Animating Chinese Paintings through Stroke-Based Decomposition[1]

SONGHUA XU[2], YINGQING XU[3], SING BING KANG[4]
DAVID H. SALESIN[5], YUNHE PAN[6], HEUNG-YEUNG SHUM[3]

March, 2006
Technical Report
MSR-TR-2006-33

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

[1]This technical report will appear in ACM TOG, April 2006.
[2]Zhejiang University and Yale University
[3]Microsoft Research Asia
[4]Microsoft Research
[5]Adobe Systems and University of Washington (Most of the work was done while this author was at Microsoft Research.)
[6]Zhejiang University

Abstract

This paper proposes a technique to animate a "Chinese style" painting given its image. We first extract descriptions of the brush strokes that hypothetically produced it. The key to the extraction process is the use of a brush stroke library, which is obtained by digitizing single brush strokes drawn by an experienced artist. The steps in our extraction technique are to first segment the input image, to then find the best set of brush strokes that fit the regions, and, finally, to refine these strokes to account for local appearance. We model a single brush stroke using its skeleton and contour, and we characterize texture variation within each stroke by sampling perpendicularly along its skeleton. Once these brush descriptions have been obtained, the painting can be animated at the brush stroke level. In this paper, we focus on Chinese paintings with relatively sparse strokes. The animation is produced using a graphical application we developed. We present several animations of real paintings using our technique.

CR Categories and Subject Descriptors:
I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding
I.3.3 [**Computer Graphics**]: Picture/Image Generation

Keywords: Computer animation, non-photorealistic rendering, image editing, image-based modeling and rendering, image segmentation.

1. INTRODUCTION

What if paintings could move? In this paper, we propose a way of animating Chinese paintings by automatically decomposing an image of a painting into its hypothetical brush stroke constituents. Most Chinese paintings are typically sparse, with each brush stroke drawn very purposefully [Smith and Lloyd 1997]. Our method is specifically geared for handling paintings that employ brush strokes economically; in addition to most Chinese paintings, other suitable styles include Sumi-e and certain watercolor and oil paintings, such as those of van Gogh.

In Chinese paintings, each stroke is often introduced to depict something specific in the real world. Thus, the output of our stroke-based decomposition of these paintings is a set of graphical objects that are meaningful with regard to the set of real objects the paintings depict. As a result, animators would likely feel comfortable manipulating these graphical objects. In addition, the number of strokes in each painting is usually small, and hence manageable.

Our approach uses segmentation techniques and a library of brush strokes for fitting. The recovered brush strokes are basically vectorized elements, which are easy to animate (Figure 1). In addition to animation, the set of recovered brush strokes can be used for synthesis of paintings or for manipulating images of paintings.

Our automatic stroke decomposition technique has other potential uses. For example, a system utilizing a camera or scanner along with the traditional media of paper, brush,

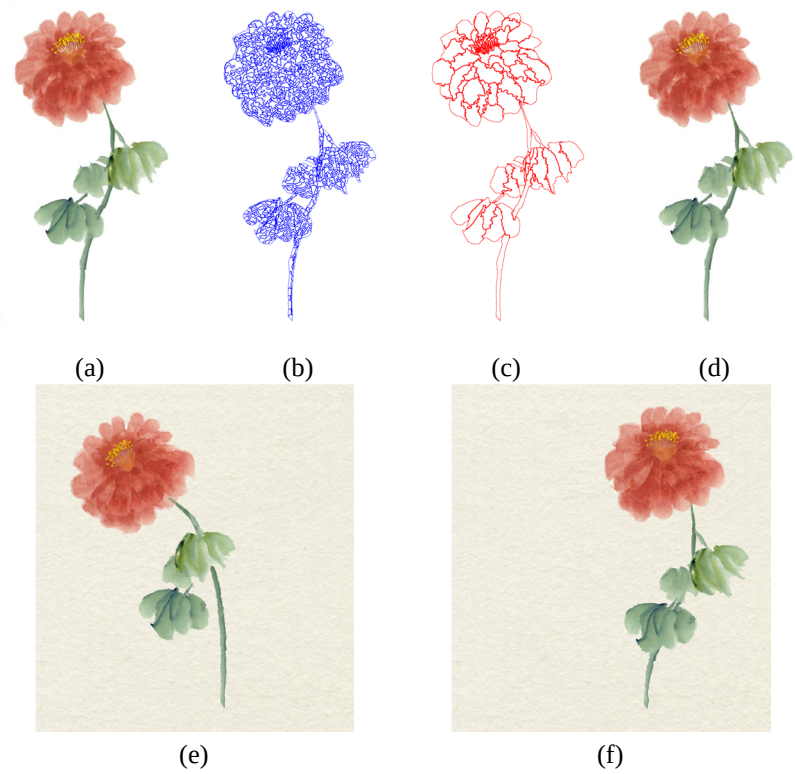


Fig. 1. *Animating a flower painting.* A painting is animated by decomposing it into a set of vectorized brush strokes. The brush strokes are produced by taking the input image (a) and over-segmenting it initially (b). These segments are then merged into coherent strokes (c), which are chosen to match strokes in a "brush stroke library." These strokes are then textured (d) using the input image as a texture source. Finally, the strokes are individually animated as vectorized elements (e), (f).

and paint can be thought of as a kind of "natural tablet" (as opposed to a digital tablet). Another application is compression—an animation sequence of a painting can be more efficiently represented and transmitted across a network. This is a direct consequence of the decomposition process producing a set of vectorized stroke elements. The resulting compressed representation could be used, for instance, to augment a textual chat system with little additional required bandwidth. Finally, the recovered representation could be analyzed to identify artistic style and identity.

To our knowledge, there has been little or no work in automatically decomposing images of paintings into brush strokes. However, several related topics have been explored. One such example is that of "optical character reader" (OCR) systems, where stroke analysis techniques are used for segmenting handwriting purely on the basis of shape (e.g., [Wang and Jean 1993]). Another related line of research is diagram recognition, which includes recognizing engineering drawings [Joseph and Pridmore 1992], mail pieces [Wang and

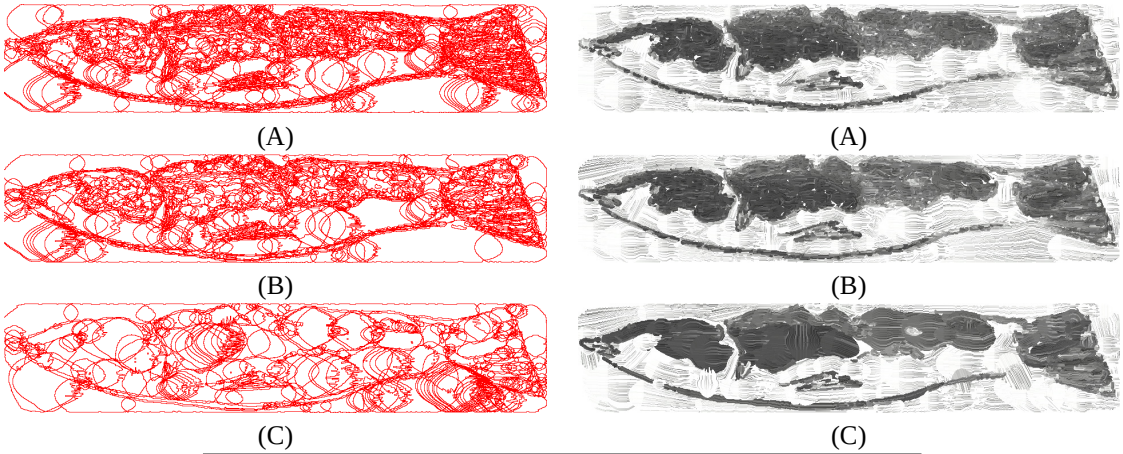
Srihari 1988], sketch maps [Mulder et al. 1988], math expressions [Zanibbi et al. 2002], and music symbols [Blostein and Haken 1999]. However, the targets in diagram recognition are usually limited to symbols or objects drawn using thin lines, which are not nearly as visually rich as brush strokes in paintings.

In computer graphics, electronic virtual brushes have been developed to simulate the effects of brush painting in a computer. One of the earliest works in this area is that of Strassman [1986], where paint brushes are modeled as a collection of bristles that evolve over the course of a stroke. Hsu and Lee [1994] introduced the concept of the "skeletal stroke," which allows strokes to be textured. This idea was later used in a 2D stroke-based animation system called LivingCels [Hsu et al. 1999]. The Deep Canvas system [Daniels 1999] allows brush strokes to be digitally created on 3D surfaces and then animated. The virtual brush for oil painting was proposed by Baxter *et al*. [2001]. The virtual hairy brush for oriental painting was suggested by Xu *et al*. [2002; 2003; 2004]. Kalnins *et al*. [2002] presented a system that supports the drawing of strokes over a 3D model.

Our stroke decomposition work is related to the extensively researched problem of image segmentation in computer vision (see [Jain 1989] and [Forsyth and Ponce 2002]). One particularly relevant approach is that of Neumann [2003]. He proposed an image segmentation technique that uses predefined graphical shape models. However, the technique requires manual selection of corresponding key points, which is non-trivial for large-scale data sets. Wang and Siskind [2003] propose the cut ratio method (a graph-based method) for segmenting images, which supports efficient iterated region-based segmentation and pixel-based segmentation. Marroquin *et al.* [2003] propose a Bayesian formulation for modeling image partitioning and local variation within each region. All these methods either require manual input or assume non-overlapping regions.

Our brush stroke extraction approach involves over-segmenting the image and incrementally merging parts. This technique is common in computer vision and has been used in computer graphics as well. For instance, DeCarlo and Santella [2002] progressively group regions based on similarity of color modulated by region size. Liu and Sclaroff [2001] use a deformable, model-guided, split-and-merge approach to segment image regions. We use a similar approach, except that we consider the similarity with brush strokes from a library as well as color distributions on region boundaries.

There are other object-based editing systems that do not involve brush strokes. In Litwinowicz and Williams's image editing system [1994], users can align features such as points, lines, and curves to the image and distort the image by moving these features. Salisbury *et al*. [1994] developed an interactive image-based non-photorealistic rendering system that creates pen-and-ink illustrations using a photograph as the reference for outline and tone. In Horry *et al.*'s "Tour-into-the-picture" system [1997], the user can interactively create 2.5-D layers, after which flythrough animations can be generated. Barrett and Cheney [2002] developed an image editing system that allows the user to interactively segment out objects

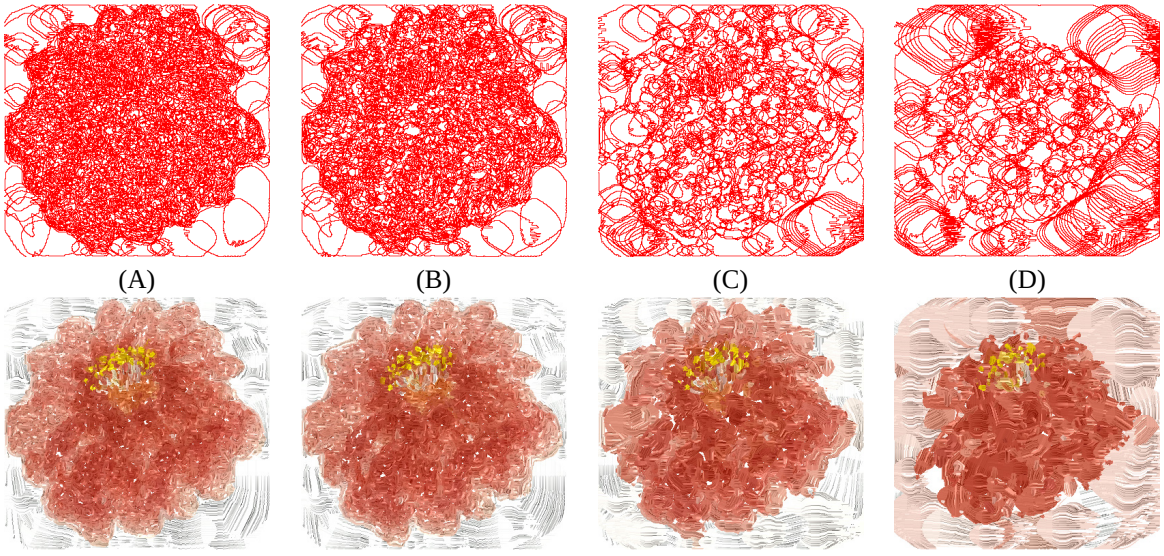


Test	Segmentation level (maximum 255)	Number of strokes
A	255	1769
B	120	1255
C	25	403

Fig. 2. *Stroke extraction results of the fish painting using Gooch's algorithm.* The original painting is Figure 15(a). Three typical segmentation levels are tested: fine (A); medium (B) and coarse (C). The contours of extracted strokes for each test are shown on the left, while their corresponding rendered results are shown on the right. The statistics for these results are listed in the table below.

in the image and manipulate them to generate animations.

