Non-Photorealistic Rendering in Chinese Painting of Animals

Jun-Wei Yeh, Ming Ouhyoung

Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, 104, Taiwan.

E-Mail: tony@cmlab.csie.ntu.edu.tw, ming@csie.ntu.edu.tw

Figure 1 Results of automatically rendering 3D animal models to Chinese painting style and at the right top corner is the original 3D animal model.

Abstract A set of algorithms is proposed in this paper to automatically transform 3D animal models to Chinese painting style. Inspired by real painting process in Chinese painting of animals, we divide the whole rendering process into two parts: borderline stroke making and interior shading. In borderline stroke making process we first find 3D model silhouettes in real-time depending on the viewing direction of a user. After retrieving silhouette information from all model edges, a stroke linking mechanism is applied to link these independent edges into a long stroke. Finally we grow a plain thin silhouette line to a stylus stroke with various widths at each control point and a 2D brush model is combined with it to simulate a Chinese painting stroke. In the interior shading pipeline, three stages are used to convert a Gouraud-shading image to a Chinese painting style image: color quantization, ink diffusion and box filtering. The color quantization stage assigns all pixels in an image into four color levels and each level represents a color layer in a Chinese painting. Ink diffusion stage is used to transfer inks and water between different levels and to grow areas in an irregular way. The box filtering stage blurs sharp borders between different levels to embellish the appearance of final interior shading image. In addition to automatic rendering, an interactive Chinese painting system which is equipped with friendly input devices can be also combined to generate more artistic Chinese painting images manually.

Keywords Chinese painting of animals, non-photorealistic rendering, color quantization, ink diffusion.

1 Introduction

Non-photorealistic rendering (NPR) is a newly developed rendering technique in recent years. The goal of NPR is not rendering objects realistically but more creatively such as in simulating various painting styles.

Curtis et al. [1] simulates the physical property of media and substrate in watercolor. But his work only focused on 2D image processing and does not use the information of 3D model in 3D scene rendering. There is another important artistic style in NPR called “pen-and-ink” [2, 3, 4, 5, 6] illustration. Most of these researches focus on rendering 3D models with pen-and-ink style depending on its parametric properties. Another research

related to pen-and-ink is pencil rendering [7, 8]. Praun et al. [9] applies a pencil hatching style to 3D models. The contribution of his work is to avoid discontinuity when changing color tones of textures. The idea “tonal-maps” is inspired from the paper “Non-photorealistic Virtual Environments” by Klein et al. [10]. Other researches in NPR are about finding silhouettes in 3D models quickly and applying various line styles to silhouettes. Markosian et al. [12] proposed a new algorithm that can detect visible lines and surfaces in real-time from 3D model, and it needs not travel all edges in the model to find the silhouette lines by his special walking mechanism. As the result of his paper, he applied several styles of strokes to the silhouette edges on 3D models. Northrup et al. [11] used a hybrid method to draw silhouettes. First they used image-based approach to find visible line paths and these paths have the resolution of object-space edges. And their algorithm can also solve the zigzag line problem. The final result can render at interactive rates.

We can easily find that in previous researches, almost all NPR systems simulated the western painting arts, such like watercolor, pen-and-ink, hatching and oil painting, etc. It is a pity that few researches in NPR field are about traditional Chinese painting except these two years. So we decide to develop a Chinese painting rendering system in this paper.

Our automatic rendering process in Chinese painting style aims on 3D animal models, and it is a new topic in NPR. Lee [13] developed an oriental black-ink painting system and this system only provided an interactive painting environment to help a user paints more easily with various brush tools and ink effects. The system focuses on the freedom of a user’s imagination and creativity. Way and Shih [14] presented a method of synthesizing rock textures in Chinese landscape painting, but users have to first specify the contour of mountain then complete the painting process, or users can paint interactively by using their system. Way et al. [15] developed another application to simulate the Chinese painting of trees by using silhouette and texture strokes. Unlike their previous system, they began to use 3D tree models to be their source models, and established four reference maps to analyze the information for the bark texture. They can draw various styles of bark texture by defining the texture patterns.

We intend to develop an automatic rendering system that includes the whole process of making a Chinese painting of animals. First, outlines are extracted, and then interior shading by diffusion is done automatically.

