.00

and stroke generation on simple shapes. We extend the approach to arbitrary complex shapes by the mean of shape analysis and decomposition (Section 3). Finally, estimated footprint sequences are converted into Oriental ink painting (Section 4). We show in this section how the proposed framework can be applied to interactively convert real images into Sumi-e renderings. Experimental results and evaluation are given in Section 5. We conclude in Section 6.

1.1 Related work

Existing methods for simulating brush strokes can be classified into: (1) methods that explicitly model the physical properties of the brush, its interaction with a paper, and the ink diffusion process, and (2) Stroke-Based Rendering (SBR) methods that simulate the rendering effects directly on a 2D canvas without modeling explicitly the physical phenomena. These methods can be further classified into automatic techniques which convert automatically real images into paintings without user interaction, and interactive techniques which are usually sketch-based. The approach we propose in this paper is sketch based.

3D virtual brush. Using a virtual 3D brush is motivated by the need to reproduce the real painting process and give to users an intuitive and natural feeling when holding a pen-like device. In digital painting software, such as Adobe brush packages, ArtRage, and Corel Painter, users draw with a mouse or a digital pen in the same way as when they draw with a real brush. These tools provide for expert users a professional environment with a canvas, brushes, mixing palettes, and a multitude of color options. However, physical modeling of the brush dynamics is very challenging. Several models have been proposed to capture the shape and dynamics of the tufts, the flow of the liquid from the tuft, and its absorption from the paper [Saito and Nakajima 1999], models that are able to mimic the brush flattening and bristle spreading [Chu and Tai 2002], and models equipped with a user-adaptation component that enables the system to learn the personal painting habits of different users [Xu et al. 2003]. Recent works focus on developing new painting tools, such as Air Brush [Konieczny and Meyer 2009] and interactive canvas [Schwarz et al. 2007], to overcome the limitations of the traditional input devices. Many other papers have proposed virtual brush models [Chu and Tai 2004; Xu et al. 2004; Baxter et al. 2005; Baxter et al. 2001; Baxter et al. 2004]. However, controlling automatically a virtual brush with six degrees of freedom in addition to the dynamics of the tuft is complex and existing models are in fact simplifications of the real physical process. On the other hand, non-expert users often require simplified environments where paintings can be generated with minimum interaction and painting expertise.

Stroke-Based Rendering (SBR). In many situations it is desirable to convert automatically real images into ink paintings, especially when the user has no painting expertise and is interested only in the painting results rather than in the painting process. Stroke-Based Rendering (SBR) approaches [Hertzmann 2003] operate in two steps; (1) first strokes are placed according to some goals, and then (2) rendered according to the desired painting style such as oil or impressionism painting. Stroke placement is very challenging. Early works, such as [Haeberli 1990], where the position, shape, size, and orientation of each stroke are specified manually, require considerable effort. Later works emphasize algorithms with higher degrees of automation with a possibility to generate a variety of stroke shapes. They are mainly guided by the low-level features of the image such as the gradient [Hertzmann 1998; Kovacs and Sziranyi 2004a] computed eventually at multiple scales, geometric moments and texture orientations [Shiraishi and Yamaguchi 2000; Li and Huang 2002], weighted edges and ridges [Kovács and Szirányi 2004b], or the medial axis of regions obtained after segmentation [Gooch et al. 2002]. Other developments concern the use of global saliency maps [Collomosse et al. 2006] or human perception models [DeCarlo and Santella 2002] to determine the position and the order in which strokes are placed. They achieved interesting artistic effects for painting styles such as impressionism and impasto where objects are represented with a large number of strokes. Their bottom-up nature however makes them not suitable for Sumi-e painting where an object should be abstracted into few strokes each one conveying maximum information about the object.

Sketch-based techniques. Alternatively, the stroke placement can be done interactively where the user inputs either the stream lines or the trajectories of the strokes. Sketch-based approaches are attractive since they give more flexibility and control to the user over the type of shapes that can be generated while not requiring advanced painting expertise. We refer the reader to [Olsen et al. 2009; Cook and Agah 2009] for recent surveys. In the following we focus on those relevant to our work.

A popular approach is to let the user sketch brush trajectories using a mouse or a pen-like device and then automatically convert them into paintings. Strassmann [1986] defines a stroke as a collection of footprint positions specified by the user. The region between two brush footprint positions is interpolated with a cubic spline. The brush is modeled as a collection of bristles which evolve over the course of a stroke, leaving a realistic image. The main difficulty is how to specify and vary the width of the strokes along a trajectory. A common approach is to define stroke width at each location as a function of the pressure at that location which requires specific input device [Strassmann 1986]. Alternatively, stroke width can be specified by inputting either the contours of the strokes [Way and Shih 2001], or control shapes, such as ellipses, at specific locations. The size of the control shape defines the width of the stroke at that location. The final stroke is then generated by B-spline interpolation [Seah et al. 2005; Su et al. 2002]. In the former, Way and Shih [2001] proposed an approach for synthesizing rock textures in Sumie painting where the user is required to specify the contours of the rocks to paint, texture stroke areas, and stroke parameters. The strokes are generated from trajectories which represent contours of the rocks to paint. It can be used in an interactive system as well as in an automatic setup if the contours of the rocks are efficiently detected.

In the latter approaches, Su et al. [2002] define a 2D region of a brush stroke with interval B-Splines where the user inputs control shapes which are then interpolated. In the simplest case, these control shapes are ellipses; ellipses of different radius will generate different types of shapes. A similar approach has been also proposed in [Hsu and Lee 1994; Seah et al. 2005]. These approaches are somehow similar to specifying the media axis and the stroke width which makes them very convenient for animating the strokes. While they offer a certain flexibility and control over the shape of the stroke, they are not intuitive since the real painting process does not involve inputting control shapes. Okabe et al. [2005] models the brush footprint characteristics using Hidden Markov Models (HMM) trained with data obtained from a real brush. This reduces the number of free parameters that need to be set by the user. Users however still have no control on the shapes of the strokes.