The closest work to ours is probably that of Gooch *et al.* [2002] because of some similarity with two important parts of our algorithms — image segmentation and medial axis extraction — and the shared goal of generating brush strokes. However, Gooch *et al.* address a very different problem: they wish to convert one image — photographs or views of synthetic 3D scenes — to another — a non-photorealistic rendering — without preserving the image's *exact* appearance. Moreover, their system's output is a static image. As such, it is not important for them whether or not the extracted strokes are amenable to animation. Also, correct recovery of overlapping strokes is not an issue for them because they are not trying to replicate exactly the appearance of the input image. By comparison, we wish to decompose an image of a painting to separate vectorized elements, or strokes, such that rendering those strokes reproduces the original image's appearance. In addition, in order to facilitate more "natural-looking" animation, the extracted strokes have to be plausible strokes that the artist may have made. Figures 2 and 3 show the results of applying Gooch *et al.* [2002]'s algorithm to two images of paintings. As can be seen, the extracted strokes do not depict anything that corresponds to the real world. This makes "proper" animation of the painting significantly more labor-intensive than if the correct original strokes were extracted. In addition, the original appearance of the painting is not preserved.



Test	Segmentation level (maximum 255)	Number of strokes
A	255	2982
B	120	2345
C	30	942
D	15	496

(e)

Fig. 3. *Stroke extraction results of the flower painting using Gooch's algorithm.* The original painting is Figure 4. Four typical segmentation levels are tested: A–D. The top row shows the contours of extracted strokes, while their corresponding rendered results are shown in the second row. The segmentation parameter and number of strokes extracted are listed in the table below.

2. PAINTING DECOMPOSITION APPROACH

Before we animate a painting, we first decompose its image into a plausible set of brush strokes. A graphical overview of our decomposition approach is depicted in Figure 4, which also shows an example image, the intermediate results, and the final output. The basic idea is simple: we segment the image, use a brush library to find the best fit for each region, and refine the brush strokes found directly from the input image. The brush library used was created with the help of a painter who specializes in Chinese paintings.

2.1 Image segmentation

Given an image of a painting, we first segment the image into regions of similar color intensities. This segmentation is done to speed up the processing for brush decomposition. We tune the mean-shift algorithm [Comaniciu and Meer 2002] to produce an over-segmented image because similarity of color intensity is a necessary but not sufficient condition for

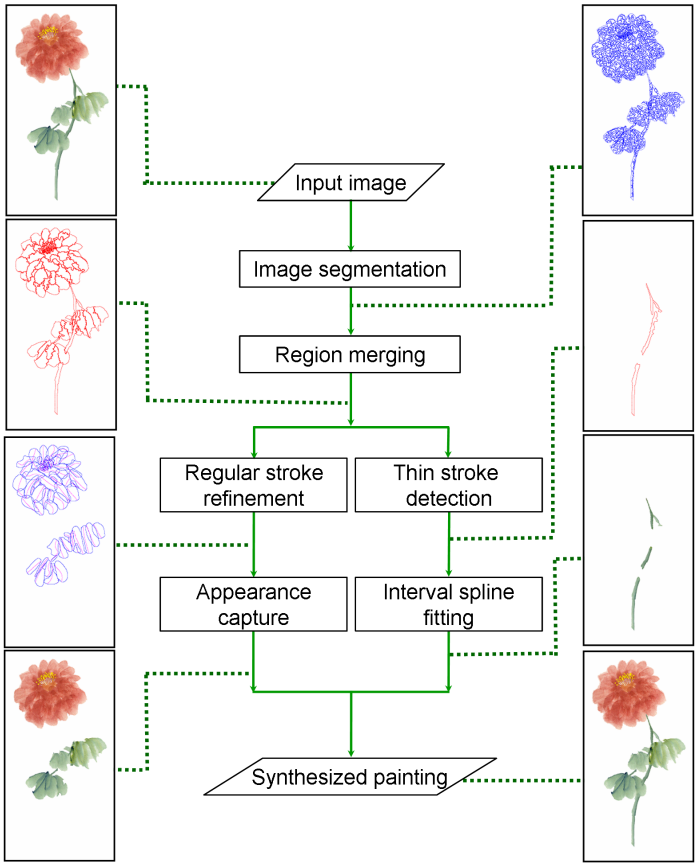


Fig. 4. *Steps involved in our painting analysis and reconstruction approach.*

brush stroke segmentation. The overly conservative segmentation ensures that each region does not straddle multiple brush strokes unless they overlap.

2.2 Stroke extraction by region merging

After over-segmentation is done, we merge contiguous regions that likely belong to the same brush strokes. Our merging process is inspired by domain-dependent image segmentation techniques proposed by Feldman and Yakimovsky [1974] and Tenenbaum and Barrow [1977] (and more recently, Kumar and Desai [1999] and Sclaroff and Liu [2001]). In these techniques, the image is initially partitioned without the use of domain knowledge. Subsequently, pairs of adjacent regions are iteratively merged based on likelihood of being single world objects.

In our approach, the domain knowledge is derived from two sources: the intuition that color gradients are low along brush strokes (the directional smoothness assumption), and a stroke library containing the range of valid stroke shapes (the shape priors). The direc-

Coefficient	κ_g	κ_c	κ_w	κ_m	κ_o
Value	0.083	0.05	16	5	4.5

Table I. The coefficients used in (1) to decompose the painting shown in Figure 4. The values used for the other experiments are similar.

tional smoothness assumption was implemented using average gradients and the difference between the average color intensities along mutual boundaries. The stroke library was obtained by digitizing single strokes drawn by an expert artist, and the resulting shape priors are used to avoid implausible shapes. The shape priors also handle brush stroke overlap, and, as such, our technique goes beyond conventional segmentation.

Before merging takes place, the *region merging criterion* ε (explained shortly) is computed for each pair of adjacent regions. Pairs of adjacent regions are then merged in ascending order of ε. In addition, we merge (or "steal") neighboring regions if the best-fit brush stroke straddles them.

We now define the region merging criterion ε. Suppose we have two adjacent regions γ_i and γ_j. The boundary region of γ_i with respect to γ_j, denoted as $\partial(\gamma_i, \gamma_j)$, is the set of pixels in γ_i that are close to some pixel in γ_j. In our work, "close" is defined as within 3 to 5 pixels of the neighboring regions, and adjacency is defined in the 4-connected sense—a pixel p is adjacent to q if p and q are horizontal or vertical neighbors. Neighboring regions are merged if the following region merging criterion ε, defined as the sum of five terms, is negative:

$$\varepsilon \triangleq \kappa_g \varepsilon_g + \kappa_c \varepsilon_c + \kappa_w \varepsilon_w + \kappa_m \varepsilon_m + \kappa_o. \tag{1}$$

The first two terms, ε_g and ε_c, measure differences in the color distributions of the two regions (gradient and intensity-based measures, respectively), while the next two terms, ε_w and ε_m, measure the shape similarities to those of library brush strokes (the names stand for "weighted shape similarity" and "maximum shape similarity," respectively). Figure 5 illustrates why the terms ε_g, ε_c, ε_w, and ε_m are necessary. The first four constants, κ_g, κ_c, κ_w, and κ_m, are all positive, while κ_o, a threshold offset, is negative. The values of these coefficients used for decomposing the Chinese painting shown in Figure 4 are given in Table I. Similar values are used for the other results.

Dividing both sides of (1) by κ_o yields only 4 independent parameters. Although the ratio between κ_g and κ_c and the ratio between κ_w and κ_m have some effect on the decomposition result, the most significant factor is the ratio between $\kappa_g \kappa_c$ and $\kappa_w \kappa_m$. For paintings with strong edges in the stroke contours, better results are obtained using relatively high values of κ_w and κ_m. In our experiments, we test the thresholds on a small representative portion of the painting before using them on the whole image.

2.2.1 *Comparing boundary color distributions.* To compare two boundary color distributions, we first extract two sets of gradients G_i and G_j, and two sets of color values C_i

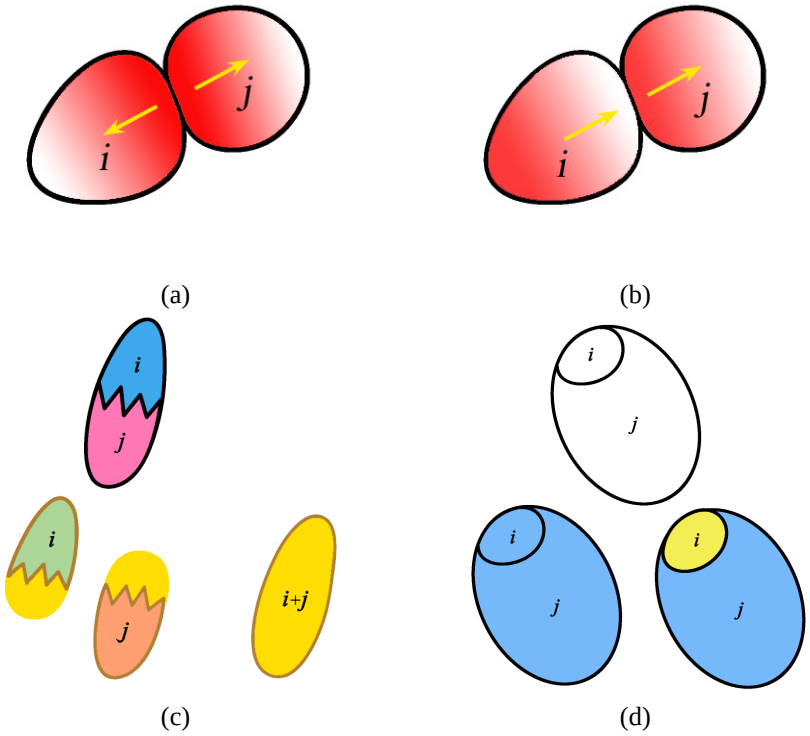


Fig. 5. *Representative cases in the region merging process to illustrate the need for ε_g, ε_c, ε_w and ε_m.* (a) ε_g: Regions i and j have the same color values in the boundary pixels, but they should not be merged because of the sharp difference between the gradients. (b) ε_c: Regions i and j have the same gradients along their common boundary, but they should not be merged due to the significant difference between the color values along the common boundary. (c) ε_w: Here, the combined shape similarity is good enough to overcome the color difference. (d) ε_m: Here, both the component strokes, i and j, and the combined stroke are all good fits with the strokes in the library. In this case, ε_m cancels out ε_w, causing the merging decision to be made based on the boundary color and gradient distributions instead.

and C_j (ranging from 0 to 255 in each color channel) for the pixels in the boundary regions $\partial(\gamma_i, \gamma_j)$ and $\partial(\gamma_j, \gamma_i)$, respectively. Figure 6 shows the boundary regions considered during the region merging process. The color distribution criteria in (1) are defined as

$$\varepsilon_g \triangleq \sum_{r,g,b} \left(\left| \overline{G_i} - \overline{G_j} \right| \arctan \left(\lambda_g \left(\frac{||G_i||}{\sigma^2(G_i)} + \frac{||G_j||}{\sigma^2(G_j)} \right) \right) \right) \tag{2}$$

$$\varepsilon_c \triangleq \sum_{r,g,b} \left(\left| \overline{C_i} - \overline{C_j} \right| \arctan \left(\lambda_c \left(\frac{||C_i||}{\sigma^2(C_i)} + \frac{||C_j||}{\sigma^2(C_j)} \right) \right) \right) \tag{3}$$

where λ_g and λ_c are constants, and $\overline{X}$, $||X||$, and $\sigma^2(X)$ are the mean, cardinality, and variance of X, respectively. In the above equations, by $\sum_{r,g,b}$ we mean the two features are computed for the r, g, and b channels separately and then added together. Note that

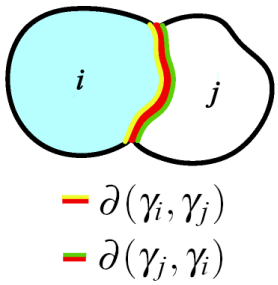


Fig. 6. *Boundary region processing.* Here, regions i and j are being considered for merging. $\partial(\gamma_i, \gamma_j)$ and $\partial(\gamma_j, \gamma_i)$ are the boundary regions used to partially decide if these regions should be merged. The red curve is one pixel thick, and consists of pixels common to both regions i and j. The yellow region is inside region i, adjacent to the red common boundary curve, and 3 to 5 pixels thick. The green region is similarly defined for region j. $\partial(\gamma_i, \gamma_j)$ consists of yellow and red regions, while $\partial(\gamma_j, \gamma_i)$ consists of green and red regions. C_i is the set of colors in the yellow region, and C_j, the set of colors in the green region. Gradients G_i and G_j are computed using pixels in $\partial(\gamma_i, \gamma_j)$ and $\partial(\gamma_j, \gamma_i)$, respectively. Note that here we use only the boundary regions, rather than the entire image region. The local computation strategy is necessary to handle strokes with significant texture variation, e.g., strokes created by dragging a semi-wet brush along a long trajectory.

