Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

# Technical illustration based on 3D CSG models*

GENG Wei-dong (耿卫东)†, DING Lei (丁 磊), YU Hong-feng (余宏锋), PAN Yun-he (潘云鹤)†

(*State Key Lab of CAD & CG, School of Computer Science, Zhejiang University, Hangzhou 310027, China*)

†E-mail: gengwd@zju.edu.cn; panyh@zju.edu.cn

Received June 17, 2003; revision accepted Aug. 15, 2004

**Abstract**: This paper presents an automatic non-photorealistic rendering approach to generating technical illustration from 3D models. It first decomposes the 3D object into a set of CSG primitives, and then performs the hidden surface removal based on the prioritized list, in which the rendition order of CSG primitives is sorted out by depth. Then, each primitive is illustrated by the pre-defined empirical lighting model, and the system mimics the stroke-drawing by user-specified style. In order to artistically and flexibly modulate the illumination, the empirical lighting model is defined by three major components: parameters of multi-level lighting intensities, parametric spatial occupations for each lighting level, and an interpolation method to calculate the lighting distribution over primitives. The stylized illustration is simulated by a grid-based method, in which we 'fill' the desirable pictorial units into the spatial occupation of CSG primitives, instead of "pixel-by-pixel" painting. This region-by-region shading facilitates the simulation of illustration styles.

**Key words:** Non-photorealistic rendering (NPR), Technical illustration, Expressive rendering, CAD
**doi:**10.1631/jzus.2005.A0469 **Document code:** A **CLC number:** TP391

Non-photorealistic rendering (NPR) is effective at conveying information, more expressive or more beautiful than its photorealistic counterparts (Lansdown and Schofield, 1995). Technical illustration is a special subset of NPR since both of them share human perception as common driving force. NPR aims at rendering 3D objects to convey engineering information, and is very popular in engineering drawing. Many researchers had investigated its application for technical illustration. Gooch *et al*.(1998) proposed a new lighting model for technical illustration in terms of warm and cool color, automating the technical illustration from 3D object; Geng *et al*.(2001) designed an architecture to simulate the technical illustration methods of human engineers. Satio and Takahashi (1990) proposed a G-buffer data structure and employed edge detectors to generate and add the shape silhouettes into the resultant image, enabling the user to have an easy understanding of the geometric properties of the 3D object. Dooley and Cohen (1990) implemented a system to generate the illustrations from 3D surface. Seligmann and Feiner (1991) simulated the emphasizing methods aiming at expressing the design and manipulation intent. Winkenbach and Salesin (1994; 1996) mimicked the pen-and-ink illustration techniques for 3D polygonal object and parametric surfaces. Buchanan and Sousa (2000) designed an edge-buffer data structure to automate the silhouette generation from 3D polygonal objects. Schumann *et al*.(1996) proposed the assessment criteria of technical illustration in CAD system. However, we have not found an effective method to easily automate the technical illustration from CAD objects, partly because most of existing work is still based on the principles and pipeline of photo-taking, ignoring the fact that the rendition process and techniques employed by human engineers are different from it. It may be easy to accomplish the technical illustration if we take into account human depiction

*Project supported by the National Natural Science Foundation of China (No. 60373032), and the Returnee Foundation of Education Ministry of China and Zhejiang Province

knowledge.

Regarding human depiction, Willats (1997) conducted many experiments to investigate it, and he summarized that human depiction framework consists of two systems: the denotation system (which is primitive to use) and the drawing system (where to put them). The denotation system maps 3D scene entities into corresponding picture primitives such as region, line or parts. The drawing system performs perspective, oblique and/or orthogonal projection that map 3D spatial relations into the 2D picture. Durand (2002) further extended Willats' framework, and proposed a 4-stage processing pipeline to render mechanical product:

1. Spatial mapping. It handles the spatial properties, and generates the corresponding spatial layout in 2D picture according to the viewpoint and projection type.

2. Pictorial units conversion. It converts 3D primitives (points, lines, surfaces, volumes) into 2D pictorial units (points, lines, regions).

3. Attributes assignment. It assigns visual properties such as color, texture, thickness, transparency, wiggleness, or orientation to the pictorial units.

4. Mark implementation. It is responsible for medium simulation (e.g. oil painting, pencil brush, watercolor, engraving), and mimics the physical strokes in traditional rendering, and realizes the visual effect of primitives placed at their spatial location.

