AI Supported Computer-Generated Pen-and-Ink Illustration

Yan Gu, Songhua Xu, Min Tang, Jinxiang Dong
CAD&CG State Key Lab
AI Institution of Zhejiang University
P.O.BOX 1549, Hangzhou 310027
guyan@cise.zju.edu.cn

Abstract

In the field of computer graphics there is an increasing demand for non-photorealistic effects. In the paper, we add the idea of pattern recognition guided by theoretical rules into previous traditional NPR system and create non- photorealistic drawings intelligently. The Style Sample Library functions as an expert library and makes it easier to render a picture with non-photorealistic effects even by an amateurish user.

1. Introduction

Recently, a variety of techniques have been proposed to non-photorealistic rendering objects. This work was driven by the impulsion that we wanted to convey necessary information but consuming less storage, so they can be easily reproduced and transmitted.

In this paper, we proposed a method for AI supported pen-and-ink illustration of CAD parts. The system recognized the drawings users created automatically and guided the users towards the patterns predefined in style sample library. Actually, this is an intellectual learning process.

1.1. Related work

Gooch [2] et al. proposed the conception of color illustration, with luminance and changes in hue to indicate surface orientation, reserving extreme lights and darks for edge lines and highlights. Deussen and Strothotte [1] applied depth discontinuities to determine what part of each drawing primitive was to be drawn to constitute the foliage. Winkenbach and Salesin [9] summarized the principles of traditional pen-and-ink illustration and introduced "stroke texture" in their paper. The Premisys Corporation [10] marketed a product called "Squiggle" that adds waviness and irregularities to CAD output as a post-process, lending a hand-drawn appearance to the drawings. In many applications, for example, when we want to describe a complex mechanical hardware, it is time costing to render it including extraneous details. With illustration we can accent on more important parts and achieve effective diagrams, for pen-and-ink illustration, it has well-established conventions. For this reason, we don't duplicate others' work simply.

The main contribution of our work is that we began with important cognitive principles of pen-and-ink illustration and created style sample library and recognized certain patterns based on the mathematical definition of pattern specification. Furthermore, with the guidance of system, we approximated to the predefined patterns.

1.2. Overview

The rest of this paper is organized as follows. Section 2 introduced the idea guiding the design of this intellectualized system and the building blocks of our system. Section 3 explores the traditional illustration principles with AI techniques and gives our synthetic algorithms applied with pattern recognition. Section 4 discusses some of our results and section 5 summarizes our work and looks into the future. Appendix gives details about implementation.

2. Implementation

The most creative investigation of this paper is to integrate AI techniques ingeniously, including specifying style parameters according to the principles of pen-and-ink illustration in pre-process and recognizing patterns with geometry taxonomy.

In pre-process, the system supports three types of stroke textures till now; they are "cross-hatching", "stippling", "glass". We got the idea from [3]. Based on these types of stroke textures, we extract five style parameters. They are L (stroke length), T (stroke thickness), W (waviness), O (stroke orientation), D (stroke distribution).

In our Style Sample Library, we give mathematical definition of these five style parameters to each style of

stroke textures. This is an artistic process, not a technical one. No abstract standards rule whether it is non-photorealist or not. We just create this Style Sample Library on our perspective and set it as criterion of latter steps.

The recognized style parameters of the user created drawings compare with the style parameters from Style Sample Library. Recognizing the difference, the system modifies user created drawings under the guidance of predefined modification rules.

The implementation is described in Fig. 1.