$||G_i|| = ||C_i||$, since both of them refer to the number of pixels in the same boundary region. Similarly, $||G_j|| = ||C_j||$. In all our experiments, λ_g and λ_c were set to 0.05 and 0.75, respectively.

The gradient term ε_g measures the distance between the average local gradients along the two boundaries modulated by their combined certainties. Each measure of certainty increases with longer mutual boundaries and smaller variances. The positive coefficient λ_g and function $\arctan()$ are used to bracket the confidence value to $[0, \pi/2)$. The color term ε_c functions exactly the same way as ε_g, except that color intensities are compared instead of local gradients. Both ε_g and ε_c measure the homogeneity of the texture variation within each stroke region; we assume the texture variation within a stroke region to be homogeneous.

While there are alternatives to comparing boundary color distributions, our design decisions were governed by simplicity and symmetry of measurement. Estimation of ε_g and ε_c is a computational bottleneck because they are estimated for each adjacent region pair. The Kullback-Leibler divergence (or relative entropy), for example, may be used, but it is asymmetric with respect to the two probability distributions. The Chernoff distance, which is another information-theoretic distance measure, may be also be used, but it requires computation of maxima (a non-trivial optimization problem).

2.2.2 *Using the brush stroke library.* The key to our decomposition approach is the use of a *brush stroke library*. The image of a painting can be segmented in a variety of ways, but the most natural approach would be to segment the image into hypothetical brush strokes

that originally generated the painting. Each brush stroke depicts part of the scene; as such, the output of our segmentation allows the animation of the painting to look more natural.

We generated our brush library by digitizing single brush strokes drawn by an artist with ten years of experience in Chinese painting. This brush library is by no means exhaustive (future work is planned in this area); in our case, the artist drew 62 different brush strokes that he thought were well representative of all the possible ones used in Chinese paintings. Each brush stroke was then binarized and its skeleton computed. Sample brush strokes from this library are shown in Figure 7.

The brush stroke library acts as shape priors to guide the segmentation so as to avoid irregularly-shaped segments. The library also allows us to hypothesize overlaps between brush strokes, which facilitates their separation. Without the brush stroke library, we can extract strokes using *only* the color distribution in the original input image. The decomposition results would likely be irregularly-shaped segments; such segments would be unintuitive from the painter's perspective and thus difficult to animate. (Note that only regions that are relatively thick are processed using the brush library. Strokes that are thin are processed differently; see Section 2.4.)

Figure 8 shows the effect of not using our stroke library, i.e., the stroke decomposition is performed purely based on color distribution without using any shape priors. Stroke decomposition results at different granularities are shown. (The different granularities refer to the different levels of coarseness controlled by segmentation parameter settings.) Regardless of the granularity, the decomposition results are not satisfactory. Ensuring proper brush stroke extraction without an explicit library is highly non-trivial. One could, for example, favor smoothness of the medial axis as well as the radius function along the axis. However, using such a heuristic would produce mostly symmetric, straight blobs, which would appear unnatural for Chinese paintings in general. In addition to producing false negatives, the smoothness preference may also result in strokes that practicing artists would find in-

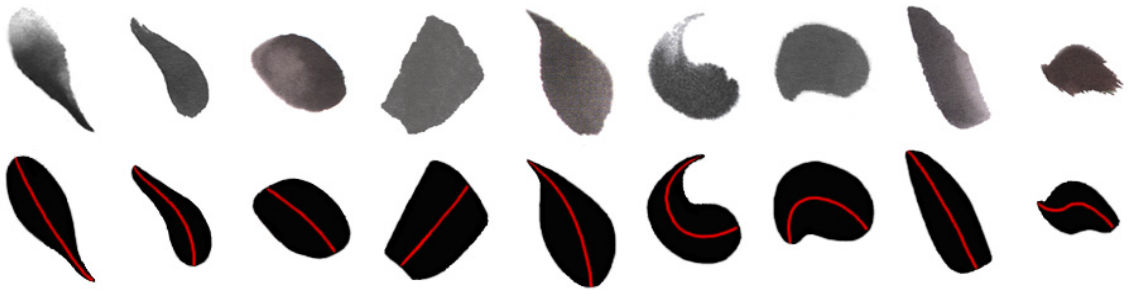


Fig. 7. *Sample library brush shapes.* Only 9 out of 62 shown here. The bottom row displays the modeled brush shapes in the library with their skeletons shown as red curves. The top row shows respective counterparts collected from real paintings.

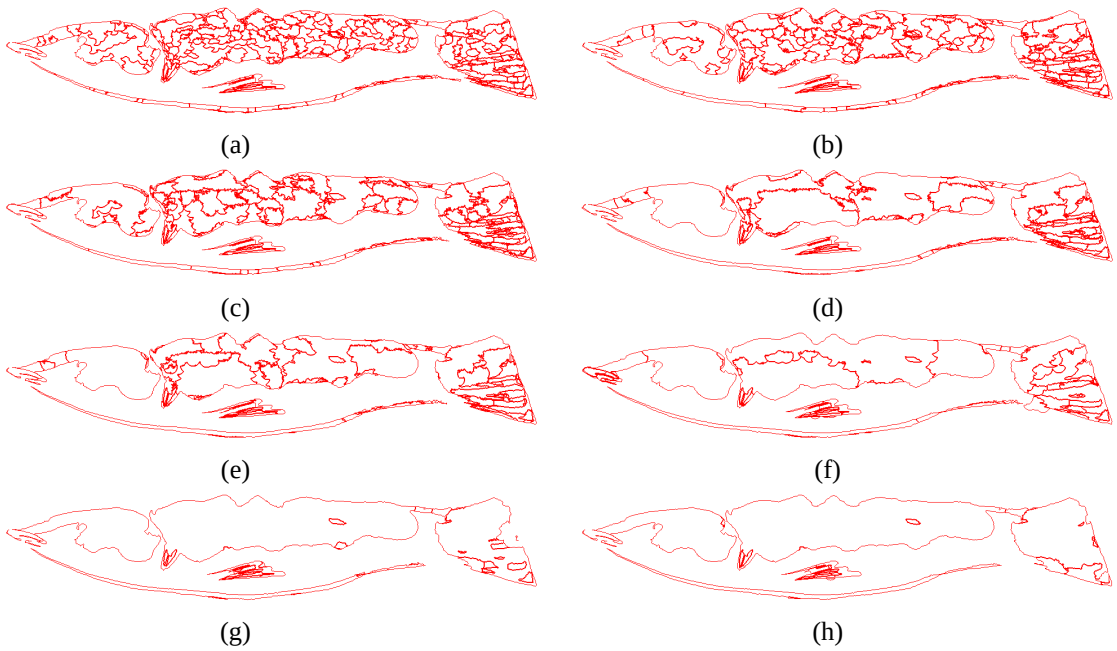


Fig. 8. *Stroke decomposition without our stroke library.* (a)–(h) show stroke decomposition results at different granularities (progressively coarser). Without the stroke library to guide the decomposition, stroke decomposition is uneven, resulting in irregular shapes.

appropriate from an aesthetic point of view. Such strokes could very likely cause incorrect style or artist identification if they were to be analyzed.

2.2.3 *Comparing shapes.* We compare each region to the model strokes in our brush stroke library and find the model brush stroke with the highest shape similarity. Since the scale, orientation, and shift of the observed brush stroke can be arbitrary, we find the best transform to optimize similarity to each library brush stroke. To compute the best transform, we first initialize the shift by aligning the centroids, the orientation by aligning the major axis directions, and the scale by comparing areas. The transform is then refined through gradient descent to maximize shape similarity. The appropriately transformed library brush stroke with the highest similarity with the observed brush stroke is then chosen.

There is extensive work on 2D shape matching; a good survey of techniques is given by Veltkamp [1999]. We chose a simple (but effective) approach to shape similarity in order to keep the computation cost manageable. Specifically, we define a similarity measure $\varphi(\gamma_i)$, which describes how well a given region γ_i fits some stroke in the library:

$$\varphi(\gamma_i) = \max_k \frac{A(\gamma_i \cap T_{ki}\beta_k)}{A(\gamma_i \cup T_{ki}\beta_k)},$$

where $A(X)$ is the area of region X, β_k is the kth stroke in the brush stroke library, and T_{ki}

is the optimal transform (shift, rotate, and scale) used to align β_k with γ_i. The functional $\varphi()$ ranges between 0 and 1—it is 1 when the two shapes are identical. Unlike many shape comparison approaches that compare contours, our shape-based criterion directly makes use of areas. Using areas is more reliable because there is high variability in the detail of the contours of brush strokes. (Pre-smoothing the contour may result in loss of critical information.)

The shape-based criteria in (1) can be defined as:

$$\varepsilon_w \triangleq \frac{\varphi(\gamma_i)A(\gamma_i) + \varphi(\gamma_j)A(\gamma_j)}{A(\gamma_i \cup \gamma_j)} - \varphi(\gamma_i \cup \gamma_j) \tag{4}$$

$$\varepsilon_m \triangleq \max\{\varphi(\gamma_i), \varphi(\gamma_j)\} - \varphi(\gamma_i \cup \gamma_j). \tag{5}$$

Thus, ε_w compares the area-weighted sum of similarity measures associated with fitting two brush strokes against the area-weighted similarity measure for a single brush stroke for the combined regions. A large positive value of ε_w means that it is better to fit the two regions with two brush strokes instead of one. The second measure, ε_m, compares the similarities of the two strokes versus the combined stroke directly; a large value signifies that it is better not to merge the regions. Both ε_w and ε_m are used in objective function (1) because we need to balance two conflicting biases: the bias towards fitting a single brush stroke on the merged regions (ε_w) versus the bias towards preserving current regions that have very good fit with the library (ε_m).

2.3 Stroke refinement and appearance capture

Note that the extracted brush shapes are not the final shapes; the brush strokes in the library are used merely to guide the segmentation process. After the brush strokes have been identified, their shapes are refined using the final segmented regions in the image. The shape of each identified brush stroke is first scaled, shifted, and rotated so as to maximize shape similarity with the corresponding stroke region. The modified shape is then dilated to assume the shape of the brush stroke as much as possible.

Once each shape has been refined, an optimization algorithm is used to produce a maximal-length skeleton within the region. This is accomplished by searching the positions of the two ends of the skeleton along the boundary. The search is done within the vicinity of the skeleton of the best-fit library brush stroke. A piecewise 3^{rd}-degree Bezier curve is used to fit the skeleton.

The appearance of the brush stroke is then captured by directly sampling texture from the image. This is necessary in order to reproduce the appearance of the original painting. Section 3 describes how texture sampling is done.

2.4 Thin brush strokes

Because thin brush strokes are very difficult to model as part of a library, we treat them separately. Each region is categorized either as a regular brush stroke or as a thin brush stroke based on a simple aspect-ratio analysis of the regions. We label a stroke as being thin if the arc length of its skeleton is at least 10 times longer than its average stroke width. Adjacent thin strokes will also be merged if the difference between their average intensities is less than 10 levels and the gradients at their mutual boundaries differ by less than 10%.

Skeletons for thin brush strokes are extracted by using a thinning algorithm [Zhou et al. 1995]. Interval piecewise Bezier splines [Sederberg and Farouki 1992; Su et al. 2002] are then used to represent the thin strokes. A piecewise Bezier curve is used to fit the skeleton of the stroke, with local widths (corresponding to local brush thickness) and intensities recorded at the spline knots. We adapted Schneider's algorithm [1990] for this purpose. In addition to placing spline knots uniformly along the skeleton, we place additional spline knots at locations of high variation of local width or intensity. We resample the width and intensity until their local variations are within acceptable limits.

At this point, let us discuss two important issues associated with our decomposition algorithm. First, what happens when the artist draws strokes that are not in the database? Our algorithm will try to force-fit the best brush stroke shape from the library. If the drawn stroke is only a little different from one of the library strokes and the drawn stroke is close to being a solid stroke (strong boundary edges with little contrast inside), it is likely that only one stroke will be extracted. However, if the drawn stroke is dramatically different from any stroke shape from the library, oversegmentation will likely happen (with possible overlap) because there is no single brush stroke that can fit it well. The second issue relates to the background of the painting. The background need not be white or some other constant color for our algorithm to work; it will work with any uniformly (finely) textured background. If the background is cluttered, it will be treated the same as the foreground objects and decomposed in exactly the same way. Our algorithm will work as long as there is enough contrast between strokes for separation.