By this pipeline, we proposed and implemented a technical illustration system for 3D CSG models. In Section 2, we will give a brief description of the CSG-based automatic illustration framework, and the two key issues, lighting models for CSG primitives and simulation of stylized illustration, are discussed in Sections 3 and 4 respectively. Finally, a brief discussion is given in Section 5.

## CSG-BASED ILLUSTRATION FRAMEWORK

The key points in implementing Durand's rendering pipeline are how to calculate the spatial mapping, how to generate the pictorial units, how to assign the attributes, and how to carry out the desirable visual effect in the resulting illustration. In our CSG-based approach, these issues are handled as follows:

1. Spatial layout calculation. We presume that 3D-scene can be decomposed into a set of 3D CSG models in 3D object space. The spatial layout in the image space is generated according to the visibility of each 3D CSG model. Spatial layout is represented as a hybrid data structure of G-buffer (Satio and Takahashi, 1990) and topological layout (Agrawala and Stolte, 2001).

2. Generation of pictorial units in image space. The user will specify illustration style for each CSG primitive, and the system automatically generates the mapping of pictorial units in image space by conventions of technical illustration for mechanical design and manufacture.

3. Attributes assignment. Based on the spatial layout, we calculate the visual attributes of pictorial units by the predefined lighting models for CSG primitives. The applicable attributes of pictorial unit, such as density of lines, line pattern, length of segments, variation of segments from baseline, contours and waviness, are all computed by the lighting and illustration conventions.

4. Stylized illustration. Its major task is to draw each region in the spatial layout using the strokes with desirable attributes. We employ the grid-based filling to accomplish this medium simulation, instead of pixel-based painting. It first decomposes the visible surface into suitable grids in terms of the silhouette shape of the projected primitive, and then for each grid, it fills pictorial units into the region according to the density of lighting distribution. Compared with pixel-based painting, this grid-based filling is much more flexible in simulating illustration styles, as the technical illustration is an abstraction of pictures features represented by regions, and the pixels usually focus more on the concrete details of a picture.

A diagram of our rendering pipeline is shown in Fig.1. However, computer depiction involves much feedback and influence from the picture space to the object space. Moreover, each illustration has its own expressive purpose, producing an illustration is essentially an optimization process with given goals and constraints. Therefore, we "open" the illustration pipeline to user's manual interaction. That is, the system automatically produces a default illustration picture, and then the user can further refine the resulting illustration by the following operations:

1. Modification of CSG primitive's visibility.

The spatial layout of 3D CSG models is maintained as a priority list based on their depth to viewpoint. The users are allowed to explicitly change the priority (visibility) relations, in case he/she needs to expose internal invisible parts to reinforce the critical engineering/structure details such as assembly instructions.

2. Modulation of illumination. Each CSG primitive in the spatial layout has its own lighting description. The system will firstly calculate the lighting using default illumination. Then the users can interactively and iteratively fine-tune the parameters of lighting models, re-map the attributes of pictorial units and re-distribute them to form the new illustration, until desirable visual effects (tone, contrast, etc.) are achieved.

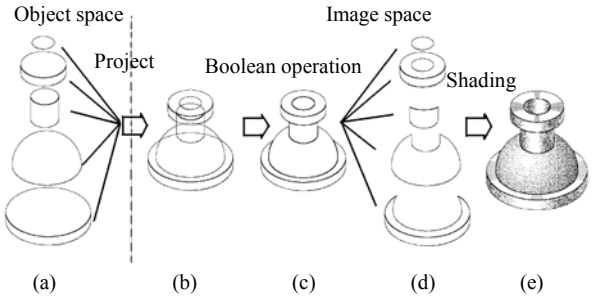


**Fig.1 The technical illustration pipeline based on CSG primitives. (a) Primitive types to be selected; (b) The object composed from primitives; (c) The object after hidden surface removal; (d) Spatial layout of the object to be rendered; (e) The resultant illustration**

## EMPIRICAL NPR LIGHTING MODELS FOR CSG PRIMITIVES

Inside the NPR lighting models, illustration conventions and empirical rendering knowledge (Hertzmann, 1998) should be taken into consideration. It will facilitate artistic control of lighting during the illustration process. Also, the technical illustration is desired to preserve the correct geometric projection and lighting distribution. Therefore, we defined the NPR lighting models as follows:

1. The lighting intensity is represented as parameters of multiple discrete levels in terms of highlight, Mach bands, mid-tone, semi-dark and dark regions, and so on. Their corresponding regions in image space are defined as "principal regions". For each level of lighting, the default value is calculated by the Phong illumination model.