3. AI supported computer-generated pen-and-ink illustration

3.1. Style Sample Library

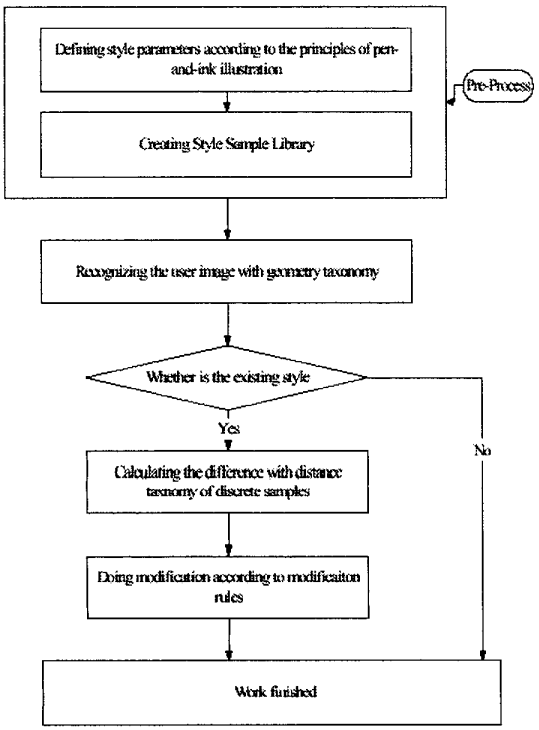

Figure 1. System implementation.

As Winkenbach and Salesin have mentioned in their paper, pen-and-ink illustration is a limited medium. The pen gives off no color or tone, so both color and shading must be suggested by combination of individual strokes. However, pen-and-ink illustration has some advantages such as each individual pen-and-ink stroke can be made expressive by employing small irregularities in its path and pressure, the small difference of stroke distribution can clearly indicate the difference between textures like smooth glass and cross-hatching.

We distill some principles from Winkenbach's paper.

The principles of stroke drawing in our implementation are summarized as below:

1. Even-weight line drawings appear lifeless; the thickness of a line should vary along its length.
2. Wavy lines are a good way to indicate that a drawing is schematic and not yet completely resolved.

The principle of stroke distribution in our implementation is summarized as below:

1. Absence of detail indicates glare.

The principles of outline in our implementation are summarized as below:

1. Crisp straight lines are good for hard objects, while a greater variety of lines quality is better for soft objects.
2. Using indication for drawing outlines is just as important as for drawing tones.

According the above theoretical rules, we extract five style parameters, L, T, W, O and D. And we define these five style parameters for each style sample.

For cross-hatching, we define that:

1. Using indication for drawing outline.
2. For each drawing, $\sum_{i=1}^{n} \frac{L_i}{n} > \alpha$ pixels.
3. Adding noise into stroke path, so W should be perturbed by a small random value.
4. The acute angle of two strokes θ must satisfy the inequality, $\sin\theta \neq 0$
5. The whole recognized area is B, detected area of no stroke is A, then $\frac{A}{B} < \beta$.

For stippling, we define that:

1. Using indication for drawing outline.
2. For each drawing, $\sum_{i=1}^{n} \frac{L_i}{n} < \gamma$ pixels.
3. The acute angle of two strokes θ must satisfy the inequality, $0 < \sin\theta < \lambda$.
4. The whole recognized area is B, detected area of no stroke is A, then $\frac{A}{B} < \beta$.

For glass, we define that:

1. Using outline detection.
2. The whole recognized area is B, detected area of no stroke is A, then $\frac{A}{B} > \beta$.

3.2. Pattern recognition

After transformation of one pattern, the pattern is mapped as a feature vector, also it is a point in feature space. In feature space, a point set belonged to pattern class ω_i can be separated from another point set

belonged to another pattern class ω_j in some degree. So using a division function, especially the linear division function independent of conditional distributing density, we can save a lot of calculating time. Out of above consideration, we employ geometry taxonomy in pattern recognition. The implementation is as follows:

We have a 5-dimension feature space, with feature parameters L, T, W, O, D, so we have to define a super-surface to distinguish the patterns.

In the implementation, we change one three-class problem to two two-class problems, and the judging interface of ω_i is adjacent to the judging interface of ω_j .

The judging function is

$g_k(X)=W_k^T X \quad k=1,2,3$ where X is a 3-dimensional feature vector.

The weight vector for judging function $g_i(X)$ is $W_i=(w_{i1},w_{i2},w_{i3})^T$.

The judging surface of ω_1 is

$$g_{12}(X)=g_1(X)-g_2(X)=0$$
$$g_{13}(X)=g_1(X)-g_3(X)=0;$$

The judging surface of ω_2 is

$$g_{21}(X)=g_2(X)-g_1(X)=0$$

The judging surface of is

$$g_{23}(X)=g_2(X)-g_3(X)=0$$

Suppose the unrecognized feature vector is $X=(1,1,1,1,1)^T$. Then substituting X in $g_1(X), g_2(X), g_3(X)$ can get the judgment result.