3. APPEARANCE CAPTURE AND SYNTHESIS OF SINGLE BRUSH STROKES

3.1 Single-stroke appearance model

Figure 9 shows an overview of how single brush strokes are refined and synthesized (if necessary).

In the case of thin brush strokes, their skeletons are represented by interval B-splines, with local brush widths and intensities recorded at the spline knots. They can be directly rendered using this information.

For regular brush strokes (i.e., those that are not considered thin), we devised a single-stroke appearance model (Figure 10). With the single-stroke model, each brush stroke un-

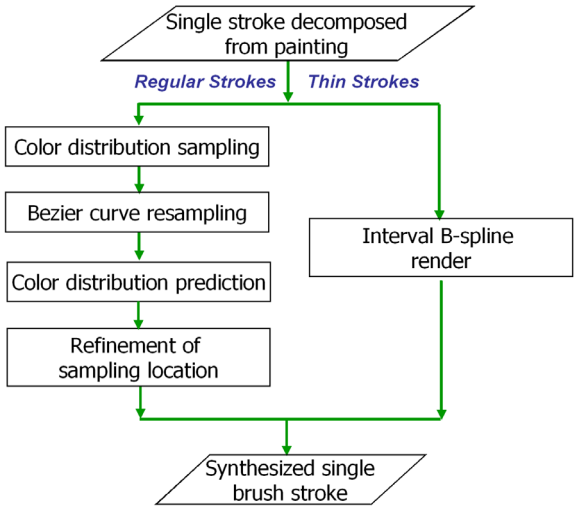


Fig. 9. *Steps in analyzing and synthesizing a single brush stroke.* (The thin and regular strokes are handled differently.)

dergoes a more complicated iterative process, which consists of four steps:

(1) **Color distribution sampling**
Given the shape of the brush stroke (i.e., skeleton and contour), normal lines are computed at regular sample points along its skeleton (Figure 10(c)). The color distribution in *RGB* space of the brush stroke is sampled along each normal, and is represented using a piecewise 3^{rd}-degree Bezier curve. We used Schneider's algorithm [1990] to automatically segment the samples. We assume that the error in fitting the color distribution is Gaussian noise. The modeled Gaussian noise is then added to the fit color distribution to prevent the synthesized appearance from appearing too smooth.

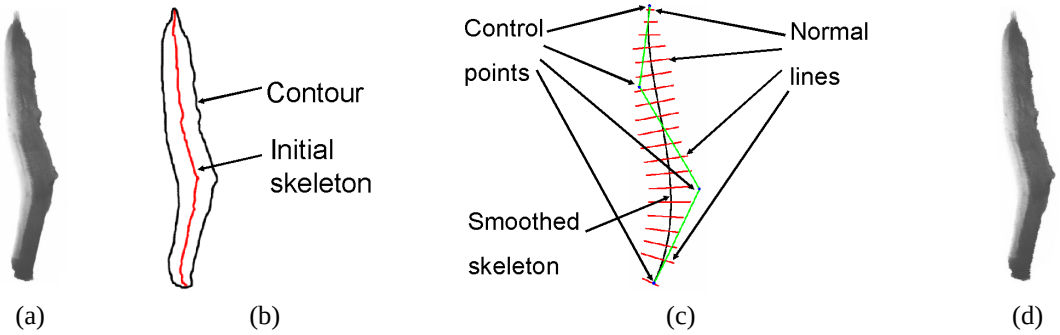


Fig. 10. *Appearance capture of a single brush stroke.* Given an input stroke (a), its contour and skeleton are initially extracted (b). The skeleton is then smoothed, and lines perpendicular to it are sampled from the input image (c). The stroke's appearance can then be generated (d).

(2) **Bezier curve resampling**
The number of Bezier segments may differ for a pair of adjacent normal lines. To simplify the next step of appearance prediction, we resample the number of segments of adjacent normal lines so that they contain the smallest common multiple of the number of samples in the originals. We refer to this process simply as Bezier curve resampling. Note that each sample line has two sets of representative Bezier segments, one to match the previous neighbor, and the other to match the next neighbor. The exceptions are the first and last sample lines, which have only one set of Bezier segments.

(3) **Color distribution prediction**
Given the Bezier approximation of color and noise distributions, we can then synthesize the appearance of the brush stroke. Every pixel in the brush stroke is filled by linearly interpolating the nearest two normal lines. This can be easily done because the number of segments per normal line pair is the same (enforced by Step 2).

(4) **Refinement of sampling location**
The synthesized brush stroke is used to refine the locations of the sampling lines along the brush skeleton. We start off with a sufficiently high sampling density along the skeleton (sampling every pixel is the safest starting point). Sampling lines are chosen at random and tested to see if the degradation is significant when they are removed. If so, they stay; otherwise, they are permanently removed. This process (which is a form of analysis by synthesis) is repeated until either the error between the reconstructed and actual brush strokes is above a threshold, or the number of iterations exceeds a limit.

3.2 Why direct texture mapping is inadequate

A straightforward method to capture and reproduce the appearance of a brush stroke would be to triangulate it followed by texture mapping. One possible tessellation strategy for dividing the brush stroke area into triangle strips is proposed by Hertzmann [1999]. There are two main problems with this approach. First, the shape may be significantly distorted in the process of animation, causing non-uniform warping of texture. Although the texture deformation within one triangle is uniform, the discontinuity of deformed texture would become obvious across the edges of adjacent triangles. In contrast, our stroke appearance model ensures texture smoothness throughout the deformed stroke area because deformation is continuously distributed according to the skeleton of the stroke. Figure 11 compares the results of significant shape distortion.

The second problem with direct texture mapping is that separate tessellation of the source and destination brush stroke shapes would introduce the non-trivial problem of establishing a one-to-one correspondence between the two tessellation results to map the texture. It is possible to handle this problem using a dynamic tessellation algorithm that generates consistent tessellation results, e.g., [Alexa et al. 2000]. However, that would introduce significant additional complexity at the expense of speed. In addition, ensuring minimum

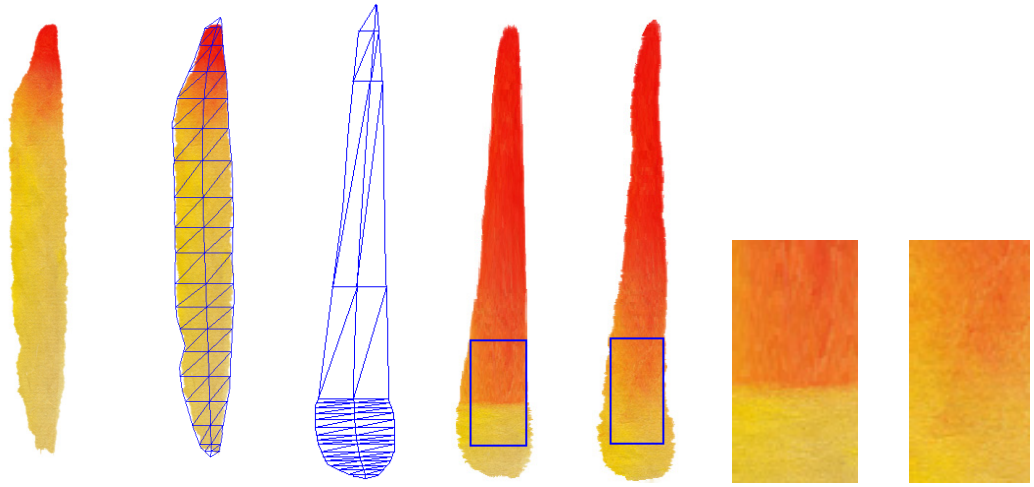


Fig. 11. *Comparison of distortion effects on texture mapped brush stroke and our appearance model.* Given the original stroke (a) and triangulation for texture mapping (b), significant deformation may result during animation (c). Compare the distorted strokes using texture mapping (d) and our appearance model (e). The close-up views of the two respective approaches, (f) and (g), demonstrate that the texture mapped version cannot handle this type of significant distortion as well as our appearance model.

distortion in the brush texture is not obvious. As a result, it is also very hard to guarantee temporal coherence during animation if direct texture mapping is used. Our appearance model does not suffer from these problems.

Our appearance model also naturally supports level-of-detail (LOD) for strokes, and has the capability of predicting appearance of areas that may be partially occluded. This predictive power is used for producing good initial appearances in the process of separating overlapping brush strokes (Section 4).

Although our appearance model outperforms texture mapping in terms of rendering quality, rendering through direct texture mapping is much faster, typically at interactive speeds. Also when the brush shape deformation is not too significant, establishing the one-to-one correspondence between tessellation results for the initial and deformed brush shapes is not very challenging. Thus, we provide two rendering modes in generating an animation clip from a collection of brush strokes extracted from paintings. During the on-line authoring process, texture mapping is used for rendering. This is to enable the animator to manipulate the brush strokes and preview the results in real-time. Once the on-line authoring stage is accomplished, the actual animation clip is generated using our brush appearance model.

4. SEPARATING OVERLAPPING BRUSH STROKES

Brush strokes typically overlap in paintings (see, for example, Figure 12(a)). In order to extract the brush strokes and animate them in a visually plausible way, we have to provide

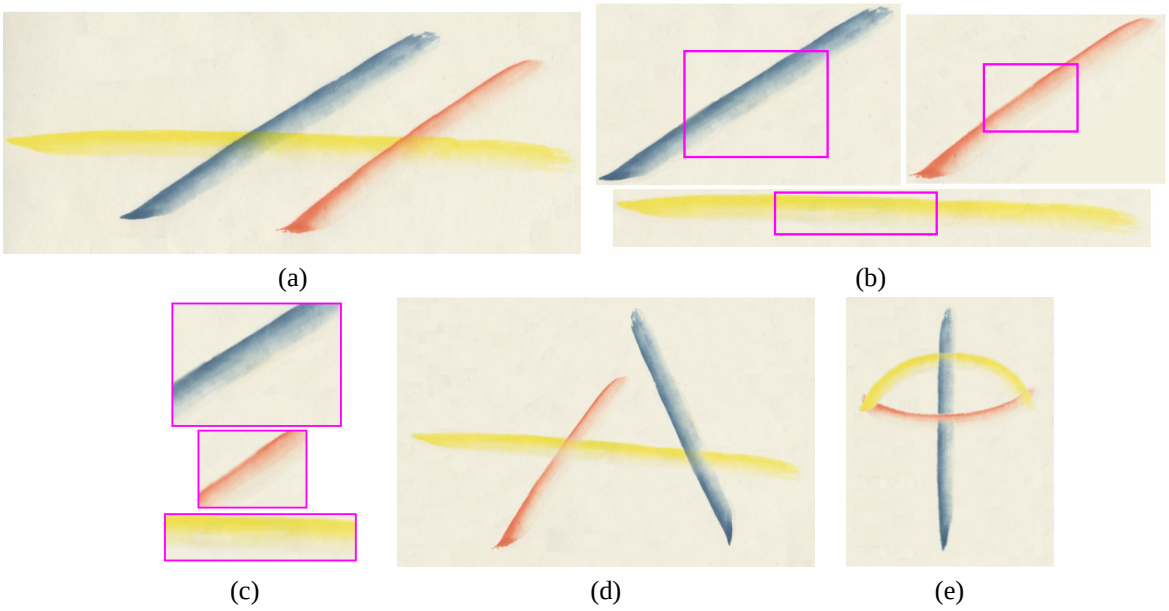


Fig. 12. *Separation of overlapping brush strokes.* Given the original image of three overlapping strokes (a), we obtain the separate strokes (b), with close-up views (c). These strokes can then be easily animated (d), (e).

a mechanism to separate the recovered brush strokes at the overlap regions. Techniques for separation of transparent layers exist in the computer vision literature. For example, Farid and Adelson [1999] show how to separate reflections off a planar glass surface placed in front of a scene. Their method can restore the image of the scene behind the glass by removing the reflections. Unfortunately, their algorithm does not handle the more general problem of image separation, i.e., under arbitrary motion and using only one image (as in our work). Another two-layer separation technique is that of Szeliski *et al*. [2000]. However, they use multiple input images, assume planar motion for the two layers, and apply an additive model with no alpha.

Levin and Weiss [2004] and Levin *et al.* [2004] also studied the problem of separating transparent layers from a single image. In the first approach, gradients are precomputed, following which users are required to interactively label gradients as belonging to one of the layers. Statistics of images of natural scenes are then used to separate two linearly superimposed images. It is not clear if such an approach would work for typical Chinese paintings (which are not photoreal), even with the benefit of manual labeling. The second approach uses a similar framework, except that it minimizes the total number of edges and corners in the decomposed image layers. However, the minimal edge and corner assumptions are not valid for typical Chinese paintings due to the sharp edges of brush strokes. By comparison, our assumption of minimum variation on the texture of brush strokes along

the stroke direction is more appropriate for our domain, and turns out to be effective for automatically separating overlapping brush strokes.