2 Finding 3D Model Silhouette

2.1 Definition of Silhouette

Markosian and Kowalski [12] generalized the definition of silhouette for polygon models: A silhouette edge is an edge adjacent to one front-facing and one back-facing polygon. A polygon is defined as front-facing if the dot product of its outward normal and a vector from the camera position to a point on the polygon is negative. Otherwise the polygon is back-facing.

2.2 Algorithm

Raskar and Cohen [16] proposed three hardware-supported methods to find silhouette edges. “Hardware-supported” means they use OpenGL culling functions for speeding up their algorithms. However, depending on hardware resources will lose much power of controlling these silhouettes. For example we cannot apply various width or textures on a long linked silhouette line.

Considering the situation above, we chose the original software implementation that is to test every edge explicitly in the polygon model while rendering. We found it does not cause any system delay by implementing the original algorithm on a Pentium III 500Hz machine. As a result, we can obtain silhouette information in the whole model at any specified viewpoint.

2.3 Silhouette Culling

Consider figure 2, silhouette edge at the right side is also adjacent to a front-facing and a back-facing polygon, but unfortunately it is occluded by faces at left side. If we do not eliminate this illegal edge by ourselves, it will be identified as a silhouette.

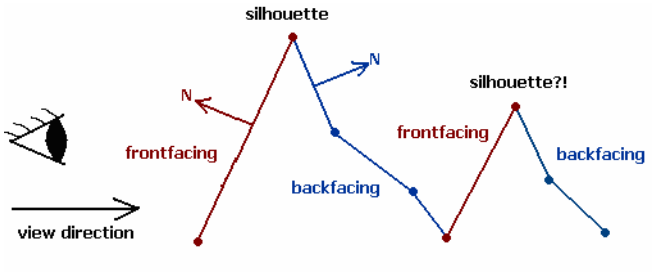

Figure 2 By the definition of silhouette, we can find two silhouette edges in the example but silhouette edge at the right side is invisible because it is occluded by faces at the left side.

We convert 3D-coordinate of both endpoints in all silhouette edges into 2D window coordinate and a depth value. By using its 2D window coordinate we can compare its depth value with system depth buffer and three cases will meet during this comparing process:

1. Both endpoints pass the depth buffer.
2. Both endpoints do not pass the depth buffer.
3. One endpoint passes the depth buffer but the other doesn't.

It is simple to handle case 1 and case 2; we keep and delete both two endpoints respectively. But in case 3 we want to preserve its visible part of whole edge. Thus we cut this edge into two halves and a new endpoint will be generated at middle; again we check this new endpoint and another visible endpoint to see whether both of them can pass the depth buffer. If not, repeat this operation until they can satisfy case 1.

3 Stroke Mechanisms

3.1 Stroke Linking Mechanism

In our definition, a stroke must be composed of a group of edge segments that are connected and both endpoints in an edge segment are control points in a stroke. However these silhouette edges extracted from a 3D model are stored as a segment type and there is no correlation among them.

We store all silhouette edges in a waiting pool, then take one edge out which has not been classified to any group number to be an initial edge and push its 2D-coordinate of both endpoints into top and bottom of link list simultaneously. Next we explicitly search silhouette edges in waiting pool to check whether this edge is connected to the impermanent stroke in link list.

Once the current searching edge is identified to be connected with the impermanent stroke in link list, we push another endpoint of this edge into control point list from top or bottom direction by different conditions. When a new control point is inserted into link list, we search for all silhouette edges again from first index in

waiting pool because top or bottom element of link list has been changed. After every silhouette in waiting pool has been searched, it means we have inserted all control points that are connected together, and all edges in link list will have the same group number to identify that they belong to the same stroke. We repeat this process until all silhouette edges have been classified to a group number.

3.2 Stroke Making Mechanism

Without any stroke making mechanism, it is only a long linked line and has no style and width. In order to grow various widths at every control point, we make some modifications to Strassmann's [18] method because we already have positions of control points in a stroke. Observing Chinese painting strokes, a stroke always has its both ends thin but fat in the middle. Depending on this principle of Chinese painting we can automatically assign pressure value at each control point. We also make an assumption that when user put more pressure on a Chinese pen brush, a bigger stroke width will be generated on the paper.

