The Synthesis of Chinese Ink Painting

翁山然　　　　施仁忠　　　　邱心怡
交通大學資訊科學系 新竹市大學路 1001 號
{gis86522, zcshih, gis87505}@cis.nctu.edu.tw

Abstract

Chinese ink painting is a time-honored traditional art in China. However, it requires experienced painting skills to accomplish a Chinese ink painting. The whole process is time-consuming. Besides, before finishing the entire process we can not get the whole view of the painting. In this paper, we propose a method to synthesize the Chinese ink painti automatically. This method extracts strokes fro painter's ink painting and re-paints it in different styles whatever we want. Using this method we can see resulting images without really performing the painting process. Moreover, the user even requires no any painting skill to achieve this job.

KEY WORDS: non -photorealistic rendering; painting; strokes; physically-based model.

1 Introduction

Traditionally, researches in computer graphics focused primarily on obtaining photorealistic images that are indistinguishable from real-world photographs. However, sometimes p eople may want to see things more aesthetically, rather than snapshot photographs. Recently, the development of non -photorealistic rendering is gettin more and more noticing. Unlike photorealistic rendering, non-photorealistic rendering focuses on creating images that look like paintings and drawings made by artists and designers.

Researches in non-photorealistic rendering can be classified into two categories. The first one [3,4,6,8,12] focuses on simulating and modeling artist's brushes accurately. Algorithms in this way generally produce realistic brush strokes on canvas. These models are often computationall expensive and require much interaction with the user for the specification of position, shape, size, dire ction, color and other parameters.

Strassmann's hairy brushes model [12] provided a basis for researches on modelin brush works. His work describes the physical properties of brush material and well simulates the effect of bristle brush on painting. It uses a one-dimensional footprint sweeping on the paper along a spline, which is defined by a set of control points. Each control point is described by a two-dimensional coordinates and the corresponding brush pressure. Each stroke has a specific amount of ink that deposited along the spline. A library of brushes, together with dips and papers, are used to generate various strokes.

Horace and Helena [6] modeled the physical processes of brush stroke to synthesize calligraphic characters. Their work aimed at calligraphic character writing simulation. Their model consists of the physical geometry of writin brush, the dynamic movement, and the amount of ink. Several aspects of brush writing, such as stroke ending and stroke turning shapes, are achieved successful ly and the results looked much close to real calligraphic characters.

Jintae [8] used a 3-D brush to simulate oriental black-ink paintings. He introduced the concept of "soft" brushes. The brush model can simulate the bristle shape variation dynamically according to the forces imposed on it from the canvas. His algorithm gets 3-D information from the elasti deformation of bristles on the paper and produces strokes by brush movement rather than editing control points of curves.

The second category [5,9,10, 1] usually takes an image and applies user-defined patterns or texture maps to it in order to transform this image into a non -photorealistic image of different styles. Images produced by this kind of algorithms usually tend to be mechanical and much different from real hand-made work. However, these models are less computationally expensive comparing with the previous approaches. There have been many researches addressed similar goals.

In this paper, we focus on the synthesis of Chinese ink painting. The Chinese ink painting is a traditional art in China with very long history. There are various paintin techniques and each gives people different kinds of impression. However, many tools and skills are required during the painting process. Moreover, if some p eople want to see paintings of different styles, we should re -paint the entire painting once again. Thus we may spend much time and effort. The purpose of this paper is to generate different styles of Chinese ink painting according to the image of an Chinese ink painting without really painting it with real brush and canvas. Users only need to specify some features and then computer will complete the entire job.

Chinese ink painting , unlike other paintings such as oil painting and watercolor, seems to be more abstractedly. It can not be done by painting rough and large brush strokes first and then painting over with progressively smaller brush strokes to decorate different levels of detail until the whole canvas has been filled. In order to synthesize a Chinese ink painting, we should first analyze the source ink painting and then try to find out each stroke and its correspondin information, such as the position, size, direction, and gra value. Then we apply this information to the painting algorithm, and finally generate the brush strokes.

Thus, in this paper, we propose a method allowing Chinese ink paintings to be synthesized automaticall according to an input source image. The source image can be a Chinese ink painting or a rough image in silhouette. his method analyzes the source image and extracts each stroke, together with its corresponding information. By referring to the stroke information and brush model, our algorithm can synthesize strokes on the canvas realisticall . Furthermore, a set of parameters can be applied to control the style of an output image. Accordingly, users can get the results in different styles without really painting.

The proposed method has two advantages: One is that it requires no any run -time interaction with users. It is totall automatic and what the user needs to do is simply to specify the style parameters at the beginning of the whole process. The other is that it uses a physically -based brush model instead of simple patterns or texture maps to render the strokes. The resulting images thus would be looked similar to the real hand-made works. Even by using a physically-based model, we could synthesize a Chinese ink painting image in a short time.

The organization of this paper is as follows: The second section describes the method for extractin stroke information from the source image. Section 3 discusses the brush model and the painting algorithm. Section shows the implementation results. Conclusion and future work are given in Section 5.

2 Stroke Extraction.

The whole image of a Chinese ink painting is composed by a set of strokes. These strokes depict objects, such as tree, stone, mountain, house, and etc, in an ink painting. Before synthesizing a painting, we need to extract the characteristics of each stroke, such as its locus, orientation, ink distribution, and etc. In this section, we discuss the extraction of strokes and their corresponding properties from the source image.