The overlap regions can be easily identified once we have performed the fitting process described in Section 2.3. Once the library brush strokes have been identified, their contours are refined using a similarity transform (scaling, shifting, and rotating) to maximize shape similarity with their corresponding stroke regions. The transformed brush strokes are further dilated enough to just cover the observed strokes in the image, after which the overlapping areas are identified.

We then apply an iterative algorithm to separate the colors at the overlap region. To initialize the separate color distributions, we use the same strategy described in Step 3 of Section 3 to interpolate the colors in the overlap regions using neighboring Bezier functions with known color distributions.

In real paintings, the color distribution at the overlap region is the result of mixing those from the separate brushes. We adapted the mixture model proposed by Porter and Duff [1984] to model overlapping strokes as matted objects because the combination color in the overlapping brush region is generally the result of mixing and optical superimposition of different pigment layers. We did not use more sophisticated models such as the Kubelka-Munk model ([Judd and Wyszecki 1975], pages 420-438) because the problem of extracting all the unknowns from only one image is ill-posed. While the problem is similar to matting (e.g., [Chuang et al. 2001]), matting does not explicitly account for brush stroke texture and orientation. Currently, we separate only pairs of brushes that overlap. Extending our method to handle multiple overlapping strokes is possible at higher computational cost.

Let $\psi_i(p)$ and $\psi_j(p)$ be the colors of two overlapping brush strokes at a given pixel location p, with brush stroke i over brush stroke j; let $\alpha_i(p)$ be the transparency of brush stroke i at p; and let $\psi_r(p)$ be the resulting color at that pixel. We model the appearance of these overlapping strokes using the ("unpremultiplied") compositing equation [Porter and Duff 1984]:

$$\psi_r(p) = \alpha_i(p)\psi_i(p) + \big(1-\alpha_i(p)\big)\psi_j(p). \tag{6}$$

In our case, $\psi_r(p)$ is observed, and so our goal will be to solve for $\alpha_i(p)$, $\psi_i(p)$, and $\psi_j(p)$ at each pixel p for which the strokes overlap. This problem is, of course, underconstrained by this single equation. Thus, we will solve for the values of these three variables that minimize a certain expression encoding some additional assumptions about the appearance of the strokes. In particular, we will assume that the colors ψ_i and ψ_j vary minimally along the lengths of their strokes, and that the transparency α_i varies minimally along both the length and breadth of the upper stroke.

Our objective function, which we will minimize using gradient descent subject to (6), is

as follows:

$$\sum_{p \in \gamma_i \cap \gamma_j} \left(V_i(p) + V_j(p) + \lambda_t T_i(p) \right). \tag{7}$$

Here, V_i can be thought of as the "excess variation" of the color of stroke i along its length, while T_i is the variation of the transparency of stroke i along both its length and breadth.

To evaluate the excess variation, we will refer to the "average variation" $\overline{V_i}(p)$ of the color $\psi_i(p)$ in the parts of the stroke that do not overlap j in which that same color appears. We will call this "exposed" region $\gamma_{i \setminus j}(\psi_i(p))$. Let ℓ be the direction that is parallel to the length of the stroke at p. Then the average variation of the color ψ_i is given by

$$\overline{V_i}(p) \;=\; \frac{1}{A(\gamma_{i \setminus j}(\psi_i(p)))} \sum_{p \in \gamma_{i \setminus j}(\psi_i(p))} \|\partial \psi_i(p)/\partial \ell\|. \tag{8}$$

The excess variation $V_i(p)$ is then given by the amount to which the derivative of the color of stroke i at p along its length exceeds the average variation of that color in other parts of the stroke:

$$V_i(p) \;=\; \max\left\{0, \|\partial \psi_i(p)/\partial \ell\| - \overline{V_i}(p)\right\}. \tag{9}$$

Finally, the variation of the transparency is given by the sum of the derivatives of the transparency both along and across the stroke:

$$T_i(p) \;=\; \|\partial \alpha_i(p)/\partial \ell\| + \|\partial \alpha_i(p)/\partial b\| \tag{10}$$

where b is the direction perpendicular to ℓ.

We generally set λ_t to a small number, around 0.05, since minimizing color variation appears to be more important than transparency variation, in most cases. An example of brush separation is shown in Figure 12. The original brush strokes are shown in (a), and the separated brush strokes are shown in (b).

Our compositing model is related to the Kubelka-Munk model [1975], which assumes that additivity is valid for the absorption and scattering coefficients in the overlapping pigment layers. In other words, $K_r = c_i K_i + (1 - c_i) K_j$ and $S_r = c_i S_i + (1 - c_i) S_j$, where K_r, K_i, K_j are the absorption coefficients in the overlapping area, brush stroke i, and brush stroke j, respectively. S_r, S_i, S_j are the respective scattering coefficients. $c_i, (1 - c_i)$ are the percentages of the amounts of pigment carried by the brush strokes i and j respectively. It is easy to see that our additive compositing equation is a highly simplified version of the Kubelka-Munk model.

The stroke decomposition and animation results show that the simple additive compositing model (6) is rather effective. Our compositing model is significantly less complex than the Kubelka-Munk model. In addition, it is not clear how the Kubelka-Munk model can be reliably used, as it requires the simultaneous recovery of multiple transparent layers from only one image.

A straightforward method for separating overlapping strokes would be to simply discard color information at the region of overlap and reconstruct via smooth interpolation from neighboring regions. However, when an artist paints a single stroke, the color distribution within that stroke is typically not uniform and not smooth. Reconstructing the missing overlap regions by just smoothly interpolating from neighboring regions will not only result in an overly smooth appearance, but also a visually incorrect one. By comparison, our technique accounts for the non-uniformity in color distribution.

5. DECOMPOSITION AND RECONSTRUCTION RESULTS

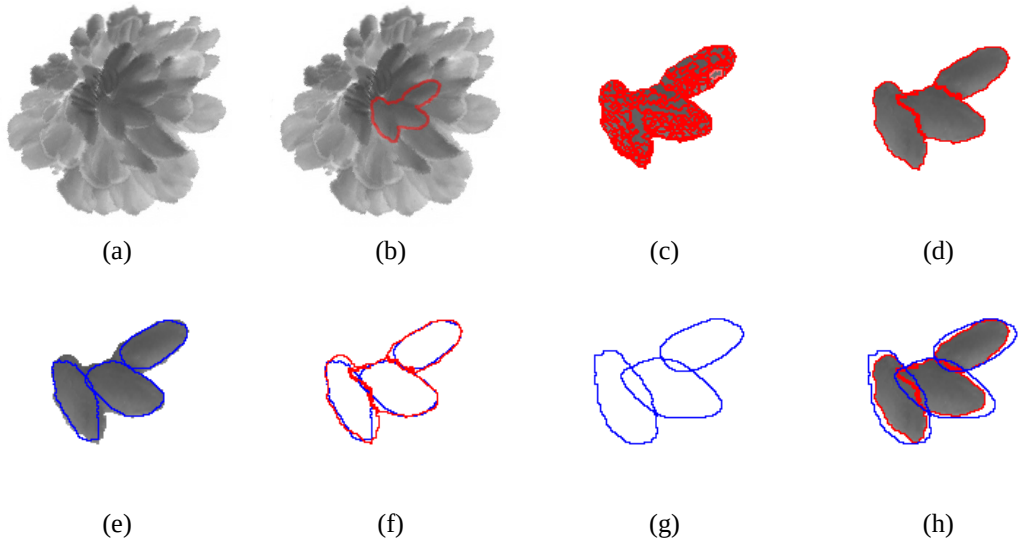


Fig. 13. *The stroke decomposition process.* We illustrate the decomposition process for input (a) by focusing on three brush strokes delineated in red (b). After over-segmentation (c), candidate stroke regions are extracted (d), followed by fitting the best library strokes (e). However, the best fit strokes typically do not completely cover the observed strokes (f), with blue contours representing the fit strokes and red contours representing the observed strokes. To correct the problem, we search (through gradient descent) the scaled rigid transform necessary for each fit stroke to minimally cover the observed stroke (g,h).

Figure 13 shows step by step the process of our stroke decomposition approach on a flower painting. Here, for ease of illustration, we focus on only three extracted brush strokes. Another illustrative example is given in Figure 14(a-i); here, both successful and failed stroke decomposition cases are shown. These cases are discussed in Section 7. Decomposition results for entire paintings are shown in Figures 4 (a different flower painting) and 15 (fish painting). As can be seen in all these examples, the appearance of these paintings have been very well captured using our brush stroke library and appearance model. In the stroke decomposition result shown in Figure 15(e), most parts of the fish body

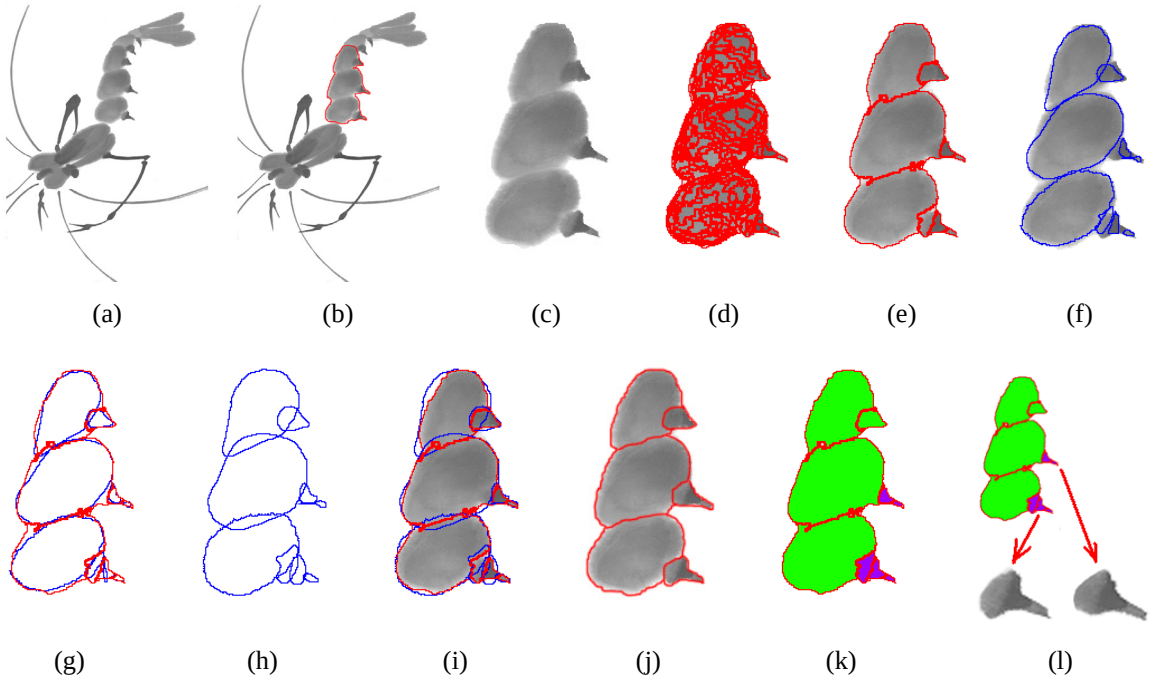


Fig. 14. *Stroke decomposition example for shrimp painting.* Given the input (a), we limit our analysis to three segments of the shrimp's body, delineated in red (b). From (c) to (f), respectively: close-up of original, after over-segmentation, after extracting candidate strokes, and after fitting library strokes. As expected, the best-fit library strokes (in blue) do not completely cover the observed strokes (in red) (g). The refined best-fit library strokes that minimally cover the observed stroke region are shown in (h) and (i). These results are a little different from the manual decomposition results (j), done by the original painter. By superimpositing both results (k), we see that the large brush strokes have been correctly extracted (in green); those that were incorrect were caused by oversegmentation (in purple). The enlarged views of the overly-segmented regions are shown in (l).

that animators would like to manipulate have been extracted as separate strokes. This decomposition is more convenient for animation than the results obtained without using our stroke library (Figure 8). Without using the stroke library, regions are either over-segmented (Figure 8(a-c)), under-segmented (Figure 8(g-h)) or inconveniently segmented (recovered strokes straddling multiple actual strokes, Figure 8(d-f)).