We set two parameters for initialization and limitation of stroke width calculation: ***width_step*** and ***MAX_WIDTH***. The former defines an increment of stroke width from the last control point and the later defines the maximum stroke width value of this stroke. We calculate stroke widths according to following equations:

$$add[i] = \begin{cases} +width\_step, & if \quad i < \frac{1}{2} \times ncp \\ -width\_step, & if \quad i \geq \frac{1}{2} \times ncp \\ 0, & if \quad width[i] \geq MAX\_WIDTH \end{cases} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$width[i] = \begin{cases} 0, & if \quad i = 0, i = ncp \\ add[i] + width[i-1], & else \end{cases} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where ***i*** is the index of control point of this stroke, ***ncp*** is the number of control points, ***add[i]*** is an 1D array which defines an increment of stroke width in this control point, and ***width[i]*** is another 1D array which defines stroke width in this control point.

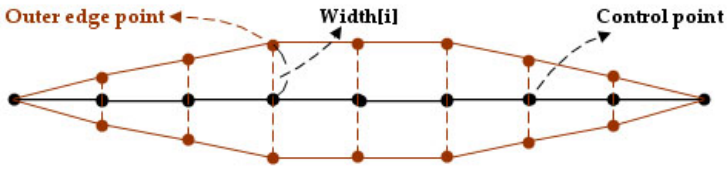

Figure 3 We grow each control point with a ***width[i]*** and form these outer edge points to be outline of a stroke. In order to fill the area inside a stroke, we draw a series of quadrilaterals representing the main body of a stroke, and two triangles representing the beginning and end of a stroke.

Outer edge points can be calculated from a control point and ***width[i]*** related to it. These outer edge points then form a stroke outline as shown in figure 3. The fastest way to fill the cover area inside a stroke is drawing a series of quadrilaterals representing the main body of a stroke, and two triangles representing the beginning and end of a stroke. Results rendered from 3D animal models with stroke mechanisms are shown in figure 4.

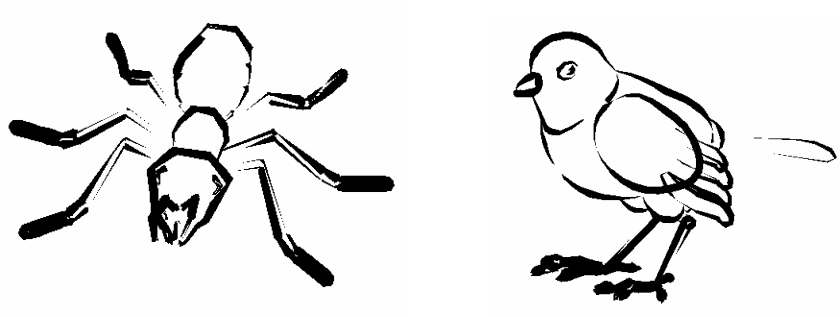

Figure 4 Rendering results from an ant model and a sparrow model. We use OpenGL drawing primitive functions to speedup this process.

4 From a Brush Model to Strokes

Referring to Way and Shih's [14] paper, a 2D brush model is used in our system to simulate the contact region on which a Chinese pen brush touches a canvas. As we mentioned in section 3.2, a stroke is composed of several control points and each of them has different stroke widths. Since we have the ability to rotate, translate and scale our brush model, it also means we can fit our brush model into any stroke even they have different widths and different rotate angles. When moving our brush model between two control points, it goes one pixel at a time. Simultaneously, brush model size and its position will be interpolated in a moving process.

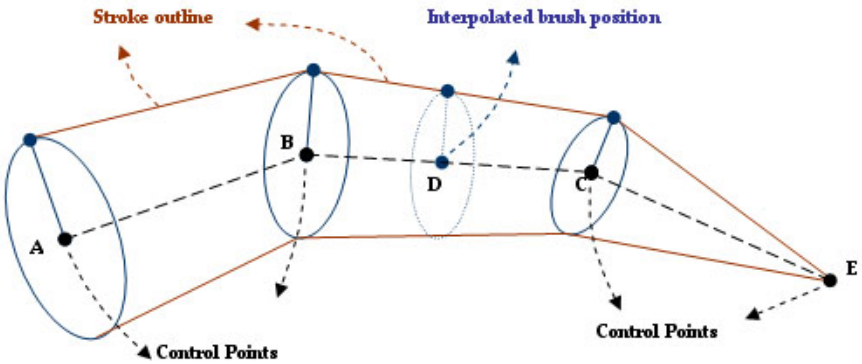

Figure 5 Fitting brush model into stroke. We locate brush model at the control point of stroke, and then we interpolate position and size of brush model between two control points.