In Chinese ink painting, it takes the skills of the Chinese calligraphy. But, unlike the calligraphy, the order of strokes is negligible in painting. A stroke in ink painting is similar to a stroke in calligraphy. It starts from a certain point on canvas and then follows the moving direction of brush. Thus a stroke is the locus of bristles on paper. It is roughly a long, thin region along a certain direction. The width of a stroke is varying along the path and depends on the force s exerted on the brush. Consider Figure 1 in which a sample of stroke is shown.

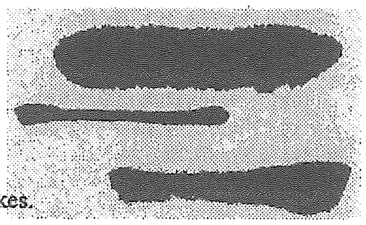


Figure 1 Sample Strokes.

For ease of tracing the locus of a stroke and extracting its corresponding properties, we use a sequence of distinct right hexagons to approximate a stroke. It can be considered as using a hexagon sweeping along the stroke. During sweeping, t he hexagon grows while the width of stroke getting wider. But, the hexagon shrinks while the width of stroke decreases. Then the locus of a stroke is the path of the hexagon's center. By recording the different radii of hexagons, we can obtain the width of a stroke at different positions.

We start the extraction process by first selecting a seed point inside a stroke on the source image. We may treat this point as a degenerate hexagon. Next, the hexagon is growing until its boundary touching the outline of the stroke. The average gray value of pixels inside the hexagon is then calculated. Then the some process is repeated to find the next hexagon. Finally we will get a sequence of hexagons which approximates the stroke. Figure 2 shows an example of stroke extraction. The details are discussed in the following sub-sections.

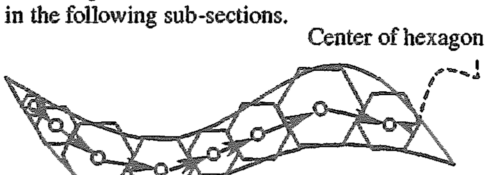


Figure 2 An example of stroke extraction.

2.1 Seed Point Selection

The first step of stroke extraction is to select a seed point inside the stroke. Once the see point is specified, we can extract a locus point of the stroke b hexagon growing. Since the input ink painting is a gray-scale image, the gray values of pixels inside a stroke are smaller than those of outside pixels. We may simply travel the source image pixel by pixel according to the scan -line order, from top t bottom and left to right. At each pixel $P_{i,j}$, we appl a 3×3 filter to it and average the gray values of inside pixels . If the average gray value is less than a threshold value we may select $P_{i,j}$ as a seed point.

2.2 Hexagon Growing

After selecting a seed point, an infinitesimal right hexagon is placed at the seed point. In order to get the width of a stroke, the right hexagon is growing until its edges reac the boun ary of stroke. The collision of hexagon edge and stroke boundary can be detected by checking the variation of gray values. For each step during the growing process, if no any edge touches the stroke boundary, we simply increase the radius of hexagon by δ and then continue the next growing step. The value of δ is determined empirically. If any edge or edges touch the bound ary, the hexagon grows toward the opposite direction. Consider Figure 4. By rotation and symmetry, the growing of hexagon can be classified into six cases. In case 1, only an edge E_1 touches the boundary. The hexagon grows by translating its center along the vector $(\cos 90^\circ, \sin 90^\circ)$ and l is increased by 1. In case 2, two edges, E_1 and E_6, reach the boundary simultaneously. In this case, the translation of the center is along the vector $(\cos 30^\circ, \sin 30^\circ)$. There are three edges in case 3 touching the boundary simultaneously and the translation of its center is the same as that of case 1. In case 4, the center translates along the direction mixed by two opposite directions E_1 and E_5. In cases 5 and 6, the hexagon can not grow anymore since there is no a ny space left to go on. In cases 1, 2, 3, and 4, since the hexagon grows, we re-calculate the summation of gray level of each new edge then repeat the process of growing. An example of the growing process is shown in Figure 5.

After the growing process com pleted, the average gra value of the points inside the hexagon can be obtained easil . Now we get a locus point, which is the center of the hexagon, together with its radius, i.e. l, and the average gray value.