There are three reasons why stroke decomposition using only a simple shape smoothness assumption instead of our stroke library (Section 2.2.3) produces less desirable results. First, strokes with large variations in width and skeleton shape tend to be segmented incorrectly due to the violation of the smoothness assumption. Second, irregular contours of brush strokes (which occur rather often) would be similarly penalized, especially when overlapping occurs. Third, the smoothness assumption is intolerant to noisy or incomplete skeletons. Unfortunately, skeletons are noisy or incomplete in the initial stages of stroke de-

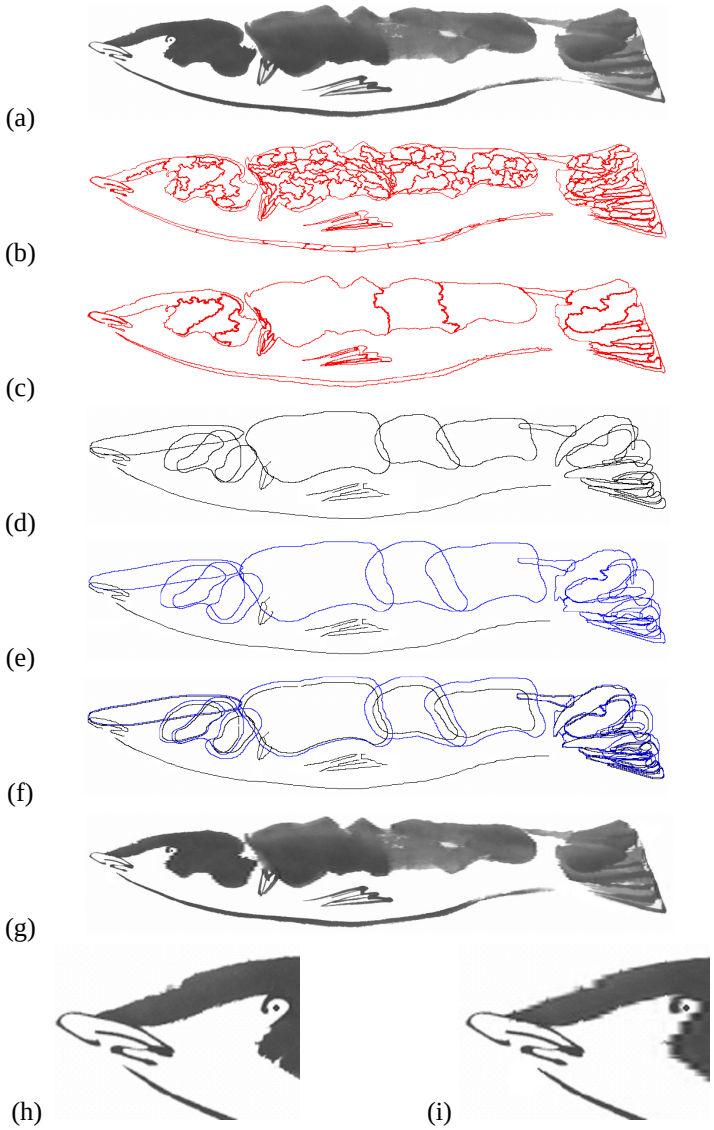


Fig. 15. *Chinese painting of a fish.* The input image (a) is first over-segmented (b). Candidate stroke regions are extracted (c) and fitted with library strokes (d). Note that the thin strokes are represented by their skeletons to distinguish them from regular brush strokes. The fitted regular library strokes are then refined through dilation (e). The dilation effect can be seen by superimposing the strokes (f). The painting can then be synthesized (g). Close-up views of the original (h) and synthesized (i) show the slight blurring effects. Selected keyframes of the animated fish painting are shown in Figure 16.

composition, especially in the vicinity of overlaps. By comparison, our stroke-library-based approach is more robust because it incorporates more accurate domain-specific knowledge in the form of commonly used stroke shapes.

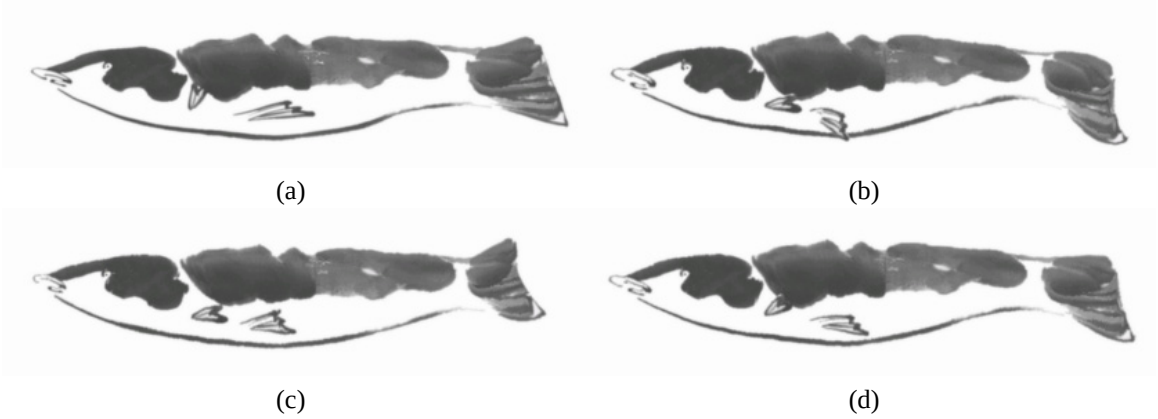


Fig. 16. *Animated fish painting.* Out of 150 frames in the animation clip, we show (a) the 1st frame, (b) the 20th frame, (c) the 60th frame, and (d) the 90th frame.

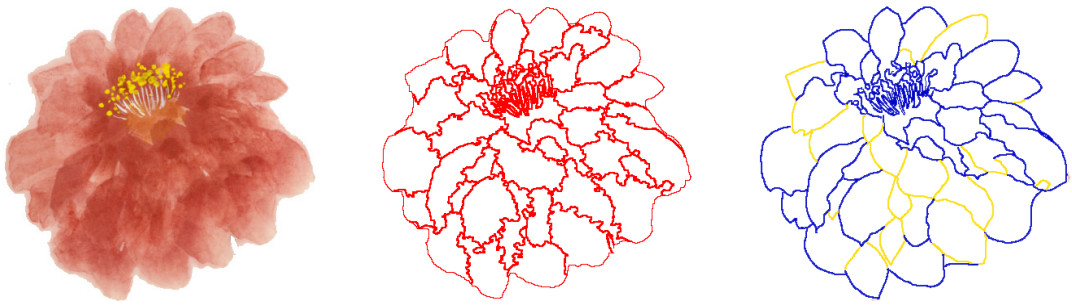
In the example of reconstructing strokes from a Chinese fish painting (Figure 15), it may seem surprising to observe that the eye of the fish is captured in our brush stroke decomposition even though it has not been segmented correctly. (It is difficult to segment correctly here because the size of the eye is very small.) The reason this "works" is that everything within the boundary of the refined brush stroke is considered its texture, and is thus sampled. Note that if overlapping brush strokes are detected, the algorithm described in Section 4 will automatically recover the appearances of the separated brush strokes. It is possible for a refined brush stroke shape to be bigger than it should be, and thus cover a little of the background or other brush strokes (as is in the case of the fish's eye in Figure 15). While an imperfect segmentation will usually not affect the synthesized appearance of a still image, it will however introduce more sampling artifacts during animation.

We have also compared the results of our automatic stroke decomposition with those manually extracted by experts. Figure 17 shows such an example. Typically, while our results are not identical to their manually extracted counterparts, the differences are minor in places where the brush strokes are obvious to the eye. Most of the differences are in locations of significant ambiguity, where even experts have trouble separating brush strokes.

6. ANIMATING PAINTINGS

Figure 18 shows a screen shot of the user interface of our application program designed for animation. The animator can select and move any control point of either the skeleton or the contour of the stroke to be animated. The appearance of the modified stroke is automatically generated by rendering our single-stroke appearance model. The key-frames for the animation can thus be produced through very simple user manipulation.

The in-betweens are generated through interpolation. Note that our animation is done at the *brush-stroke level*. Our brush appearance and mixture models allow the animated



(a) (b) (c)

Fig. 17. *A comparison between our decomposition result with manual stroke decomposition.* (a) The flower portion of Figure 1. (b) The decomposition result (candidate stroke regions). (c) The result of manual decomposition by an experienced Chinese painter who did not create the painting. The blue lines are the edges of strokes extracted with high confidence while lines in yellow are extracted with much less confidence (i.e., deemed ambiguous). Although (b) is different from (c) in a number of places, the major differences are mostly on the yellow lines, where multiple interpretations exist. Our recovered brush strokes agree well in areas where the brush strokes are distinguishable by eye.

painting to be visually acceptable.

Our animation system has the following important features that make it fast and easy to use:

—Addition and removal of brush strokes. Brush strokes from other paintings can be imported and used.

—Grouping of brush strokes for simultaneous manipulation or editing.

—Ability to edit shape and location of the common boundary between two adjacent strokes or to manually decompose a stroke into multiple separate strokes. The latter feature is useful if parts of the decomposition results are not considered fine enough.

—Preservation of stroke connectivity, so that changes to any brush stroke will be appropriately propagated.

—Shape interpolation using critical points (points with locally maximal curvature) on the stroke boundary to better preserve the local shape characteristics during animation.

—Timeline support for editing motion trajectories (e.g., changes in speed or phase). The motion trajectory for each brush stroke can be modified independently.

—The shapes of the brush contour and its skeleton are directly linked; if one of them is manipulated, the other is automatically updated.

—The user can operate directly on either the candidate strokes (Figure 15(c)) or the refined strokes (Figure 15(e)). Note that in Figure 18, groups of candidate strokes are manipulated.

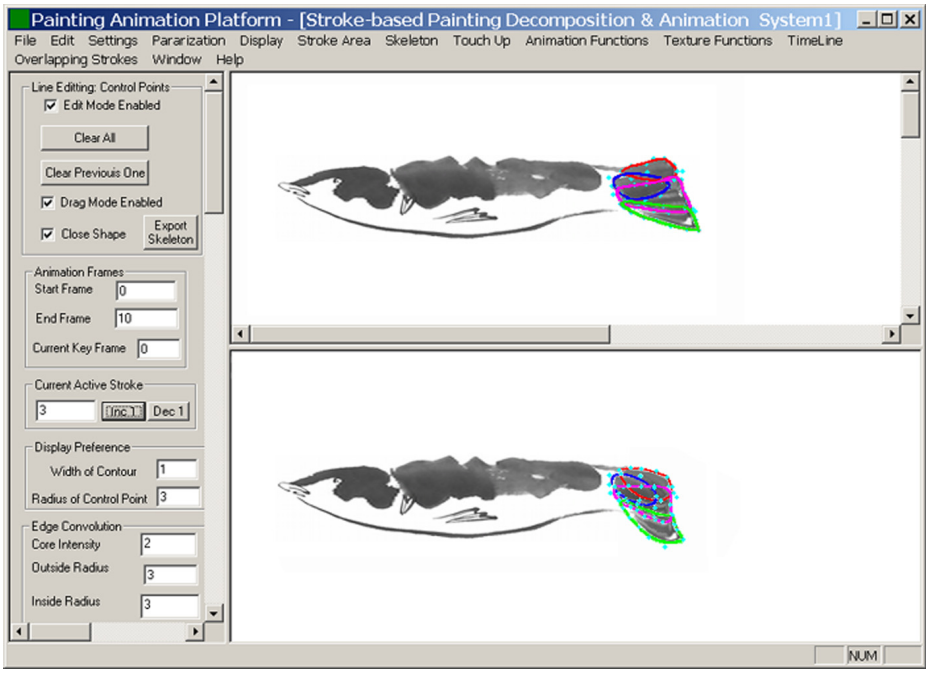


Fig. 18. *Graphical user interface for animation.* This interface uses as input the vectorized strokes generated by our decomposition algorithm. The blue dots are the control points of Bezier curves associated with the groups of brush strokes representing the fish's tail. There are four groups shown here. Note that each group is represented by a different line color, and each group's contour is that of the union of its constituent brush strokes. The shape of each group is manipulated by moving the control points. The top and bottom fish images are generated before and after manipulation, respectively.

Snapshots of animations can be seen in Figures 1 and 19.

It is possible for our stroke decomposition algorithm to make mistakes. It may over-segment (requiring more work to animate), under-segment (resulting in inadequate degrees of freedom for animation), or even produce segments straddling multiple actual strokes. Some of the features in our authoring tool are designed specifically to allow users to manually touch up the decomposition results or correct mistakes.