Thus a brush model can be kept moving, scaling and rotating when it travels along the moving trajectory. Each time it scales its long axis to fit the stroke width, we have to re-calculate the ellipse equation for retrieving the position of each bristle inside the brush model.

Smaller brush model size will not decrease the number of bristles because it only shortens the distance between bristles on the same radius. Similarly, when using the Chinese pen brush, same numbers of bristle touch the paper even if a smaller area is painted. Therefore, the amount of calculation work is independent to the size of a brush model. We have to do this calculation every few steps because the stroke width of each control point is different. Unfortunately it causes a large overload in our system. By designing several patterns of brush model for dynamic brush size may be a good solution to ease this overloading.

Finally we implement two special effects called dry brush effect and turning effect that referred to Way and Shih's paper [14] and Hsu's master thesis [19]. However, we did a little modification so that their algorithms can accommodate to our mechanism. For example we have to determine the value ***P*** (brush pressure) at each step when brush model is moving on a stroke. We decide to assign larger ***P*** at control point whose included angle of adjacent two edges is smaller. And we also interpolate this pressure value between two control points to get a smooth stroke line.

The following result images in figure 6 are rendered after applying our simulated brush model on strokes with both dry brush effect and turning effects.

Figure 6 Result images of applying brush model into strokes.

5 Interior Shading

Although stylus strokes along silhouettes of 3D animal model are generated, the interior area is still kept blank. An interior shading pipeline is needed to simulate the real painting process in Chinese painting of animals. Three stages are proposed in our interior shading pipeline: color quantization, ink diffusion and box filtering as shown in figure 7.

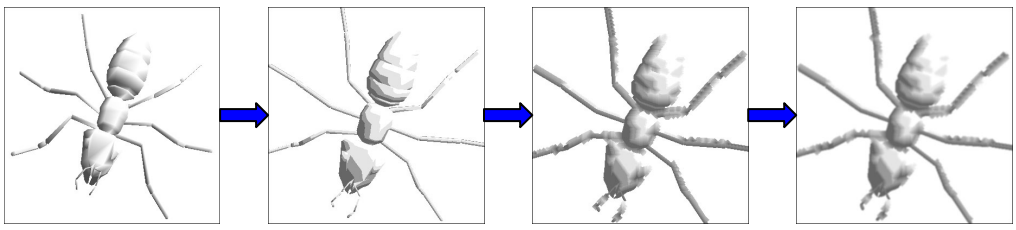

Figure 7 Interior shading pipeline

In order to get pixel color information in the interior area, we render 3D animal models with general Gouraud-shading. However, we increase parameters of diffuse lighting and specular lighting whereas decreasing the parameter of ambient lighting. The purpose is to create a large contrast in the whole image so that the color distribution range is enlarged and the color distribution histogram is equalized.

We quantize all pixels in the interior area into four color levels. Color ranging from pure black to pure white is distributed from 0.0 to 1.0. From the color distribution histogram as shown in the top of figure 8, we set three thresholds for color quantization: 0.4, 0.6, 0.9. A pixel located at any place in color distribution histogram will be quantized to a relative threshold color value. The color mapping table is also shown in the bottom of figure 8.

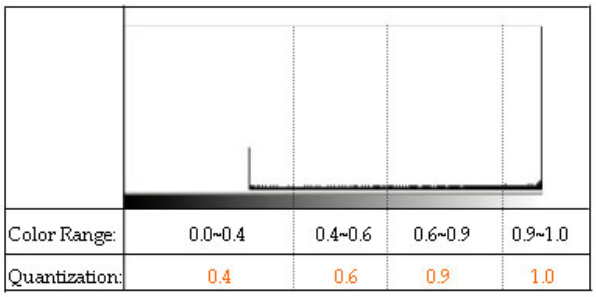

Figure 8 Three thresholds in color distribution histogram are defined as: 0.4, 0.6, 0.9, and each pixel color is assigned to a threshold value according to their location in color distribution histogram.