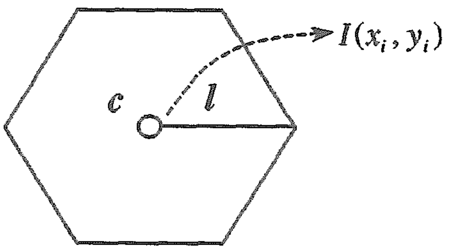
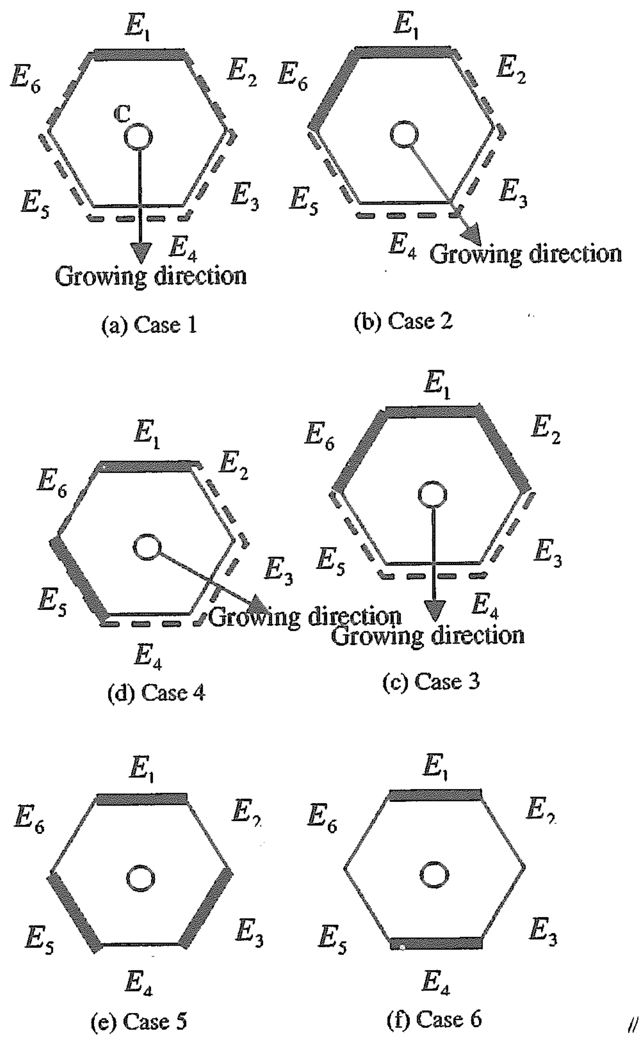


Figure 3 The hexagon mask.

2.3 Hexagon Sweeping

Since each stroke is represented by a sequence of hexagons, after the above process, we are now going to find the next hexagon. First we should locate a seed point. Then, b applying the hexagon growing process, we can obtain the next hexagon. By repeating this proce ss, we can obtain a sequence of hexagons which approximates an input stroke. First we define a weight for each edge of a hexagon. It is the total gray value of pixels on the edge. The weight of an edge totally inside a stroke is smaller than that of a partiall inside edge. Then the seed point of the next hexagon can be selected outside the current hexagon but at a small distance to the edge with the smallest weight.



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4 (e) Case 5 (f) Case 6

Figure 4 Six cases of the growing of hexagon.

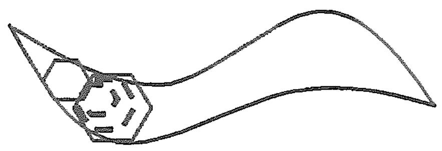


Figure 5 An example of the growing process.

3 Painting Algorithm

After extracting the stroke information, we shall discuss the painting algorithm. In hand-made paintings, there are many different styles. In order to simulate a hand-made painting, we should consider the variety of paintings.

The proposed painting algorithm consists of three components: a physicall -based brush model, a brush moving control mechanism, and an ink depositin mechanism. The brush model describes the shape of brush, the number and locations of bristles. The brush mov ing control mechanism determines the position and orientation of a brush during the painting process. The ink depositing mechanism simulates various real brush-stroke effects such as ink-decreasing effect, ink-soaking effect, bristle fan-out effect, and dr -off effect.

3.1 The Brush Model

Horace [6] used a three-dimensional model for the brush and an ellipse to simulate the intersection of brush and paper. Thus he simulated the aspects of calligraphic very well. Since the painting does not emphasize on the stroke turning so much as calligraphic, we can simulate the contact region of brush and canvas with a circle. Consider Figure 6. Let o and r be the center and radius of the circle respectivel . There are a lot of bristles inside the circle, where each bristle will generate a footprint, denoted as h_i, on the canvas. E quation (1) is the representation of footprint for each bristle in polar coordinates with respect to the center o.

$h_i = (r_i, \theta_i)$ (1)

where

$i = 1, 2, 3, \ldots$ is the index of bristles,

r_i is the distance from o to h_i, and

$0 \le \theta_i < 360$ is the angle between $\overline{oh_i}$ and x-axis.

There are different ways for the footprint generation. For the efficiency of implementation, we first split the central angle of the circle into equal parts. Then, for each angle, we select several points on the radius. In our work, we split the center angle into 50 parts and select 20 points on each radius to generate 1000 footprints totally, as shown in Figure 7. In this way, the footprints can be generated very efficiently. The location of a footprint with respect to center o thus can be represented as

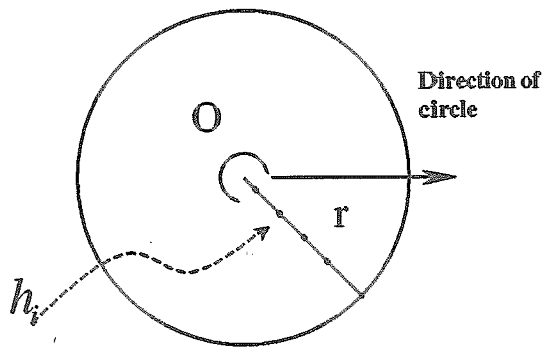


Figure 6 The contact region of brush on canvas.

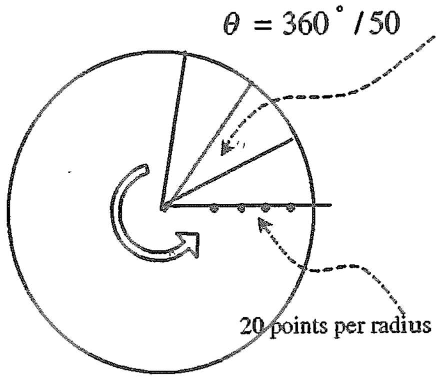


Figure 7 The placement of bristles.