1.2 Overview and contributions

The distinct feature of our approach is that it can handle a larger variety of Oriental ink painting styles, including up-right and oblique brush strokes, which cannot be generated by previous methods. Instead of using the medial axis as the brush trajectory, we automatically estimate the optimal trajectory, given the shape of the object to paint, by minimizing an energy function parameterized by the physical parameters of the painting process. This offers more flex-

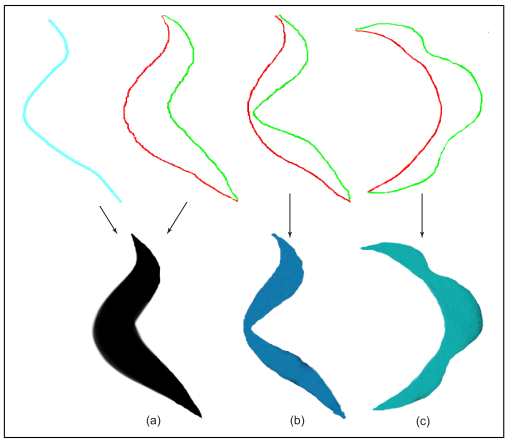


Figure 2: *Stroke rendering with brush trajectory vs. shape boundaries. (a) A stroke that can be generated by both approaches, (b) and (c) Typical failure examples of trajectory-based approaches.*

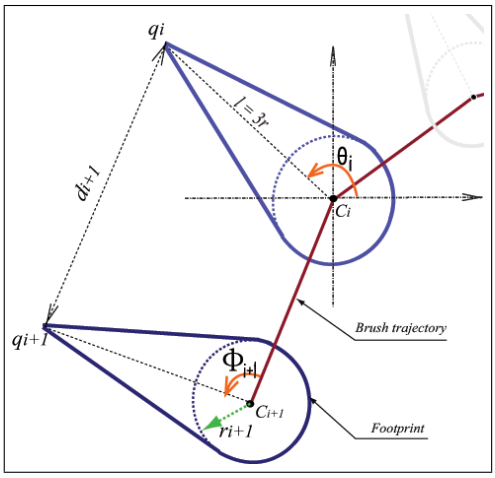


Figure 3: *The brush model, footprint trajectory, and the parameters of the energy function that quantifies the quality of a trajectory.*

ibility and allows to generate a larger variety of stroke shapes. To synthesize ink paintings from the estimated trajectories, we map real brush footprint textures with different ink quantities onto the estimated positions and interpolate in between to render smooth strokes. The artistic effects are also enhanced using an ink diffusion process constrained by the boundaries of the shape. The approach operates as follows;

- **Optimal brush trajectory estimation.** Frst complex shapes are automatically decomposed into elementary shapes that can be rendered with a single brush stroke. The decomposition is done by analyzing the distribution of branches of the medial axis of the shape. Second, by constraining the parameters related to the painting process, such as the variation of the footprint position, size, orientation, and angular velocity of the brush along the trajectory, we formulate the problem of optimal brush trajectory estimation as the minimization of an energy function that depends on these physical parameters.

- **Oriental ink painting style.** In the second stage, the estimated trajectory is rendered into a brush stroke by mapping footprint textures scanned from real pictures. We propose a procedure for synthesizing six basic stroke styles: dry ink, full-ink, hollow, first-half hollow, middle hollow, and both-ends strokes. We also simulate the pigment and water dispersion effects using Lattice Boltzmann Method (LBM) to generate smooth and realistic strokes.

- **Interactive image painting.** We combine the proposed framework with interactive object segmentation in order to convert real images into Oriental ink paintings.

The novelty introduced in this paper lies in the fact that we estimate the trajectory of the brush given contours of the shape the user would like to convert into Sumi-e painting. Previous works require the trajectory of the brush to be specified by the user. Our algorithm automatically estimates this trajectory. We are also able to handle simple shapes that can be rendered with a single stroke as well as complex shapes by the mean of shape decomposition. Our stroke placement algorithm combines high-level decomposition and the minimization of an energy function based on the parameters of the painting process. This results in different painting styles that cannot be handled in trajectory based approaches.

2 Processing elementary shapes

Sumi-e painting, unlike western styles, can be seen as the process of abstracting a complex scene into few strokes, each one conveys as much as possible information about the scene. Sumi-e favors long strokes. When an artist paints a stroke, the brush evolves along a smooth trajectory. In the basic Sumi-e style, called *Chokuhitsu*, the brush tuft follows the direction of the stroke, the base of the footprint touches as much as possible both sides of the shape to paint, and the trajectory of the tuft tip evolves at a very close distance to the medial axis of the shape. In the *Sokuhitsu* style, the disk of the brush footprint remains tangent to one boundary of the shape while the tip touches the other, and the brush footprint maintains a large angle with its trajectory. In this style the brush trajectory does not necessarily match the medial axis of the shape.

Previous sketch-based techniques assume that the trajectory of the brush footprint matches the medial axis of the stroke shape. This makes them suitable for Chokuhitsu painting style. Figure 2 shows a typical example where such approaches fail since users have no control over the thickness of the stroke. This limitation has been compensated either by explicitly specifying the thickness or by using pressure sensors [Seah et al. 2005]. The former is not intuitive and requires significant manual input, while the latter is based on specific hardware. Instead we propose a shape-driven approach where the user sketches directly the contours of the object to paint. This offers more flexibility and control over the shape of the strokes with no additional interaction as shown in Figure 2. First we describe our approach on simple shapes that can be rendered with a single stroke. The we extend it to complex shapes in Section 3.