3.3. Similarity measurement

When the system recognizes the user created drawing, we can judge which pattern it belongs according to the quantitative feature vector, but the vector is a fuzzy area and is difficult to quantify because of distortion and noise. Furthermore, the samples are discrete, which is dependent on the quality of sample, predisposal and feature extraction. So if the system cannot recognize the pattern of user created drawing with geometry taxonomy, the distance d can be used to describe the difference. Considering the discrete samples, the distance $d(X,\omega_i)$ between unrecognized sample and class ω_i can be denoted as the average distance of all samples, that is

$$d^2(X,\omega_i)=\frac{1}{s}\sum_{t}^{s} d^2(X,Y_t).$$

And we adopt judgment rule, that is

$d^2(X,\omega_j) < d^2(X,\omega_i)\ \forall i \neq j \Rightarrow X \in \omega_j$

The smallest value of d is the pattern class that X belongs to.

3.4. Modification

According to the most approximate pattern distinguished by similarity measurement, we do some modification on user created drawing. This modification doesn't obliterate the user individuality, but an advanced reformation based on predefined rules as well. As matter as fact, the system provides an intelligent direction to user and helps user to learn the non- photorealistic rendering techniques. Step by step, user masters the styles in Style Sample Library with the help of the experts.

If the most approximate pattern is cross-hatching, then the system will modify the user created drawing with following rules:

1. Adding indication, the indication formula is expressed in appendix A1.
2. Supposing the length of style sample as Ls, the length of user created drawing as Lu, then we adjusted the final stroke length with La, given $La=\frac{Ls+Lu}{2}$.
3. Producing 2-dimension random function and moving each pixel along X, Y axes a random value.
4. Adjusting the acute angle of strokes θ so that θ satisfies the inequality $\theta > \arcsin\lambda$.
5. Supposing the percent of area filled with cross-hatching in style sample stroke is η, β is the percent of area filled with cross-hatching in user created drawing, then the area of $\eta-\beta$ must be filled with cross-hatching.

If the most approximate pattern is stippling, then the system will modify the user created drawing with following rules:

1. Rendering the drawing with stroke length as γ pixels.
2. Adjusting the acute angle of strokes θ so that θ satisfies the inequality $0<\theta<\arcsin\lambda$.
3. Supposing the percent of area filled with stippling in style sample stroke is η, β is the percent of area filled with stippling in user created drawing, then the area of $\eta-\beta$ must be filled with stippling.

If the most approximate pattern is glass, then the system will modify the user created drawing with following rules:

1. Detecting the outline with detail. There are many algorithms published on silhouette detection and extraction.
2. Emphasizing the highlight area. If the percent of highlight area in style sample η is smaller than the percent of highlight area in user created drawing β, then erase $\beta-\eta$ with the procedure Erase_Texture (Polygon3D).

4. Results

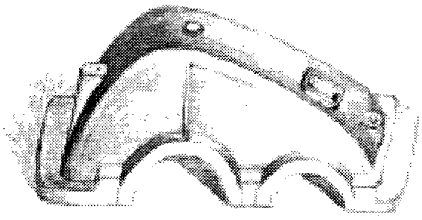

Figure 2. (a) CAD part mapping with cross-hatching texture.

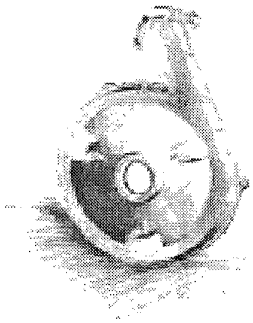

Figure 2. (b) CAD part mapping with cross-hatching texture.

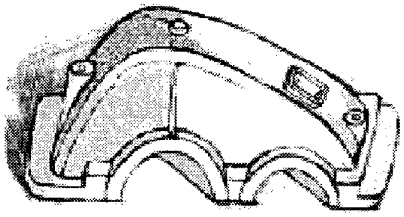

Figure 3. (a) CAD part mapping with stippling texture.

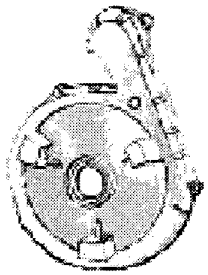

Figure 3. (b) CAD part mapping with stippling texture.