$h_i = (r_i \cos\theta_i, r_i \sin\theta_i)$ (2)

3.2 The Moving Control Mechanism

While painting, we drag the contact circle of brush on canvas along the locus of brush center o to generate strokes The moving control mechanism is to simulate the movement of brush. The movement of brush is controlled by three parameters: the position of brush center, the radius of brush and the orientation at each time instance. We use Bézier curve to describe the path of brush centers. This curve is defined by an ordered set of control points for each stroke.

The radius of the contact circle represents the width of stroke at current position. This value is obtained by the liner interpolation of the radius values on two neighborin locus points. Then we can obtain a smooth contour for a sample stroke at different positions. Figure 8 shows the contour of a stroke at different positions.

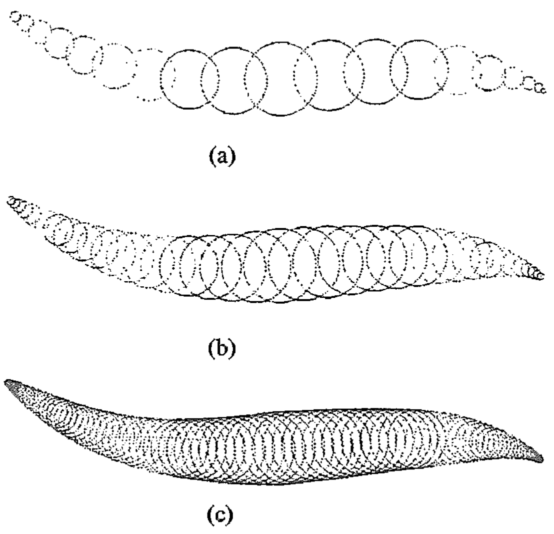


Figure 8 The variation of brush radius along the curve. (a), (b), and (c) show the results of different density.

At each point (x_i, y_i) on the locus of stroke, the brush moves along the tangent direction. Thus the direction of the circle should be aligned to the tangent direction. We use an approximate tangent vector T as follows

$$T(x_i, y_i) = (x_i - x_{i-1}, y_i - y_{i-1}) \quad (4)$$

The footprints inside the circle then rotates an angle θ with respect to the center o. The new coordinate of a bristle after rotation is:

$$(h_{i,x}\cos\theta - h_{i,y}\sin\theta, h_{i,y}\cos\theta + h_{i,x}\sin\theta) \quad (5)$$

where θ is the angle between T and x-axis, as shown in Figure 9.

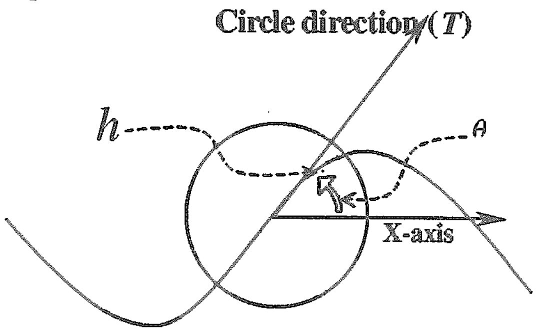


Figure 9 The rotation of circle.

Without rotating the circle, the tracks of bristles may cross each other thus result in the effect as shown in Figure 10. Figure 11 shows the strokes painted in different number of bristles.

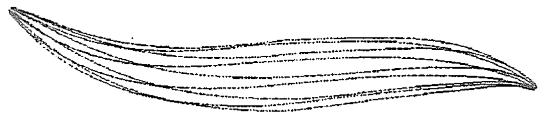


Figure 10 Paint by using four bristles without rotating the circle.

3.3 Ink Depositing Model

As discussed in Sections 3.1 and 3.2, we can obtain the locus of bristles on the canvas. Then we should decide the amount of ink that deposited onto the canvas. In this section, we discuss the ink depositing model to simulate the amount of ink deposited and its effects on canvas.

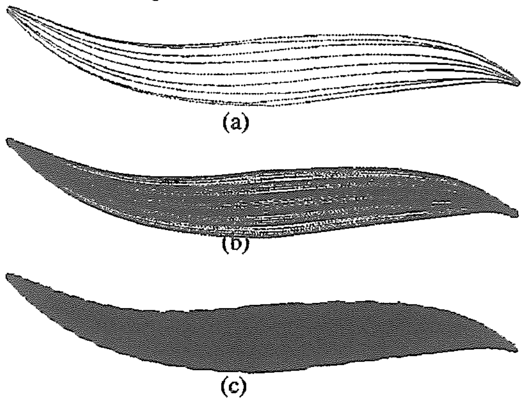


Figure 11 Resulting strokes of brush model.

(a) The same as Figure 10, but the circle is rotated to align with the tangent direction.
(b) Stroke painted by 108 bristles.
(c) Stroke painted by 1000 bristles.

In the ink depositing model, there are four effects of ink deposited onto the canvas from each bristle. These effects are ink decreasing , ink soaking variation, bristle fan-out, and dry-off. The former two effects describe the global behavior of the brush. The others describe the local behavior. The ink decreasing effect determinates the variation of gray level along the brush path. The ink soaking variation affects the gray level variation along the normal direction of the brush path. The bristle fan-out effect is simulated by the difference between each bristle and the dry-off effect simulates the discontinuous effect of strokes due to insufficient ink.