2.1 The brush model

We model the shape of the brush footprint as a disc of center c and radius r, and a line connecting the center c to the tip q of the footprint, with $|cq| = 3r$ as proposed in [Mi et al. 2004] and illustrated in Figure 3. In the discrete case, we define a stroke Γ as a series of footprints $\gamma_i = (c_i, r_i, p_i), i = 1 \ldots n$, where c_i, r_i, and p_i are respectively the center, radius, and tip of the $i-$th footprint γ_i. The radius r_i will be referred as the size of the $i-$th footprint. We derive also four other attributes, as illustrated in Figure 3:

- The orientation (θ_i) defined as the angle between the horizontal axis of the coordinate system and the line $(c_i q_i)$ that passes

through the center and the tip of the brush.

- The angular velocity (ω_i) which provides a measure of how jitter the trajectory is.
- The offset angle (ϕ_i) defined as the angle between the brush trajectory at c_i and the line $(c_i q_i)$.
- The distance (d_i) between the tips of two adjacent footprints.

These terms capture the attributes of the brush motion inertia and the effects of friction between the brush and the paper. We use this model to synthesize brush strokes on elementary shapes. In this paper we consider a shape to be elementary if it can be painted with a single stroke. We will derive in Section 3 a simple criteria for testing whether a shape is elementary or not. We also assume that the shape to paint is specified by inputting, or automatically detecting, its boundaries which we automatically classify into *left* (red) and *right* (green) boundaries (see Figure 2), according to whether it is located to the left or to the right of the shape's medial axis.

2.2 Formulation of the painting process

The movement and trajectory of the brush during the process of painting a stroke is subject to several constraints. We divide them into shape constraints and smoothness constraints;

Shape constraints. A Sumi-e painter moves the brush in such a way that the disk of the footprint remains tangent to one boundary while the tip moves along the other. These two constraints insure that the geometry of the shape to paint is faithfully reproduced. The first one can be formulated mathematically by constraining the distance between the footprint center c and the shape boundary to be equal to the radius of the disk r. The second condition is equivalent to minimizing the distance between the footprint tip q and the other boundary of the shape.

Smoothness constraints. By definition, a stroke is a continuous sequence of footprints. Any jump can be interpreted as the end of the current stroke and the beginning of a new one. The quality of a trajectory Γ can be measured by the amount of variation of the brush attributes along the trajectory. Mathematically it is formulated as the minimization of a cost function $E(\Gamma)$ that integrates together the constraints previously defined:

$$E(\Gamma) = \sum_{i=1}^{n} U(\gamma_i). \tag{1}$$

where

$$U(\gamma_i) = \alpha_1 \Delta\theta_i + \alpha_2 \Delta r_i + \alpha_3 \Delta\omega_i + \alpha_4 \Delta\phi_i + \alpha_5 \Delta d_i. \tag{2}$$

The weights $\alpha_i \in [0,1], i = 1 \ldots 5$, are positive and sum to one. For notation convenience we set $U(\gamma_0) = 0$. The five terms of Equation 2 are defined as follows:

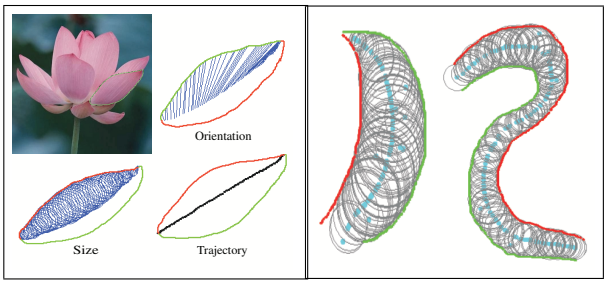
- $\Delta\theta$ penalizes sudden changes in the orientation of the footprint along a trajectory by minimizing the difference in the orientation of two adjacent footprints along the trajectory.

$$\Delta\theta_i = (\theta_i - \theta_{i-1})^2 \tag{3}$$

- Δr_i ensures that two neighboring footprints have approximately the same size by minimizing the difference in size between two consecutive footprints:

$$\Delta r_i = \begin{cases} \frac{(r_i - r_{i-1})^2}{r_i^2 + r_{i-1}^2} & \text{if } r_i \neq 0 \text{ and } r_{i-1} \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

where r_i is the radius of the $i-$th footprint disc.



(a) Effects of the energy function terms. (b) Estimated trajectories on basic strokes.

Figure 4: *(a) Effect of each term of the energy function. The blue-line segments indicate the orientation of the footprint (term $\Delta\theta_i$), the radius of the circles corresponds to the size of the footprint disk (term Δr_i), and the the angular velocity term $\Delta\omega_i$ controls the smoothness of the footprint locations. (b) Two elementary shapes and their estimated brush trajectory.*

- $\Delta\omega$ ensures that variations in the curvature of the brush trajectory is minimal and guarantees its smoothness by minimizing the variation in angular velocity between adjacent footprints.

$$\Delta\omega_i = (\omega_i - \omega_{i-1})^2. \tag{5}$$

where $\omega_{i,j} = \gamma_i / \Delta t$. γ_i is the angle at c_i formed by the segments $(c_{i-1}c_i)$ to $(c_{i+1}c_i)$. Δt is the time step.

- $\Delta\phi$ measures the difference of the offset angle between two consecutive footprints:

$$\Delta\phi_i = (\phi_i - \phi_{i-1})^2. \tag{6}$$

where ϕ_i is the offset angle between $(c_i q_i)$ and the trajectory segment $(c_{i-1}c_i)$.

- Δd measures the variation of distance between the tips of two adjacent footprints.

$$\Delta d_i = \begin{cases} \frac{(q_i q_{i-1})^2 - (q_{i-1} q_{i-2})^2}{(q_i q_{i-1})^2 + (q_{i-1} q_{i-2})^2} & \text{if } |q_i q_{i-1}| \neq 0 \\ & \text{and } |q_{i-1} q_{i-2}| \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Finding the best brush trajectory can now be formulated as a problem of finding a trajectory Γ^* that minimizes the energy E:

$$\Gamma^* = \arg\min_{\Gamma} E(\Gamma). \tag{7}$$

Recall that there are several previous work on Stroke Based Rendering that formulated the painting process as an optimization of an objective function [Hertzmann 2003]. The main difference is that previous work optimize for the stroke placement. Our formulation, however, deals with the shape of the stroke itself; an elementary shape is represented with a single long stroke. Our optimization procedure finds the optimal placement of the footprints which reproduce faithfully the shape.