7. DISCUSSION

There are other possible methods for extracting brush strokes. The simplest is to have the artist draw directly using an interface to the computer, e.g., a haptic interface [Baxter et al. 2001]. Another method would be to record the painting process and infer the brush strokes. The idea would be to digitize the intermediate results of the painting after every stroke or groups of strokes. This may be accomplished by using an overhead camera that sees the entire painting. To avoid the problem of occlusion, the artist could leave the field of view of the camera after each stroke or a small number of strokes. However, the painting process

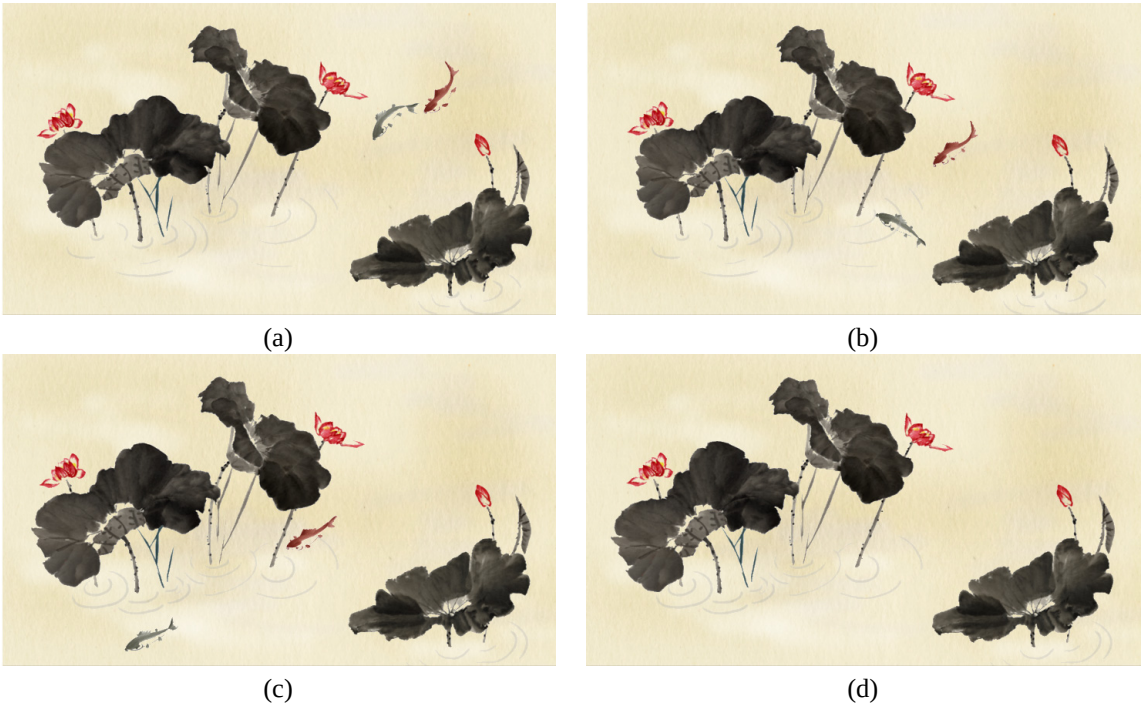


Fig. 19. *Animated lotus pond painting.* Out of the 580 frames in the animation clip, we show the 1st frame (a), the 196th frame (b), the 254th frame (c), and the 448th frame (d). The 1st frame corresponds to the original painting.

is no longer natural. The artist has to adapt to the change in the conditions for painting, be it using the haptic interface or (worse) with the stop-and-paint approach. Furthermore, existing paintings could not be handled.

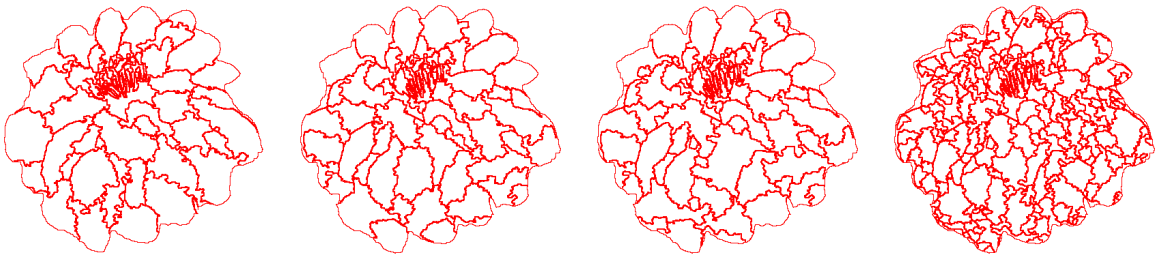
Another straightforward (but more manually intensive) alternative is to design an authoring tool that allows users to merge small stroke segments into meaningful ones or have users roughly delineate the boundaries of strokes. This solution would provide a higher degree of control but comes at the cost of extensive manual effort. Automatic color separation such as ours would have to be incorporated in such a tool (common image editing tools such as Photoshop^{TM} do not have such a feature).

For the animation example shown at Figure 19, it took a single animator 40 hours to use our authoring system to produce a 40-second clip. While there is no record of the exact cost of making the famous 18-minute 1988 video, “Shan Shui Qing” (“Love for Mountains and Rivers”), descriptions of the work involved (e.g., [Chen 1994; Chen and Zhang 1995]) suggest that it required dozens of people working for about a year.

As shown in Section 5, the reconstructed images look very close to the original ones (e.g., Figure 15). On closer examination, however, we can see artifacts introduced by our

brush stroke representation (Figure 15 (h) and (i)). In all our examples, we see that the reconstructed paintings appear fuzzier and the boundaries of the brush strokes are more irregular. This is due to the discrete sampling of the appearance along the brush skeleton (with intermediate areas merely interpolated). In addition, the sampling along the brush skeleton is done independently, i.e., there is no spatial coherence between samples. We plan to investigate sampling techniques that better handle spatial continuity along the brush stroke skeleton.

While many brush strokes appear to be correctly extracted, our algorithm did make mistakes, especially in areas where brush strokes overlap significantly and where the strokes are thick and short. One way of improving this is to extract the brush strokes globally, e.g., ensuring better continuity in the brush stroke direction. In addition, our overlap separation algorithm is currently applicable to overlaps between two brush strokes only. It is not clear how robust our current algorithm is to overlaps of an arbitrary number of brush strokes, but this is a topic we intend to investigate further.



(a) (b) (c) (d)

Fig. 20. *The effect of different library sizes on decomposition.* The example in Figure 17 is used for comparison. (a) is the result using the full library (62 brush strokes), (b) is the result using 31 brush strokes, (c) with 16 brush strokes, and (d) with 8 brush strokes. The brush stroke shapes in the libraries used for (b–d) were randomly chosen from the full library.

What happens if we were to use only a subset of brush stroke library for the decomposition process? Figure 20 shows that the effect is over-segmentation, which worsens as the size of the library is decreased. This is not surprising, as the impoverished versions of the brush stroke library are unable to adequately account for the rich variety of stroke shapes in the painting.

We currently used Chinese-style and watercolor paintings for our work. There are instances where our algorithm did not work well, e.g., Figure 21, where there are extensive overlaps between many short brush strokes. Our brush appearance model is also no longer a good fit when there is large color variation along the brush strokes. Because the decomposition for such a painting would result in a large number of small brush strokes, the process of animating the painting would be very labor-intensive. We have plans to work on images

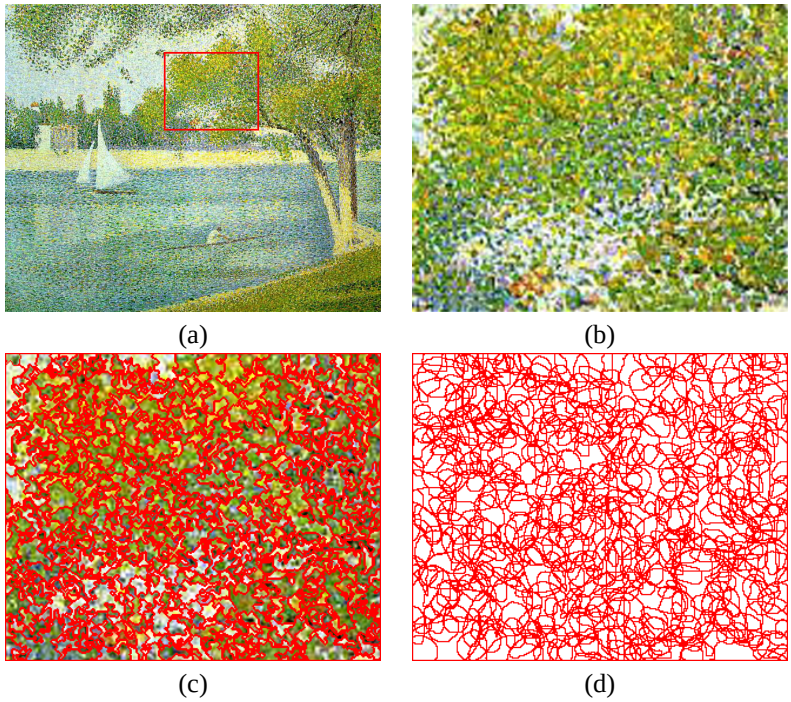
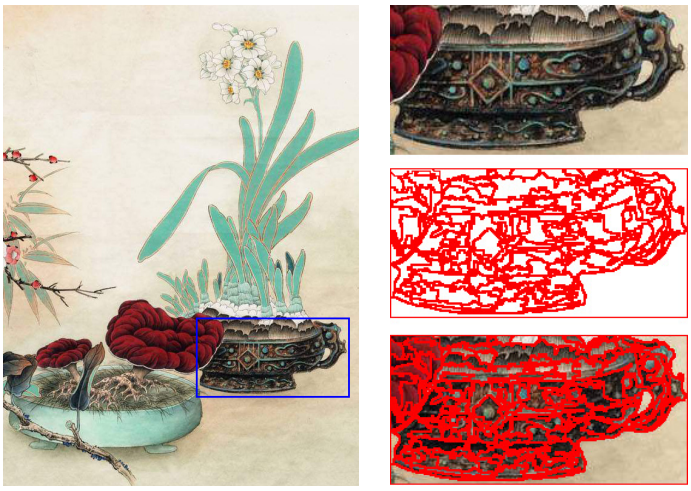


Fig. 21. *A failure example.* One painting that our algorithm failed to decompose properly is "The Seine at La Grande" by Georges Seurat in 1888 (a). The stroke decomposition algorithm resulted in a very large number of small brush strokes. (b) is the close-up view of the area enclosed by the red box in (a). Its corresponding decomposition result is shown in (c), with the final refined brush strokes shown at (d). (Here we do not include the stroke skeletons in the stroke regions for ease of visualization.) Obviously, animating paintings of this kind using our current algorithm would be very labor-intensive. Secondly, our brush appearance model is also no longer a good fit since there is large color variation along the brush strokes. This makes our stroke extraction less accurate.

of paintings with significantly different styles (e.g., Renaissance oil paintings). It is likely that we will need to expand our brush stroke library to handle the different brush stroke styles available in different types of paintings.

Our algorithm can fail even for some Chinese paintings; more specifically, it is unable to decompose paintings drawn in a realistic style. Figure 22 shows such a failure case. In such paintings, both the shapes and the color of brush strokes are deposited strictly according to the actual appearance and geometry of real-world objects. This makes our brush appearance model no longer a good fit since there can be large color variations along the stroke skeletons. In addition, our stroke library would no longer be adequate because the shapes of brush strokes are drawn more arbitrarily to resemble the shapes of real-world objects. To make the painting as realistic as possible, many tiny strokes (which may significantly overlap with each other) are often drawn. This style of painting violates the mainstream

principle of “economical use of brush strokes” for Chinese paintings.



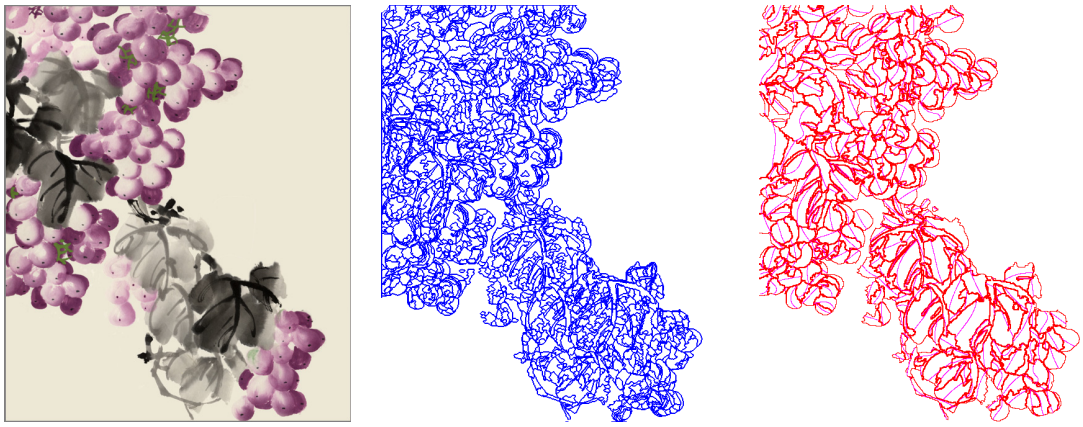
(a) (b)

Fig. 22. *A failure case for Chinese painting.* Our decomposition algorithm usually fails for realistic Chinese paintings such as this one (a). The right side of the figure shows a close-up of the painting, the decomposition result (candidate stroke regions), and the result of superimposing the decomposition result onto the original painting. Note the over-segmentation effect due to the original’s arbitrarily shaped brush strokes and significant color variation.