From the interior shading result of a real Chinese painting, we find that every time when artists add a darker color to an existing color level, the darker level will blend with others at borders and its ink will diffuse to other levels in an outward direction. We need to simulate this phenomenon in our interior shading pipeline because it is also an important characteristic in Chinese painting

We first define two parameters: ink quantity and water quantity. Ink quantity is used to transfer inks between different levels whereas water quantity determines whether this pixel should transfer inks to its neighbors. Neighbors are defined as four adjacent pixels related to the current pixel. Ink quantity contained in one pixel can be calculated according to its color value:

$$ink = 255 \times (1 - color) \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

where color value is ranging from 0.0 to 1.0. If the color value of one pixel is 1.0, it means pure white and its ink quantity is zero. We also define four levels according to the quantization levels mentioned above; they are dark level, medium level, light level and paper level; the ink quantities are 153, 102, 25.5 and 0, respectively.

By observing the ink diffusion result from figure 7, we can still find sharp borders between different color levels. Therefore we apply a 5x5 box filter to each pixel of whole image to blur sharp borders.

Instead of using our interior shading algorithms, Lien [20] developed another interactive Chinese painting system that takes a tablet and a pen as its input devices. The tablet can detect current position and pressure from pen during the painting process. Bringing different pressure on a pen will change different hues of brush on a paper. By using this interactive painting system, a user can easily complete the interior shading process and create an interesting artwork.

6 Conclusion and Future Works

A simplified algorithm to make silhouette strokes automatically from 3D models is proposed, and the rendering system is enhanced with the capability of automatic interior shading. Special effects on brush model consist of dry

brush effect and turning effect and more are needed such as the ink soaking effect, ink decreasing effect and back-runs. We can also add several brush model patterns in our system to let users choose their favorite one and by applying different brush models it will greatly change the style of rendering results.

7 Results

As shown at the top of this paper, those two result images are rendered automatically from 3D animal models to Chinese painting style images. The following figure shows more rendering results of using various 3D animal models. Interior shading of figure 10 is painted by Lien's interactive Chinese painting system that uses 2D brush silhouette generated from our system as its input image.