2.3 Optimal brush trajectory estimation

There are many approaches for optimizing Equation 1. In our case, we discritize the space around the elementary shape by sampling N locations $P = \{p_1, \ldots, p_n\}$. Each location p_i is considered as a node and is connected to its neighbors with an edge. The

weight of this edge, defined by the term $U(\gamma_i)$ called *transition energy* (Equation 2), can be interpreted as the cost of extending the brush trajectory with one time step. This will result in a graph $\mathcal{G} = (P, V, W)$, where P is the set of nodes, V is the set of edges, and W the weights of the edges defined by the transition energy of each pair of neighboring nodes. The optimal brush trajectory that minimizes the cost function 1 is the shortest path that connects the two extremities of the elementary shape to paint. Finding the shortest path is a well studied problem in graph theory. We use Dijkstra shortest path algorithm since the graph is acyclic and all its weights are positive. These properties are enforced during the graph construction. In the following we detail each step of the process.

2.3.1 Graph construction

Given an outline of an elementary shape $\mathcal{S}$ which is drawable with a single stroke, we segment it into a left and a right boundary; we first compute the medial axis of the shape and then classify the points in one side of the medial axis as left and the points in the other as right. The underlying assumption is that the strokes are elongated shapes which are common in Sumi-e paintings. We assume, without loss of generality, that the right boundary is longer than the left one. We proceed as follows:

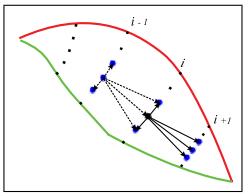


Figure 5: *Discretized space around an elementary shape.*

- We sample an elementary shape into n equidistant slices by dividing the right and left boundaries into n levels. We use the index i to refer to the $i-$th slice (Figure 5).
- We sample from each slice k equidistant points. We use the notation $j^{(i)}$ to refer to the $j-$th point on the $i-$th slice.
- We connect with an edge the point $j^{(i)}$ with the points $(j-1)^{(i)}, (j+1)^{(i)}, j^{(i+1)}, (j-1)^{(i+1)}$, and $(j+1)^{(i+1)}$. The weight of each edge is the transition cost given by Equation 2.
- We append a start node p_s and connect it with an edge to every node in the first layer. The weight of these edges are the transition energies associated to the nodes of the first layer. The nodes of the last layer are set as targets.

We obtain a weighted acyclic directed graph where all the edge weights are positive. The optimal brush trajectory estimation becomes then the problem of finding the shortest path in a graph with a single source and multiple targets. Shortest pathes can be efficiently computed with Dijkstra algorithm.

2.3.2 Implementation details and parameters

The algorithm requires setting two types of parameters;

- **Weights of the terms in the energy function.** experimentally we found that equal weights provide the best performance. In all our experiments we choose $\alpha_i = 0.2, i = 1 \ldots 5$.
- **Parameters of the optimization algorithm.** The optimization procedure requires setting the number of slices and the number of points in each slice. The number of slices is determined by sampling the longest boundary at four-pixel interval. The number of points in the slice is set experimentally to 51.

Finally, the optimization algorithm produces a set of key footprints. We apply B-spline interpolation to generate the in-between footprints. Figure 4 illustrates the trajectory estimation process. The effect of each term of the energy function is shown in Figure 4-(a), and examples of the estimated trajectories on two elementary shapes in Figure 4-(b).

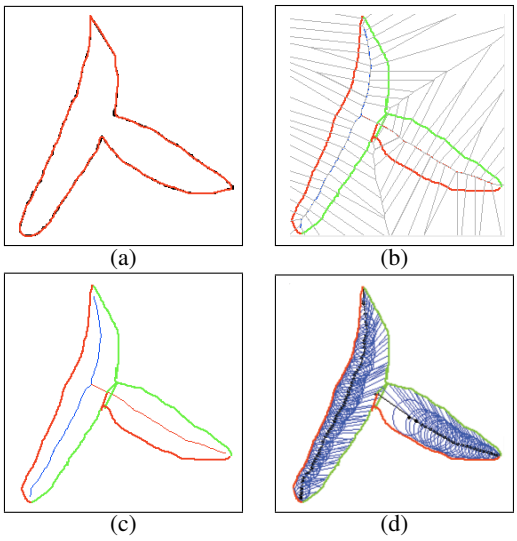


Figure 6: *Handling complex shapes: (a) Input shape, (b) Medial axis-based estimation from the Voronoi diagram of the shape, (c) Decomposition into two elementary shapes, (d) Brush trajectory estimation after decomposition.*

3 Extension to complex shapes

In the previous section we assumed that the user sketches elementary shapes that can be rendered with a single stroke. Painting complex shapes requires several strokes each one corresponding to an elementary component. This is illustrated in Figure 6 where a single stroke cannot capture faithfully the shape. A painter in this situation would use two strokes, each one corresponds to a shape component.

Geometrically, the complexity of a shape is related to the number of branches in its medial axis. We use this property to derive a four-steps procedure for painting complex shapes:

Medial axis computation. First we sample the boundaries of the shape to paint, specified from the user, into N points at one-pixel interval and compute their associated Voronoi diagram. The medial axis is then the set of vertices and edges of the Voronoi diagram lying inside the shape [Attali et al. 2009]. An example is shown in Figure 6(b).

Identifying complex shapes. The complexity of a shape is directly related to the structure of its medial axis. Elementary shapes have a medial axis with a single branch. Complex shapes with several components have a medial axis with many branches. Figure 6 shows an example of a shape with a medial axis of two branches.