Unfortunately, even a reasonable decomposition may not always be amenable to animation. This is especially true if the painting involves many small objects clustered closely together and if the animation requires complex interacting motions. A good example of such a case is shown in Figure 23. While the decomposition of the grape painting looks reasonable, animating each grape and leaf relative to other objects would be challenging. For such complicated paintings, it is not clear what a good solution would be.

Currently, our stroke model extracts transparency only at overlapping regions. The proper procedure would be to calculate transparency throughout the overlapping stroke region. Unfortunately, the separation of colors using *a single image* is ill-posed. We handle this by specifying relative transparency at the overlap regions with spatial regularization. One possible solution is to allow users to manually (locally or globally) specify the natural transparency of a stroke. In our current implementation, equation (6) assumes an additive color model, while ink tends to be subtractive. We would like to explore more sophisticated pigment mixing models in the future.

Another limitation of our algorithm lies in the stroke separation and texture modeling steps being independent. As Figure 14(k-l) shows, our algorithm resulted in over-segmentation. This is caused by significant texture changes within the failed regions. Our



(a) (b) (c)

Fig. 23. *A decomposition result unsuitable for animation.* The input image of a grape painting (a), the initial segmented image regions (b), and the extracted candidate strokes with skeletons (c).

current stroke decomposition algorithm is designed under the assumption that texture variation within a stroke region is approximately homogeneous. Unfortunately, for paintings whose pigment/ink diffusion effect is significant, the uniform texture variation assumption no longer holds, leading to the failure cases in Figure 14. To handle such a problem, we would have to incorporate texture modeling in the stroke decomposition process and replace the uniform texture variation assumption with the step of directly fitting a texture model. This would obviously increase the computational cost of the decomposition process.

Our current implementation is unoptimized. For the flower example shown in Figures 1 and 4 (with resolution 560×1080), the times taken for each step on a Pentium III 1.2 GHz computer are: image segmentation (10 secs), region merging (5 hrs), regular stroke refinement (40 mins), regular stroke appearance capture (35 mins), thin stroke detection (10 mins) and interval spline fitting (1 min). We plan to optimize our code to speed up the performance. Note that these steps are done off-line and executed only once. During the actual on-line editing process, rendering of manipulated brush strokes is at interactive rates (30 fps when simple texture-mapping is used for previewing).

Once the brush strokes have been identified, it is entirely possible to analyze the painting by analyzing the brush strokes themselves. By looking at the distribution of directions, stroke thickness, variation of thickness along each stroke, and the color distribution along each stroke and within the painting, the task of identifying the painting style and even the artist may be easier.

Decomposition results with arbitrarily shaped segments complicate the process of animation, and would very likely adversely affect the final visual output quality. Overly small segments increase the amount of effort involved in specifying their motion trajectories. (This effort can be reduced by grouping the small segments, but the grouping operation

can be laborious and tedious as well.) On the other hand, overly large segments straddle multiple brush strokes (wholly or partially), which severely limits the degrees of freedom in animating. In addition, in cases where the large segments straddle partial brush strokes, it is very difficult to ensure correct appearance if the large segments are manipulated independently because the separated brush strokes are distorted differently.

Our current decomposition algorithm does not handle very closely drawn brush strokes very well. In such cases, it may create overly large refined strokes. It is possible to improve the decomposition process by looking at boundary concavities and hypothesizing those to be boundaries of at least two strokes. This is a difficult problem that we intend to investigate further.

Our current rendering implementation uses a simplistic approach to handling overlapping normal lines (which occur when the user puts a sharp kink into the edited stroke, for example). The renderer merely averages the color distributions of the overlapping normal lines. It is not clear what the right solution to this situation is, but the technique used by Hsu and Lee [1994] may be better. Another failure mode occurs when the brush stroke is too distorted, causing severe deformation of the local appearance. Fortunately, these problems do not occur very often.

8. CONCLUSIONS

We have shown a new technique for animating paintings from images. What is particularly interesting is that the animation is done at the *brush-stroke level*.

In order to decompose the image of a painting into hypothesized strokes, we proposed an approach that uses a library of brush stroke shapes to aid region segmentation. Our brush stroke model plays a critical role in allowing the painting's appearance to be captured and subsequently rendered with good fidelity. Finally, our overlap separation algorithm allows full appearance of strokes to be extracted despite the presence of overlaps.

A key contribution of our work is the *automatic* recovery of separate, vectorized brush strokes. This is a tremendous time saver compared to manual segmentation, especially when the painting has hundreds of brush strokes. In addition, proper automatic color separation in the overlap regions is not trivial and is not a feature in common image editing tools such as Photoshop^{TM}. The animation is significantly easier once the segmentation is done.

Experimental results show that our method of decomposition is capable of producing high-quality reconstructions of paintings. The quality of the sample animations also serves to illustrate the effectiveness of our decomposition approach.

9. ACKNOWLEDGEMENT

We would like to especially thank Yuanyuan Su, who produced the animations using our program. Ce Liu and Yanyun Chen contributed some of the paintings used in this paper,

and they helped us better understand the nature of Chinese painting. Steve Lin narrated for our accompanying video, and Dongyu Chao assisted us with the illustrations in this paper, for which we are grateful. We also appreciate Julie Dorsey's gracious support during the revisions of this paper.

REFERENCES

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 157–164.

BARRETT, W. A. AND CHENEY, A. S. 2002. Object-based image editing. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 777–784.

BAXTER, B., SCHEIB, V., LIN, M. C., AND MANOCHA, D. 2001. DAB: Interactive haptic painting with 3D virtual brushes. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 461–468.

BLOSTEIN, D. AND HAKEN, L. 1999. Using diagram generation software to improve diagram recognition: A case study of music notation. *IEEE Trans. Pattern Anal. Mach. Intell. 21*, 11, 1121–1136.

CHEN, H., Ed. 1994. *Encyclopaedia of China, Film Volume (in Chinese)*. Encyclopaedia of China Press.

CHEN, J. AND ZHANG, J., Eds. 1995. *Dictionary of Chinese Films (in Chinese)*. Shanghai Dictionary Press.

CHUANG, Y.-Y. ET AL. 2001. A Bayesian approach to digital matting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*. Vol. II. Kauai, Hawaii, 264–271.

COMANICIU, D. AND MEER, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 5, 603–619.

DANIELS, E. 1999. Deep canvas in Disney's Tarzan. In *SIGGRAPH '99: ACM SIGGRAPH 99 Conference Abstracts and Applications*. ACM Press, 200.

DECARLO, D. AND SANTELLA, A. 2002. Stylization and abstraction of photographs. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 769–776.

FARID, H. AND ADELSON, E. 1999. Separating reflections from images by use of independent components analysis. *Journal of the Optical Society of America 16*, 9, 2136–2145.

FELDMAN, J. A. AND YAKIMOVSKY, Y. 1974. Decision theory and artificial intelligence: I. a semantics-based region analyzer. *Artificial Intelligence 5*, 4, 349–371.

FORSYTH, D. A. AND PONCE, J. 2002. *Computer Vision: A Modern Approach*. Prentice Hall.

GOOCH, B., COOMBE, G., AND SHIRLEY, P. 2002. Artistic vision: Painterly rendering using computer vision techniques. In *NPAR '02: Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering*. ACM Press, New York, NY, USA, 83–90.

HERTZMANN, A. 1999. Introduction to 3D non-photorealistic rendering: Silhouettes and outlines. *S. Green, editor, Non-Photorealistic Rendering. SIGGRAPH 99 Course Notes*.

HORRY, Y., ANJYO, K.-I., AND ARAI, K. 1997. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 225–232.

HSU, S., LEE, I., LOU, C., AND SIU, S. 1999. Software. *Creature House (www.creaturehouse.com)*.

HSU, S. C. AND LEE, I. H. H. 1994. Drawing and animation using skeletal strokes. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 109–118.

JAIN, A. 1989. *Fundamentals of Digital Image Processing*. Prentice Hall.

JOSEPH, S. H. AND PRIDMORE, T. P. 1992. Knowledge-directed interpretation of mechanical engineering drawings. *IEEE Trans. Pattern Anal. Mach. Intell. 14,* 9, 928–940.

JUDD, D. B. AND WYSZECKI, G. 1975. *Color in Business, Science, and Industry*. John Wiley and Sons, New York.

KALNINS, R. D., MARKOSIAN, L., MEIER, B. J., KOWALSKI, M. A., LEE, J. C., DAVIDSON, P. L., WEBB, M., HUGHES, J. F., AND FINKELSTEIN, A. 2002. WYSIWYG NPR: Drawing strokes directly on 3D models. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 755–762.

KUMAR, K. AND DESAI, U. 1999. Joint segmentation and image interpretation. *Pattern Recognition 32,* 4, 577–589.

LEVIN, A. AND WEISS, Y. 2004. User assisted separation of reflections from a single image using a sparsity prior. In *European Conference on Computer Vision (ECCV'2004)*. 602–613.

LEVIN, A., ZOMET, A., AND WEISS, Y. 2004. Separating reflections from a single image using local features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*. 306–313.

LITWINOWICZ, P. AND WILLIAMS, L. 1994. Animating images with drawings. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 409–412.

LIU, L. AND SCLAROFF, S. 2001. Region segmentation via deformable model-guided split and merge. In *Proceedings of the International Conference on Computer Vision (ICCV'2001), Vol. 1*. 98–104.

MARROQUIN, J. L., SANTANA, E. A., AND BOTELLO, S. 2003. Hidden Markov measure field models for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 25,* 11, 1380–1387.

MULDER, J. A., MACKWORTH, A. K., AND HAVENS, W. S. 1988. Knowledge structuring and constraint satisfaction: The mapsee approach. *IEEE Trans. Pattern Anal. Mach. Intell. 10,* 6, 866–879.

NEUMANN, A. 2003. Graphical Gaussian shape models and their application to image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 25,* 3, 316–329.

PORTER, T. AND DUFF, T. 1984. Compositing digital images. In *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 253–259.

SALISBURY, M. P., ANDERSON, S. E., BARZEL, R., AND SALESIN, D. H. 1994. Interactive pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 101–108.

SCHNEIDER, P. J. 1990. An algorithm for automatically fitting digitized curves. *Graphics Gems (A. S. Glassner, ed.), Academic Press,* 612 –626.

SCLAROFF, S. AND LIU, L. 2001. Deformable shape detection and description via model-based region grouping. *IEEE Trans. Pattern Anal. Mach. Intell. 23,* 5, 475–489.

SEDERBERG, T. W. AND FAROUKI, R. T. 1992. Approximation by interval Bezier curves. *IEEE Comput. Graph. Appl. 12,* 5, 87–95.

SMITH, R. AND LLOYD, E. J. 1997. *Art School*. Dorling Kindersley Ltd, Inc.

STRASSMANN, S. 1986. Hairy brushes. In *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 225–232.

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

SZELISKI, R., AVIDAN, S., AND ANANDAN, P. 2000. Layer extraction from multiple images containing reflections and transparency. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*. Vol. I. Hilton Head Island, 246–253.

TENENBAUM, J. M. AND BARROW, H. G. 1977. Experiments in interpretation-guided segmentation. *Artificial Intelligence 8,* 3, 241–274.

VELTKAMP, R. C. AND HAGEDOORN, M. 1999. State of the art in shape matching. In *Technical Report UU-CS-1999-27, Utrecht University, Netherlands*.

WANG, C. AND SRIHARI, S. 1988. A framework for object recognition in a visually complex environment and its application to locating address blocks on mail pieces. *Intl. J. Computer Vision 2*, 125–151.

WANG, J. AND JEAN, J. 1993. Segmentation of merged characters by neural networks and shortest-path. In *SAC '93: Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice*. ACM Press, New York, NY, USA, 762–769.

WANG, S. AND SISKIND, J. M. 2003. Image segmentation with ratio cut. *IEEE Trans. Pattern Anal. Mach. Intell. 25,* 6, 675–690.

XU, S., LAU, F. C., 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., AND PAN, Y. 2002. A solid model based virtual hairy brush. *Computer Graphics Forum 21,* 3, 299–308.

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

ZANIBBI, R., BLOSTEIN, D., AND CORDY, J. R. 2002. Recognizing mathematical expressions using tree transformation. *IEEE Trans. Pattern Anal. Mach. Intell. 24,* 11, 1455–1467.

ZHOU, R. W., QUEK, C., AND NG, G. S. 1995. A novel single-pass thinning algorithm and an effective set of performance criteria. *Pattern Recogn. Lett. 16,* 12, 1267–1275.