3.3.1. The Ink Decreasing Effect

While a brush moving along the track on the canvas, it deposits ink onto the canvas continuously. The amount of ink remaining on the brush thus decreases. In other words, the ink deposited becomes less and less. This fact results in the difference of gray level from the beginning to the end of a stroke. We define a Decreasing Parameter P_I to implement this effect. Given the initial and final gray values of P_I, we can control the changes of gray level linearly along the path of each bristle. Figure 12 shows the result that only considers the ink decreasing effect.

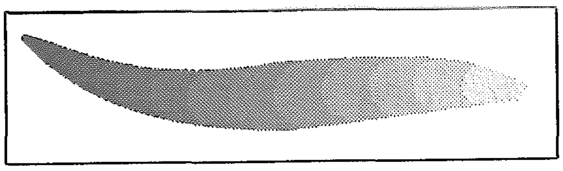


Figure 12 A result consider only the ink decreasing effect.

3.3.2. The Ink Soaking Variation

The ink soaking variation determines the variation of gray values along the normal direction of the brush path. The ink soaking variation is resulted from ink 's different stickiness or. man-made on purpose to paint some strokes such as the trunk of bamboo. This often brings us an illusion of shading on object. The parameter P_2 is used to simulate the ink soaking variation on the brush and its value is from 0 to 255. The mapping from P_2 to bristles is determined by the ratio of $h_{i,y}$ and r. We define several types of the gray-level variation, as shown in Figure 13. Together with the ink decreasing effect, we can generates a lot of different strokes. Figure 14 sho ws four strokes in different styles generated from the skeleton points in Figure 7.

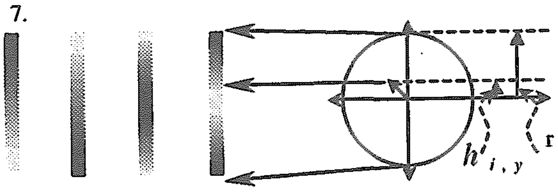


Figure 13 Four types of variation and the mapping from P_2 to h_i

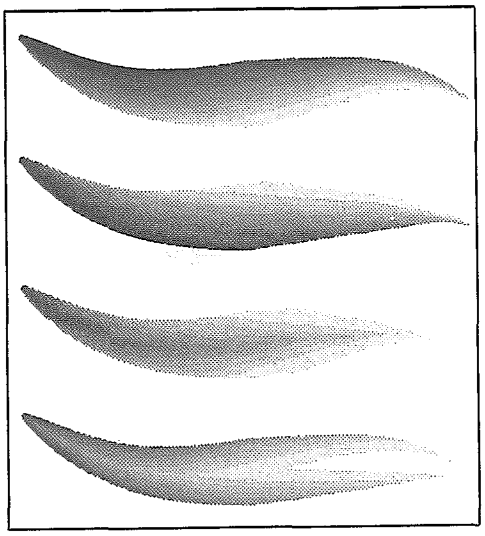


Figure 14. Four examples combines the effects of ink decreasing and ink soaking.variation

The gradient of the strokes in Figure 14 is too uniformly smooth for lack of individual bristle's difference. Because of the difference of each bristle, the strok es' may look roughly and have some discontinuity. Next two sections describe the local effects to individual bristles.

3.3.3. The Bristle Fan-Out Effect

This effect is simulated by a *difference parameter*. This parameter gives a random value to each bristle to generate the effect due to the fanning out of bristles. In order to give randomness to each piece of bristles, we define a difference array as follows

Difference [100] = $\{d_0, d_1, d_2, \ldots\ldots, d_{99}\}$

where $-5 \le d_z \le 5$, for $0 \le z \le 99$, is a random number. Each bristle has a corresponding d_z in *Difference[]*. Every time we decide the amount of ink deposited by a bristle, the corresponding d_z is added to generate random effect of the bristle. The mappings from bristles to *Difference[* are calculated by the ratio of $h_{i,y}$ and r. As shown in Figure 17.

$$\frac{|h_{i,y}|}{r} = \frac{z}{range} \tag{8}$$

where r is the current radius of the stroke and z is the index of d_z, $h_{i,y}$ is the y value of bristle h_i before rotation, 0<*range*<100 is the segment in *difference[]* that could be mapped to.

By changing the value of *range*, we can map the brush width into different length of segment in *difference[]*. When *range=0*, it means all bristles on the circle map ping to *difference[0]*. In other cases, each piece of bristles gets a random number adding to its gray value and generates the non-uniform effect of stroke. Before a stroke is painted, we can assign new values into *difference*[] by a random function. So each stroke is different from the other. The ink deposition can be represented as

$$\textit{ink deposited} = W_{P_1} P_1 + W_{P_2} P_2 + W_{diff} d_z \tag{9}$$

where P_1, P_2, d_z are ink decreasing , soaking variation, and fan-out parameters, W_G, W_V, W_{diff} are the weights of each item. Figure 15 shows the results of different W_{diff} and *range*.