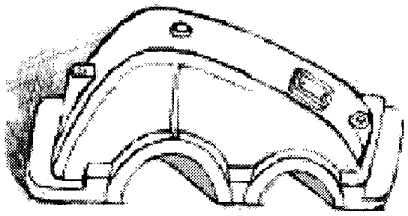

Figure 4. (a) CAD part mapping with glass texture.

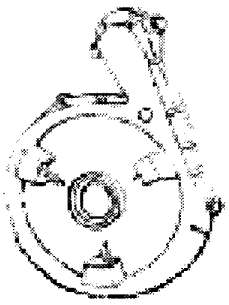

Figure 4. (b) CAD part mapping with glass texture.

5. Future work

This paper doesn't propose any improved algorithms for non-photorealistic rendering techniques except providing an integration of non-photorealistic techniques and AI techniques. The work described in this paper is just an early step in the exploration of pen-and-ink illustration system employed with AI techniques. There are many ways to extend this work, including:

1. Extracting more precise features from the non-photorealistic samples, creating more vivid drawings.
2. Recognizing the patterns with BP neural network, this will be a most challenge brainstorm.
3. Enduing the theoretical rules with self- studying, which means the rules can learn from the human-and-computer interaction dynamically.

6. Acknowledgements

We would like to thank Dai Lu for useful discussion during the early process of this paper and invaluable advice and support.

7. References

[1] Oliver Deussen, Thomas Strothotte, Computer- Generated Pen-and-Ink Illustration of Trees", *SIGGRAPH 2000 Proceedings*.

[2] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen,

"A non-photorealistic lighting model for automatic technical illustration", *SIGGRAPH '98 Proceedings.*

[3] Cassidy J. Curtis, Sean E. Anderson, Kurt W. Fleischer, and David H. Salesin, "Computer-generated watercolor", *SIGGRAPH '97 Proceedings.*

[4] M. Salisbury, M. Wong, J. F. Hughes, and D. Salesin, "Orientable textures for image-based pen-and-ink illustration", *SIGGRAPH '97 Proceedings.*

[5] J. Schumann, T. Strothotte, A. Raab, and S. Laser, "Assessing the effect of non-photorealistic images in computer aided design", *ACM Human Factors in Computing Systems, SIGCHI 96: 35–41*, April 1996.

[6] G. Winkenbach and D. Salesin, "Rendering parametric surfaces in pen and ink", *SIGGRAPH '96 Proceedings.*

[7] Adam Finkelstein and David H. Salesin, "Multiresolution curves", *SIGGRAPH '94 Proceedings.*

[8] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin, "Interactive pen-and- ink illustration", *SIGGRAPH '94 Proceedings.*

[9] G. Winkenbach and D. Salesin, "Computer- generated pen-and-ink illustration", *SIGGRAPH '94 Proceedings.*

[10] The Premisys Corporation, Chicago. Squiggle, 1993.

[11] Debra Dooley and Michael F. Cohen, "Automatic illustration of 3D geometric models: Lines", *Computer Graphics, 24(2): 77–82*, March 1990.

[12] Debra Dooley and Michael F. Cohen, "Automatic illustration of 3D geometric models: Surfaces", *proceedings of Visualization'90:307–314*, October 1990.

Appendix

Implementation Detail: Indication

Indication is one of the most notoriously difficult techniques for us to master, but it is also a most important techniques to indicate enough details. It requires putting just enough detail in just the right places, and also fading the detail out into the unornamented parts of the surface in a subtle and unobtrusive way. We referenced the formula of [3], and made some extend of it. A field w (x, y) is generated by the detail segment s at a point (x, y) in texture space according to

$$W(x,y) = (a + b \times dis\tan ce((x,y),s))^{-c}$$

where a , b , and c are non-negative constants that can be used to change the effect of the field; can be educed from the above similarity measurement function, s is the detected detail segment of outline. When several detail segments are present, we define the field at a point (x, y) to be that of the closest segment. So as not to create patterns that are too regular, the field w (x, y) is perturbed by a small random value. We created random value with Gauss Distribution. Textures such as "cross-hatching" and "stippling" evaluate the strength of the field of indication at the outline. The set of strokes for that element is generated only if the indication field is above some preset threshold.