Shape decomposition. Theoretically, each branch in the media axis corresponds to one component of the shape. However, in practice, the medial axis is not stable due to noise in the shape boundaries which results in small branches. We eliminate them by smoothing the shape boundaries in a preprocessing step. Next we:

- **Detect the longest branch.** Sumi-e painting, unlike other styles, favors long and smooth strokes over short ones. We use this observation as a decomposition criteria by selecting first the longest branch in the medial axis.

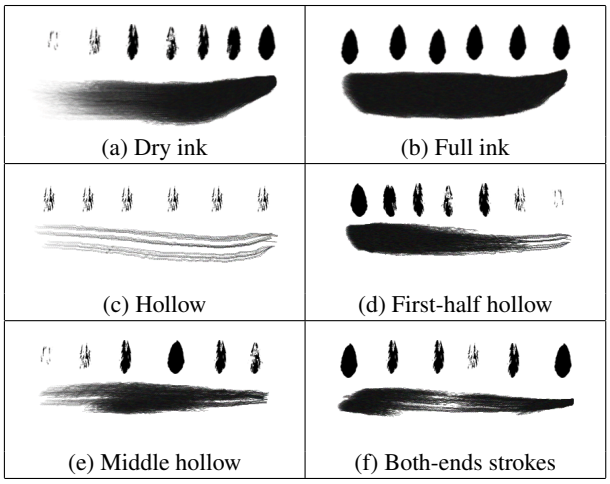


Figure 7: *We generate six different basic stroke styles by combining six types of basic footprints. The small footprints on the top of each stroke show the interpolation order.*

- **Extract the elementary component drawable with a single stroke.** Given one branch of the medial axis, we find its associated contour points by assigning each contour point to the closest skeleton point inside its Voronoi cell.
- **Repeat** the process with the remaining medial axis branches.

Trajectory estimation. Finally, for each elementary component, we estimate the optimal brush trajectory using the procedure described in the previous section.

Figure 6 shows the results of the result of this procedure when applied to a two-stroke Chinese character.

4 Sumi-e style rendering

The trajectory estimation stage provides a sequence of footprint locations, their sizes, and their orientations. The next step is to render this trajectory into a Sumi-e painting style. To do so, we use scanned textures of six real footprints kindly provided by the authors of [Okabe et al. 2005], each one corresponds to one basic stroke. Then we define six rendering styles, each one is characterized by the specific order in which the basic textures are placed and interpolated, as shown in Figure 7. For example, we generate the dry-ink style by placing the six scanned footprints at equidistant locations on the estimated brush trajectory and in the decreasing order of the ink amount. The in-between footprints are generated using B-spline interpolation [Su et al. 2002; Seah et al. 2005]. Formally, given an estimated trajectory Γ:

- We subdivide it into six equidistant segments.
- We place at the extremity of each segment a basic footprint. The order is defined by the style as shown in Figure 7. We use a simple texture mapping scheme that adjusts the size and orientation of the texture to fit the estimated footprint. These techniques are very common in non-photo realistic rendering [Hertzmann 1998; Litwinowicz 1997; Meier 1996].
- We generate footprint textures at every location on the trajectory by B-spline interpolation [Su et al. 2002; Seah et al. 2005].
- Finally, we simulate pigment and water dispersion effects to generate smooth and realistic paintings.

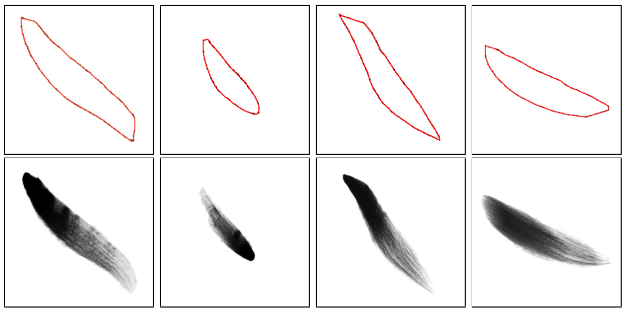


Figure 8: *Rendering results on basic strokes.*

During the painting, the user manually selects the painting style either for the entire artwork or separately for each stroke.

Handling colored paintings and real images. Colors can be easily added to the process described above by letting the user select the color of the ink from a palette. The selected color will be used as the color of the basic footprints which are mapped and interpolated to the estimated trajectories.

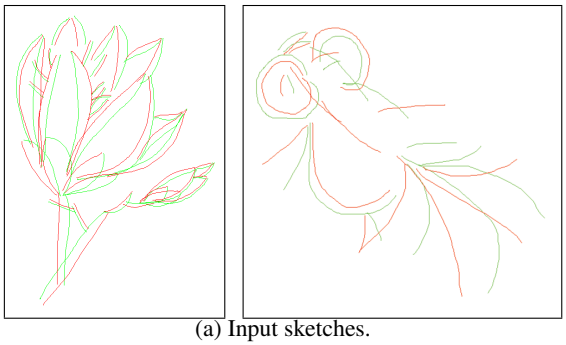
Sketching on an empty canvas requires artistic skills and therefore obtaining beautiful paintings is time consuming. The proposed framework is very suitable for interactively converting real images into Sumi-e-like paintings. Converting real images into Sumi-e requires efficient image segmentation. Efficient automatic image segmentation is still an open research problem. In our current implementation we provide the user with a tool for interactive segmentation. We use Drag and Drop tool [Pérez et al. 2003]; first an initial boundary is drawn around the object to segment. The boundary is then refined until the objet has been detected. Once a region is segmented we process it in the same manner as a user drawn sketches except for the color of the pigment which is copied from the image texture by averaging the colors around the footprint locations. This interactive segmentation procedure allows the user to convert in few minutes real images into Sumi-e paintings as shown in Figure 1.