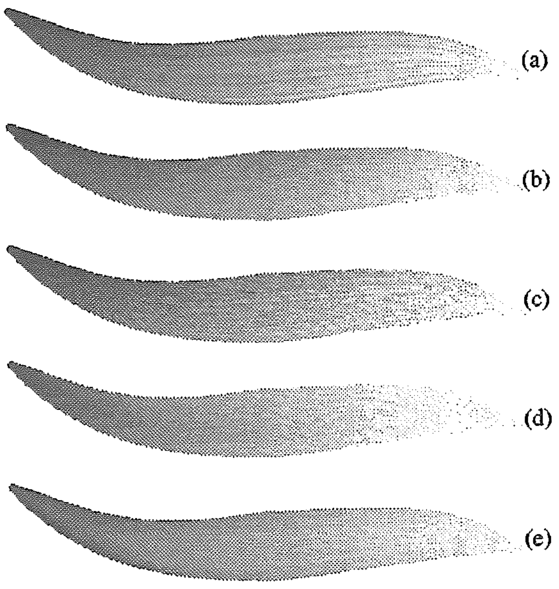


Figure 15 Results of equation (9) with varied parameters.

(a) *range*=10, W_{diff} =5, P_2 type =0

(b) *range*=90, W_{diff} =3, P_2 type =0

(c) *range*=100, W_{diff} =15, P_2 type =

(d) *range*=50, W_{diff} =3, P_2 type =2

(e) *range*=50, W_{diff} =3, P_2 type =3

3.3.4. The Dry-Out Effect

During the painting process, bristles may out of ink and leave discontinuous gaps on its track. This happens when there is no enough ink remaining on the brush. If the ink deposited by a bristle is less than a threshold value, we may generate a gap of random size in the track. For this purpose, we define a discontinuity arra *discontinuity[100]*. Each element of *discontinuity[* contains a distinct gap size. Users can specify an upper bound of gap size as the value "Max _gap" in the pseudo-code to limit the size of gaps.
Up to now, we have discussed all parameters of ink deposition. We can put them together as follows:

$$Ink\ Deposited = (P_1 + P_2 + difference\ [\frac{|h_{i,y}| * range}{r}]) * (discontinu\ ity\ [\frac{|h_{i,y}| * range}{r}]) \quad (10)$$

Figure 16 shows the results after applying the dry-off effect.

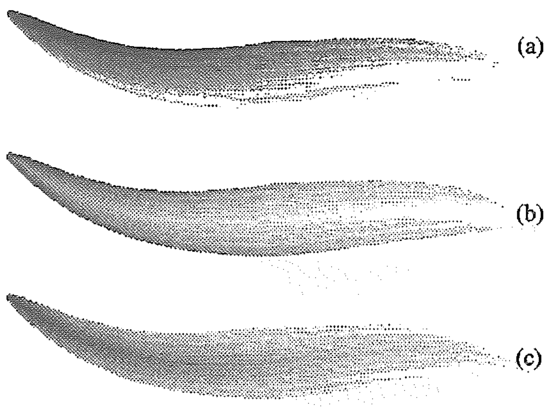
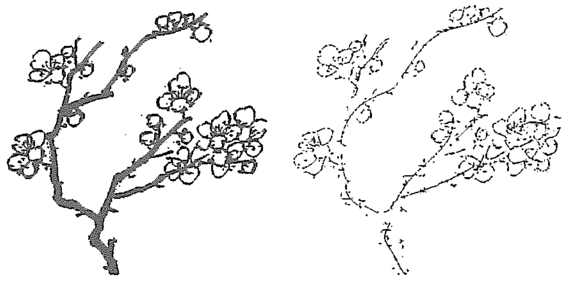


Figure 16 Results of discontinuity effect.
(a) Active threshold = 170, gap_Max = 30, P_2 type = 0.
(a) Active threshold = 230, gap_Max = 30, P_2 type = 2.
(c) Active threshold = 235, gap_Max = 30, P_2 type = 3.

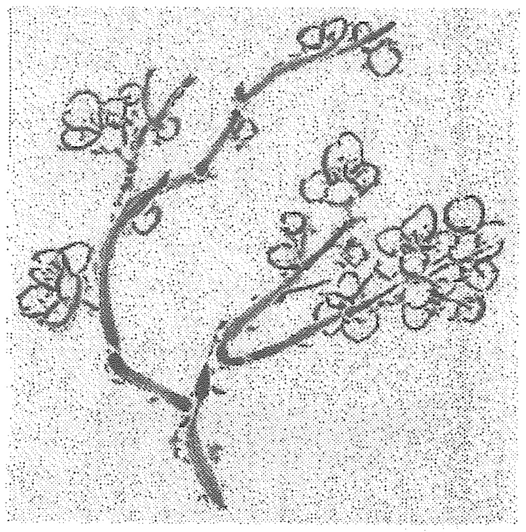
4 Results

In this section, the implementation and results are discussed The input sources are gray-level images obtained b scanning the paintings from Chinese ink paintin exemplification [1,2]. The algorithms are implemented in language on SGI O_2 workstation with an R5000 CPU and 192 MB RAM.
Figure 17 (a) is the source image of a plum blossom. Figure 17 (b) shows the skeleton of Figure 17 (a) and Table 2 shows the parameters of this example. Figure 21 (a) and (b) are the resulting images of different styles. Figure 22 and Figure 24 are examples of a bamboo image and a stone image. Figure 28 is an example of a 1250 x 830 image.