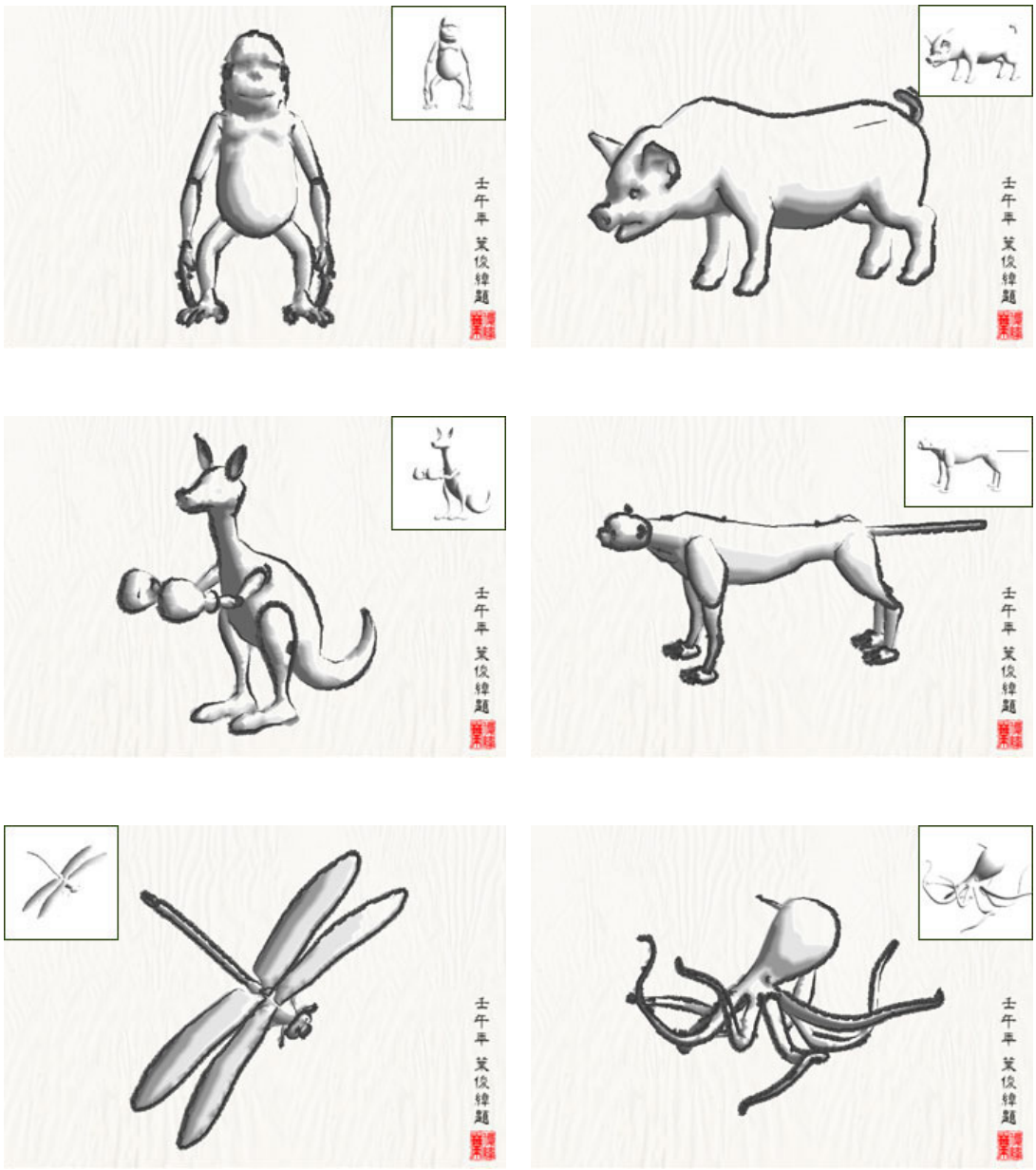

Figure 9 These result images are generated automatically from 3D animal models to Chinese painting style and image at right top corner is its original 3D model.

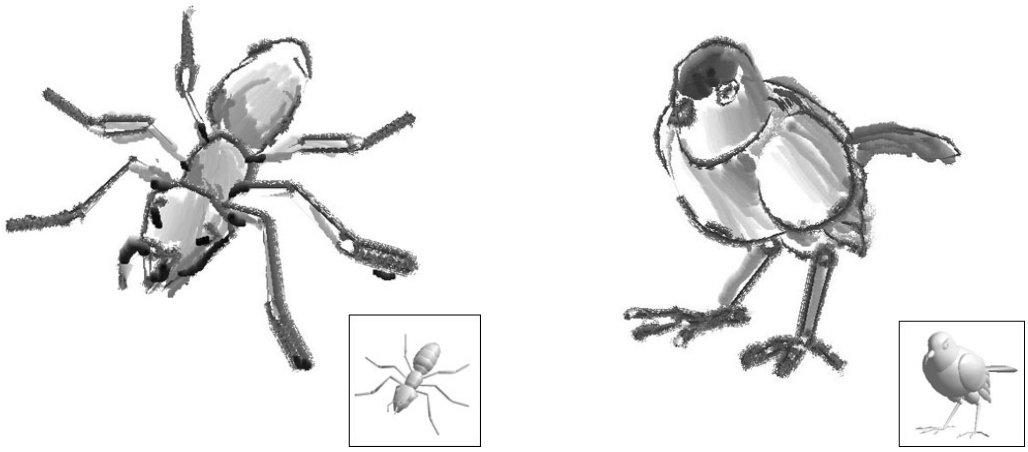

Figure 10 Final results after using Lien's interactive painting system to add interior strokes from figure 6 and the original 3D model is at the right bottom corner of image.

7 References

[1] CURTIS, C. J., ANDERSON, S. E., SEIMS, J. E., FLEISCHER, K. W., AND SALESIN, D.H. "Computer–generated watercolor." In Proceedings of SIGGRAPH 97, Computer Graphics Proceedings, pp.421-430, 1997.