5 Results

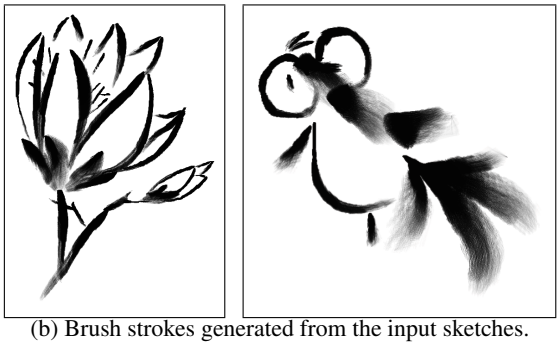
We implemented the approach described in this paper into an interactive painting system which allows the user to sketch the shapes he wants to render into Sumi-e by drawing their contours. This can be done either on a blank canvas or on a real image. In the latter case, the user segments interactively the image which is then rendered into Sumi-e. Figure 1 shows complex examples painted with the algorithm described in this paper.

Sketch to Sumi-e. We provide the user with an empty canvas where he sketches the shapes to paint. Figure 8 shows an example of basic strokes automatically rendered from the user's sketches. Figure 9 shows two complex sketches composed with elementary shapes that can be rendered with a single stroke. In both examples, each contour representing an elementary shape is automatically decomposed into red and green boundaries (first row) then automatically rendered into Sumi-e (second row). More complex strokes are shown in Figure 10 where the user draws freely the contours. They are then automatically decomposed and rendered into Sumi-e. We can also handle color paintings as shown in Figure 11. In this figure, the user selects the ink color every time he draws a shape as is the case when painting with a real brush. This example combines different brush stroke styles and shows the essence of Oriental ink painting as an abstraction of real scenes.

The user sketches on real images. In this section we show that



(a) Input sketches.



(b) Brush strokes generated from the input sketches.

Figure 9: *Examples of Sumi-e paintings rendered in black and white. The user inputs basic elementary shapes.*

Figure 10: *Examples of Sumi-e paintings rendered in black and white (right) from free-hand drawn sketches (left).*

users can easily convert real images into beautiful and naturally looking brush strokes with minimum interaction. Figure 12 shows three input images. The user sketches approximate boundaries which are then automatically optimized to fit the image regions. The Sumi-e rendered results are shown in Figure 1 where the stroke colors are interpolated from the textures of the corresponding regions on the source image. Other examples are shown in Figure 13 where the first column corresponds to the input image and the optimized contours, and the second column to the generated Sumi-e strokes. These examples show that our framework can handle very complex strokes. The ink diffusion process also improves the quality of the colors.

Evaluation. We first compare our results with those that can be obtained with trajectory-based techniques [Okabe et al. 2005]. Figure 2 shows a typical example of basic strokes that can be rendered with our approach using only a mouse while trajectory-based approaches require additional input such as a pressure sensing device. In this example, the trajectory, as specified by the user, may correspond to several possible shapes as shown in the first row. Trajectory-based stroke synthesis can render only the stroke shown

Figure 11: *Abstraction power of Sumie painting.*

in Figure 2(a). Our approach, which requires as input the boundaries of the shape to render, is able to recover the brush trajectory and then automatically fit the stroke to the shape. The output stroke corresponds to the desired shape as shown in Figure 2(b) and (c).

To evaluate the quality on complex paintings we compare our results with the ones obtained using: (1) the state of the art commercial products, such as CorelPainter 11, and (2) trajectory-based techniques. Figure 14 illustrates few examples; the first column is the input outlines which are traditionally used for teaching the writing of Chinese characters. The second column shows the results obtained with our system while the third and last columns show the results obtained using CorelPainter11 and [Okabe et al. 2005] respectively. The strokes we obtained are smoother and fit very well the input shape which clearly its superiority. We conducted also a small-scale quantitative user study following the same approach as in [Xu et al. 2008]. We invited 15 Chinese individual to compare the painting results of our system with trajectory-based tools and authentic paintings drawn by people. We showed them randomly 14 paintings and asked them to tell which ones are real, and which ones are simulated. Among them seven are generated by our system, three using trajectory-based techniques, and four are real Sumi-e paintings. Figure 15 shows the respective accuracies achieved by these individuals. The best score has been obtained by the sixth painting (93%), which is a real one. The third painting, which has been generated by our system, achieved a score of 85.7%, i.e., 85.7% of the individuals have classified it as a real painting. In average, our paintings scored 55.1%, trajectory based paintings 23.8%, and real paintings 73.2%. This shows clearly that our simulated paintings are sufficiently indistinguishable to people from the authentic Sumi-e paintings.

Finally all examples illustrated in this paper have been generated within 3 to 15 minutes including sketching and processing time. Due to space limitation, more experiment results are available at the accompanying video and the dedicated project homepage.

6 Conclusion

We developed in this paper a practicable sketch-based approach for generating strokes in Oriental ink painting style. Compared to previous work, our method allows the user to define the shape of the target stroke. This provides the user better flexibility and control over the shape of the final strokes while maintaining the simplicity of sketch-based techniques. Our experiments show that the pro-

Figure 12: *Sketches drawn by the user and optimized to fit image regions. Sumi-e painting results are shown in Figure 1.*