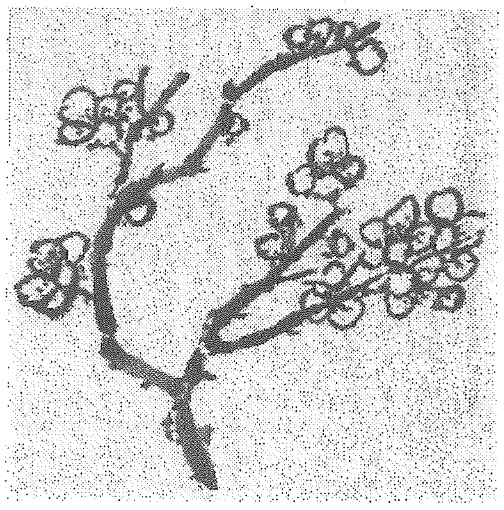


(a) Source image. (b) Skeleton of (a).

Figure 17 Example 1.

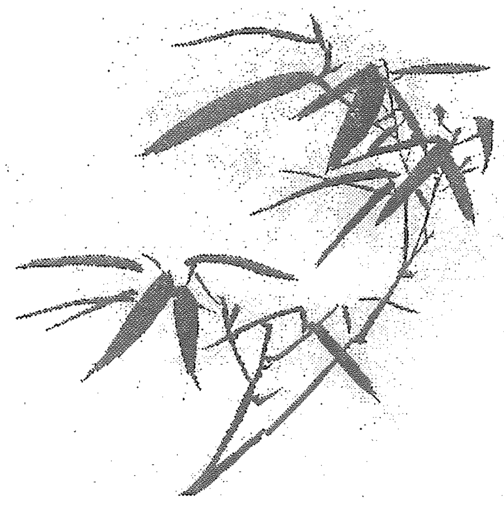


18 (a)



18 (b)

Figure 18 Resulting images of example 1.


19 (a) Source image.

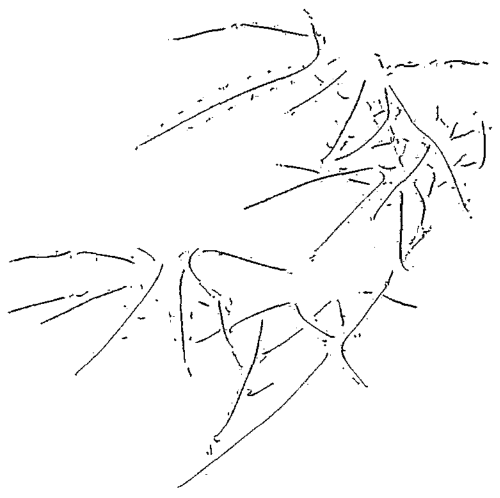
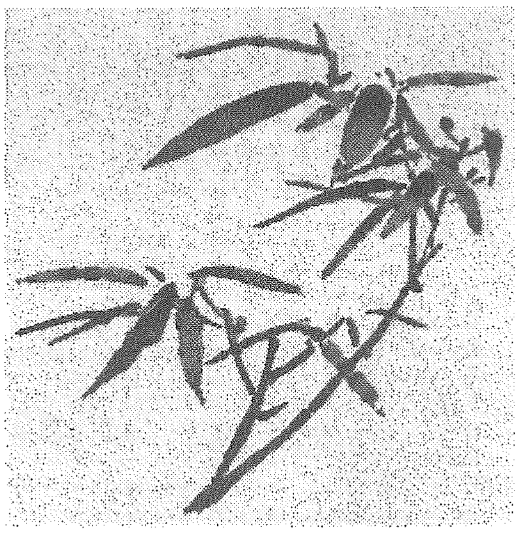
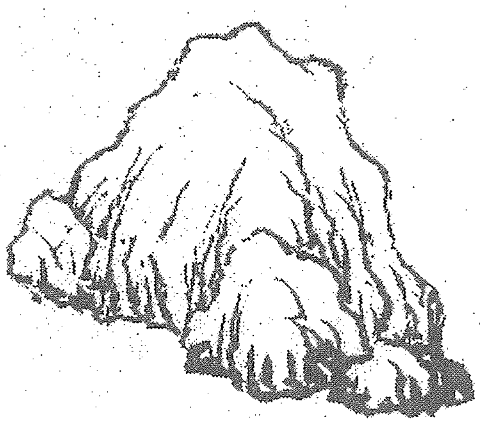

19 (b) Skeleton of 24 (a).

Figure 19 Example 2.


20 (a)


21 (a) Source image.

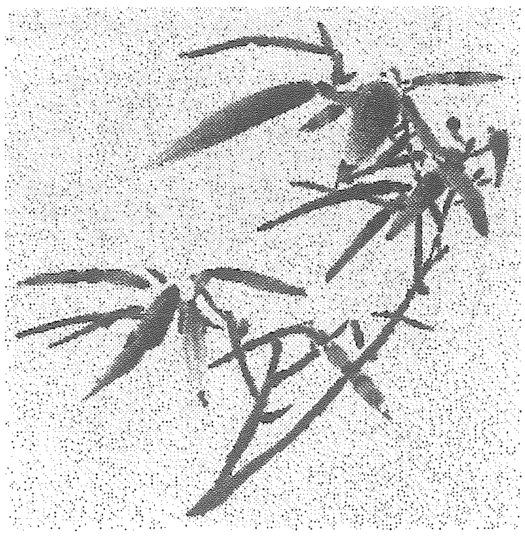

23 (b)

Figure 20 Resulting images of example 2.