[2] SALISBURY, M. P., ANDERSON, S. E., BARZEL, R., AND SALESIN, D. H. "Interactive pen-and-ink illustration." In Proceedings of SIGGRAPH 94, Computer Graphics Proceedings, pp.101-108, 1994.

[3] WINKENBACH, G., AND SALESIN, D. H. "Computer-generated pen- and-ink illustration." In Proceedings of SIGGRAPH 94, Computer Graphics Proceedings pp.91-100.

[4] WINKENBACH, G., AND SALESIN, D. H. "Rendering Parametric Surfaces in Pen and Ink." In Proceeding of SIGGRAPH 96, Computer Graphics Proceedings pp.469-476.

[5] SALISBURY, M. P., WONG, M.T., HUGHES, J. F., AND SALESIN, D.H. "Orientable textures for image-based pen-and-ink illustration." In Proceedings of SIGGRAPH 97, Computer Graphics Proceedings, pp.401-406, 1997.

[6] SALISBURY, M., ANDERSON, C., LISCHINSKI, D., AND SALESIN, D. H. "Scale-dependent reproduction of pen-and-ink illustrations." In Proceedings of SIGGRAPH 96, Computer Graphics Proceedings, pp.461-468, 1996.

[7] SOUSA, M. C., AND BUCHANAN, J. W. "Observational model of blenders and erasers in computer-generated pencil rendering." In Proceeding of Graphics Interface 99, pp.157-166, 1999.

[8] SOUSA, M. C., AND BUCHANAN, J. W. "Computer-generated graphite pencil rendering of 3D polygonal models." Computer Graphics Forum 18(3): pp.195-208, September 1999.

[9] PRAUN, E., HOPPE, H., WEBB, M., FINKELSTEIN, A. "Real-Time Hatching" In Proceedings of SIGGRAPH 01, Computer Graphics Proceedings, pp.581-586, 2001.

[10] KLEIN, A. W., LI, W., KAZHDAN, M. M., CORREA, W. T., FINKELSTEIN, A., AND FUNKHOUSER, T. A. "Non-photorealistic Virtual Environments" In Proceedings of SIGGRAPH 00, Computer Graphics Proceedings, pp.527-534.

[11] NORTHRUP, J. D., MARKOSIAN, L. “Artistic Silhouettes: A Hybrid Approach” In Proceedings of NPAR 2000, pp.31-38.

[12] MARKOSIAN, L., KOWALSKI, M, A., TRYCHIN, S. J., BOURDEV, L. D.,GOLDSTEIN, D., HUGHES, J. F. “Real-Time Nonphotorealistic Rendering” In Proceedings of SIGGRAPH 97, Computer Graphics Proceedings, pp.415-420, 1997.

[13] LEE, J. “Simulating Oriental Black-Ink Painting.” IEEE Computer Graphics and Applications, May/June 1999 (Vol. 19, No. 3). pp. 74-81.

[14] WAY, D. L., AND SHIH, Z. C. “The Synthesis of Rock Texture in Chinese Landscape Painting.” COMPUTER GRAPHICS Forum Volume 20, Number 3, pp.C123-C131, 2001.

[15] WAY, D. L., LIN, Y. R. AND SHIH, Z. C. “The Synthesis of Trees Chinese Landscape Painting Using Silhouette and Texture Strokes.” Journal of WSCG Volume 10, Number 2, pp.499-506, 2002.

[16] RASKAR, R. AND COHEN, M. “Image Precision Silhouette Edges.” 1999 ACM Symposium on Interactive 3D Graphics, pp.135-140, April 1999.

[17] LI, C, M. “The Method of Painting Animals.” ART Book Co., Ltd, 1990.

[18] STRASSMANN, S. “Hairy Brushes.” Computer Graphics (Proc. SIGGRAPH 86) 20(4):225-232 (August 1986).

[19] Hsu, Chih. Wei. Master thesis “The Synthesis of Rock Textures in Chinese Landscape Painting” pp.47-48. Dept of CIS, National Chiao Tung University, 2000.

[20] Lien, Ting. Yu. Master thesis “A Chinese Painting System with real-time Brush Dynamics Simulation.” Dept of CSIE, National Taiwan University, 2002.