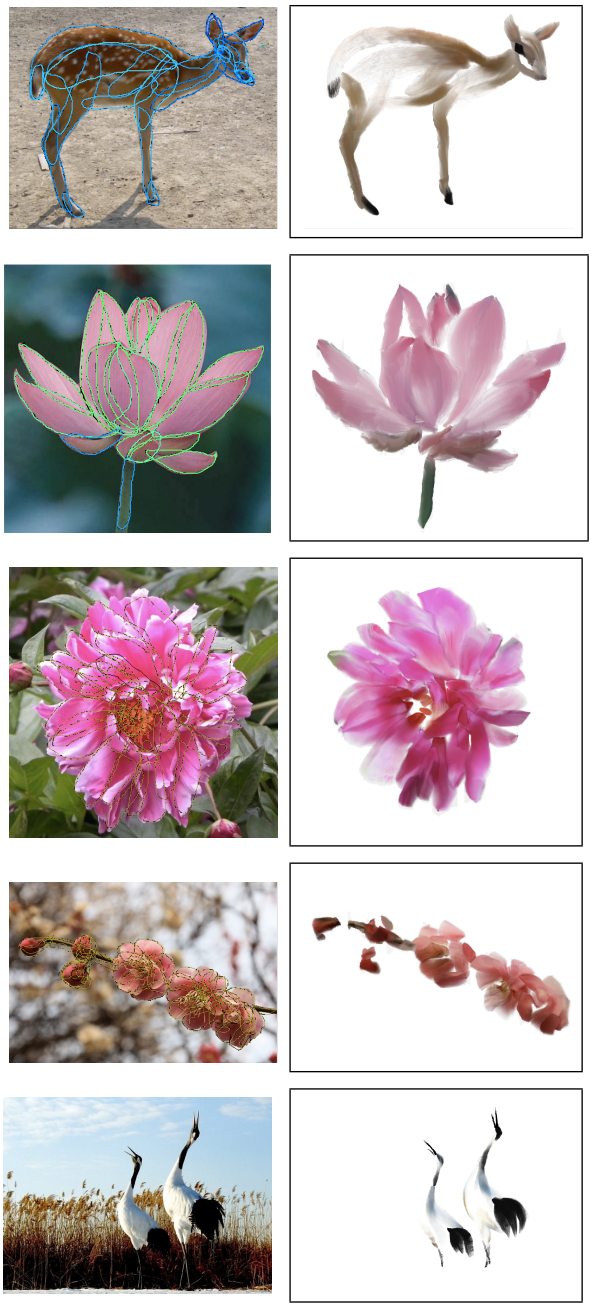


Figure 13: *Examples of real images interactively rendered into Sumi-e paintings. Left: real images and interactively segmented object boundaries. Right: Sumi-e paintings generated with our framework.*

posed approach can generate various types of complex shapes that cannot be generated by specifying only the trajectory of the stroke. It is particularly suitable for converting interactively real images into Sumi-e paintings as shown in the results. Ideally, a novice user would prefer to convert automatically real images and videos into Sumi-e like paintings. This is a very challenging problem and requires high level object detection and scene abstraction. We plan in the future to push further in this direction. Particularly, we believe that low level saliency maps combined with top-level knowledge of the scene can be exploited in automatic scene abstraction for Oriental ink paintings.

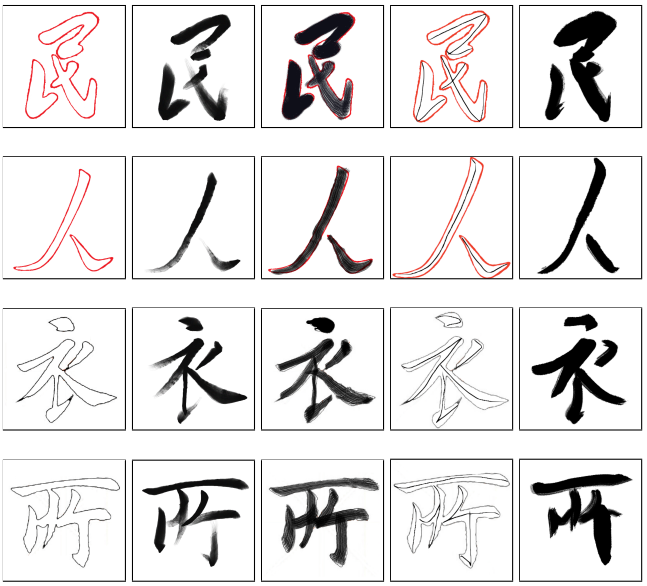


Figure 14: *Rendering Chinese characters with our system (second column), with CorelPainter11 (third column), and trajectory-based approach of Okabe et al. (fifth column). The first column shows the shape outlines. The fourth column shows the trajectories drawn by users when using Okabe et al.'s approach.*

Acknowledgment. The implementation of the Drag and Drop tool for interactive object segmentation is kindly provided by David Gavilan Ruiz. The authors would like to thank particularly Prof. Nelson Max and Prof. Masashi Sugiyama for their suggestions and advices at different stages of this research, Supaporn Spanurattana for sharing her painting expertise, and Junyoung Park for his assistance in making the supplementary video material. Hamid Laga is supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology program Promotion of Environmental Improvement for Independence of Young Researchers under the Special Coordination Funds for Promoting Science and Technology.

References

ATTALI, D., DANIEL BOISSONNAT, J., AND EDELSBRUNNER, H. 2009. Stability and computation of medial axes: a state-of-the-art report. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration, Springer Series on Mathematics and Visualization*, Springer, 109–125.

BAXTER, B., SCHEIB, V., LIN, M. C., AND MANOCHA, D. 2001. DAB: Interactive Haptic Painting with 3D Virtual Brushes. In *ACM SIGGRAPH 2001 video review on Animation theater program*, ACM, USA, 10.

BAXTER, W., WENDT, J., AND LIN, M. C. 2004. IMPaSTo: a realistic, interactive model for paint. In *Proceedings of NPAR*, ACM, USA, 45–148.

BAXTER, B., SCHEIB, V., LIN, M. C., AND MANOCHA, D. 2005. Dab: interactive haptic painting with 3d virtual brushes. In *ACM SIGGRAPH 2005 Courses*, ACM, USA, 180.

CHU, N. S. H., AND TAI, C.-L. 2002. An Efficient Brush Model for Physically-Based 3D Painting. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, 413–421.