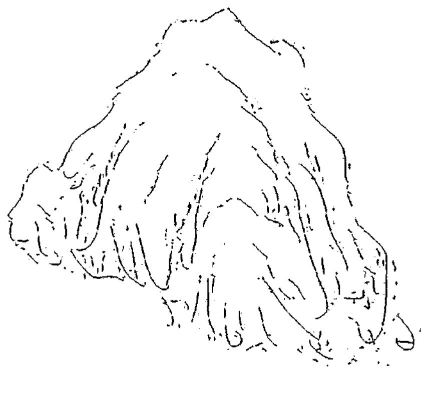
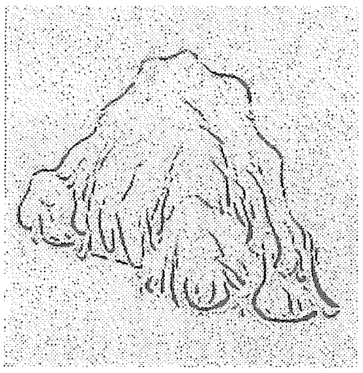
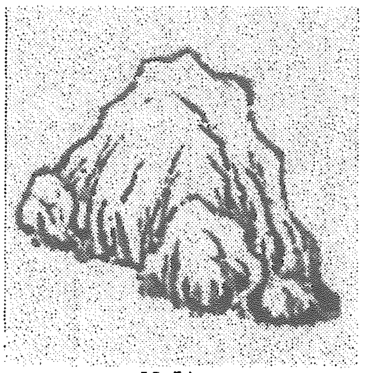

21 (b) Skeleton of 21 (a).

Figure 21 Example 3.



22 (a)



22 (b)

Figure 22 Results of example 3.

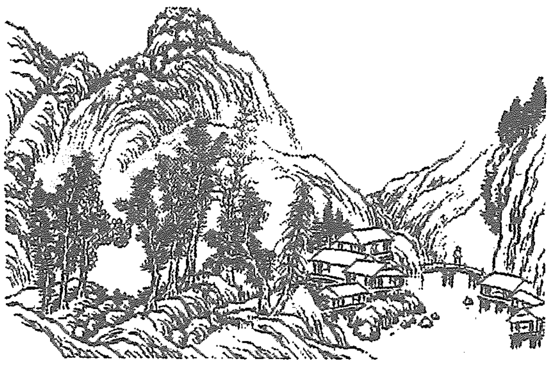


Figure 23 Source image of example 4.

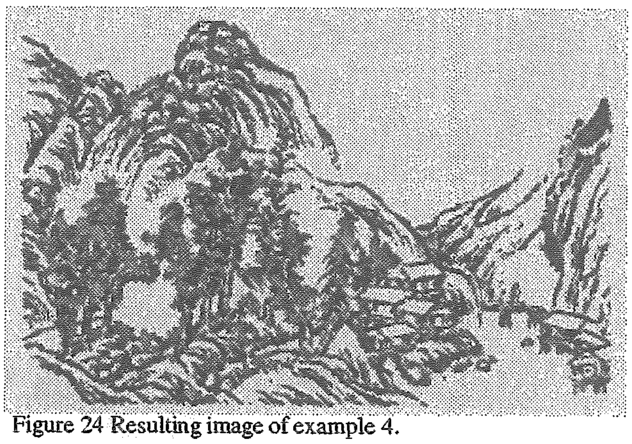


Figure 24 Resulting image of example 4.

5 Conclusions

In this paper, we propose a method to simulate the Chinese ink painting in different styles. The whole process is automatic. Users just specify the style parameters at the beginning to control the painting styles and then leave the rest to computer. Moreover, the algorithm is based on a physically-based brush model. The simulation of bristles can generate very realistic strokes. Users can obtain different results in a short time.

Reference

[1] 《芥子園畫譜》，華正書局印行。

[2] 陳耀庭 編，《山水畫法 1.2.3》，雄獅美術印行，民國七十五年出版。

[3] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, David H. Salesin, "Computer-Generated Watercolor," *Proceedings of ACM SIGGRAPH 97*, 1997.

[4] Qinglian GUO "Generating Realistic Calligraphy Words," *IEICETransactions on Fundamentals of Electronics Communications and Computer Sciences*, E78A(11):1556-1558, November 1996.

[5] Aaron Hertzmann, "Painterly Rendering with Curved Brush Strokes of Multiple Sizes," *Proceedings of ACM SIGGRAPH 98*, page 453-460, 1998.

[6] Horace H S Ip, Helena T F Wong, "Calligraphic Character Synthesis Using a Brush Model," *Proceedings of Computer Graphics International 1997*, page 13-21, 1997.

[7] John Lansdown, Simon Schofield, "Expressive Rendering: A Review of Nonphotorealistic Techniques," *IEEE Computer Graphics and Applications*, May 1995.

[8] Jintae Lee, "Simulating Oriental Black-Ink Painting," *IEEE Computer Graphics and Applications*, May/June 1999.

[9] Peter Litwinowicz, "Processing Images and Video for An Impressionist Effect," *Proceedings of ACM SIGGRAPH 97*, 1997.

[10] Barbara J. Meier, "Painterly Rendering for Animation," *Proceedings of ACM SIGGRAPH 96*, page 477-484, 1996.

[11] Michael P. Salisbury, Michael T. Wong, John F. Hughes, David H. Salesin, " Orientable Textures for Image-Based Pen-and-Ink Illustration," *Proceedings of ACM SIGGRAPH 97*, 1997.

[12] Steve Strassmann, "Hairy Brushes," *Proceedings of ACM SIGGRAPH 86*, page 225-232, 1986