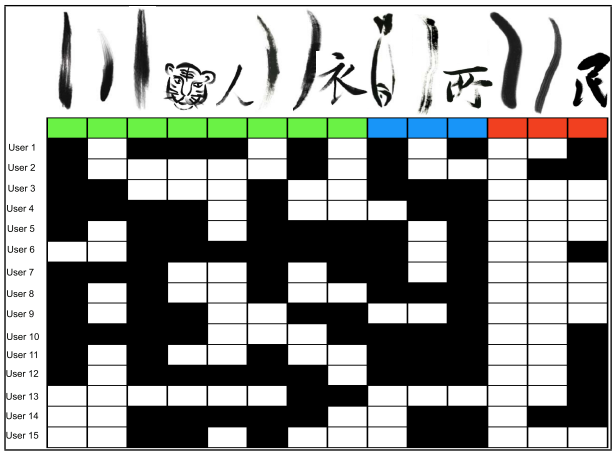


Figure 15: *Results of the user study: green column indicate paintings generated with our system, red for those generated with Okabe et., and blue are real paintings. Rows in the matrix correspond to users' answers. Black means the painting looks real for the user.*

CHU, N. S. H., AND TAI, C.-L. 2004. Real-time painting with an expressive virtual chinese brush. *IEEE Computer Graphics and Applications 24*, 5, 76–85.

COLLOMOSSE, J. P., AY, B. B., HALL, P. M., AND AY, B. B. 2006. Salience-adaptive painterly rendering using genetic search. *International Journal on Artificial Intelligence Tools 15*, 4, 551–575.

COOK, M. T., AND AGAH, A. 2009. A survey of sketch-based 3-D modeling techniques. *Interacting with Computers 21*, 3, 201–211.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. In *Proceedings of SIGGRAPH*, ACM, USA, 769–776.

GOOCH, B., COOMBE, G., AND SHIRLEY, P. 2002. Artistic vision: painterly rendering using computer vision techniques. In *Proceedings of NPAR*, ACM, USA, 83–90.

HAEBERLI, P. 1990. Paint by numbers: abstract image representations. In *Proceedings of SIGGRAPH*, ACM, USA, 207–214.

HERTZMANN, A. 1998. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of SIGGRAPH*, ACM, USA, 453–460.

HERTZMANN, A. 2003. A survey of stroke based rendering. *IEEE COMPUTER GRAPHICS AND APPLICATIONS 23*, 70–81.

HSU, S. C., AND LEE, I. H. H. 1994. Drawing and animation using skeletal strokes. In *Proceedings of SIGGRAPH*, ACM, USA, 109–118.

KONIECZNY, J., AND MEYER, G. 2009. Airbrush simulation for artwork and computer modeling. In *Proceedings of NPAR*, ACM, USA, 61–69.

KOVACS, L., AND SZIRANYI, T. 2004. Efficient coding of stroke-rendered paintings. In *Proceedings of the 17th International Conference on Pattern Recognition*, IEEE Computer Society, Washington, DC, USA, 835–838.

KOVÁCS, L., AND SZIRÁNYI, T. 2004. Painterly rendering controlled by multiscale image features. In *SCCG '04: Proceedings of the 20th spring conference on Computer graphics*, ACM, USA, 177–184.

LI, N., AND HUANG, Z. 2002. Feature-guided painterly image rendering. In *International Conference on Image Processing*, I: 653–656.

LITWINOWICZ, P. 1997. Processing images and video for an impressionist effect. In *Proceedings of SIGGRAPH*, ACM Press/Addison-Wesley Publishing Co., USA, 407–414.

MEIER, B. J. 1996. Painterly rendering for animation. In *Proceedings of SIGGRAPH*, ACM, USA, 477–484.

MI, X.-F., TANG, M., AND DONG, J.-X. 2004. Droplet: a virtual brush model to simulate chinese calligraphy and painting. *J. Comput. Sci. Technol. 19*, 3, 393–404.

OKABE, Y., SAITO, S., AND NAKAJIMA, M. 2005. Paintbrush rendering of lines using HMMs. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, ACM, USA, 91–98.

OLSEN, L., SAMAVATI, F. F., SOUSA, M. C., AND JORGE, J. A. 2009. Sketch-based modeling: A survey. *Computers and Graphics 33*, 1, 85–103.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. In *Proceedings of SIGGRAPH*, ACM, USA, 313–318.

SAITO, S., AND NAKAJIMA, M. 1999. 3D physics-based brush model for painting. In *Proceedings of SIGGRAPH, Conference abstracts and applications*, ACM, USA, 226.

SCHWARZ, M., ISENBERG, T., MASON, K., AND CARPENDALE, S. 2007. Modeling with rendering primitives: an interactive non-photorealistic canvas. In *Proceedings of NPAR*, ACM, USA, 15–22.

SEAH, H. S., WU, Z., TIAN, F., XIAO, X., AND XIE, B. 2005. Artistic brushstroke representation and animation with disk b-spline curve. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, ACM, USA, 88–93.

SHIRAISHI, M., AND YAMAGUCHI, Y. 2000. An algorithm for automatic painterly rendering based on local source image approximation. In *Proceedings of NPAR*, ACM, USA, 53–58.

STRASSMANN, S. 1986. Hairy brushes. In *Proceedings of SIGGRAPH*, ACM, USA, ACM, 225–232.

SU, S. L., XU, Y.-Q., SHUM, H.-Y., AND CHEN, F. 2002. Simulating artistic brushstrokes using interval splines. In *Proceedings of the 5th IASTED International Conference on Computer Graphics and Imaging (CGIM '02, Kauai, Hawaii, August 2002)*, 85–90.

WAY, D.-L., AND SHIH, Z.-C. 2001. The Synthesis of Rock Textures in Chinese Landscape Painting. *Computer Graphics Forum 20*, 3, 123–131.

XU, S., LAU, F. C. M., TANG, F., AND PAN, Y. 2003. Advanced design for a realistic virtual brush. *Computer Graphics Forum 22(3)*, 533–542.

XU, S., TANG, M., LAU, F. C. M., AND PAN, Y. 2004. Virtual hairy brush for painterly rendering. *Graphical Models 66*, 5, 263–302.

XU, S., JIANG, H., JIN, T., LAU, F. C. M., AND PAN, Y. 2008. Automatic facsimile of chinese calligraphic writings. *Computer Graphics Forum 27*, 7, 1879–1886.