2. The spatial size, location and shape of these principal regions in image space are parameterized by projection conventions and type of CSG primitives.

3. The lighting distribution over each CSG primitive is computed by interpolating the lighting intensities of its principal regions.

4. Each CSG primitive has its own lighting model, and each level of the lighting is subject to user's manual adjustment based on user's preferences or illustration styles.

Based on this definition, our NPR lighting models will have three major components: the multi-level lighting parameters, the parametric description of spatial occupations (principal regions) corresponding to each lighting level, and an interpolation method to calculate the lighting distribution over the entire primitive during run-time. The default lighting is the same as that of Phong model, except that we employ a region-based discrete calculation of lighting intensity, rather than Phong's pixel-by-pixel calculation. This feature somewhat reflects the lighting calculation utilized by human artists, as it offers the user a flexible interface of local refinement of lighting. More detailed descriptions of lighting model for each type of CSG primitive are given in Fig.2.

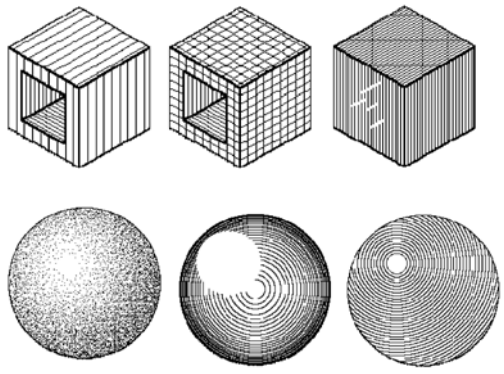

**Fig.2 Lighting and illustration for sphere and hexahedron**

Cylindric surface is divided into four principal regions as shown in Fig.3a. The light distribution is approximated and interpolated by sine function, as shown in Fig.3b.

For conical surface, the lighting distribution is first calculated by the lighting model in Fig.3, using

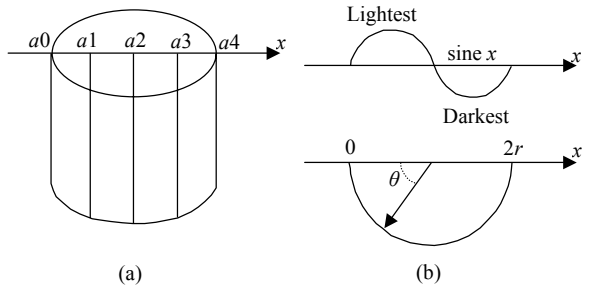


**Fig.3 (a) Segments in cylinderic surface; (b) Lighting model for cylinder**

the two radiuses of its bottom and top face respectively, and then we blend them along the major axis to mimic the illumination distribution of cone. The principal regions and the interpolation method are the same as those for cylindric surface.

In the case of revolving surface, we spread the revolving surface out as flat as a trapezium, and then divide it into sub-trapeziums as shown in Fig.4.

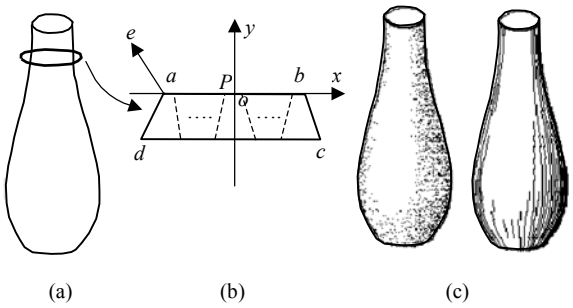


**Fig.4 The partition of revolving surface. (a) The revolving surface to be rendered; (b) The flattened sub-trapeziums; (c) The resulting illustration with different styles**

Lighting intensity at the sub-trapezium vertices is calculated as follows: let ***P*** be a vertex of the sub-trapezium, and ***Q*** be the 3D point in revolving surface corresponding to ***P***. Let ***ae***=[*x y z* 1] (as shown in Fig.4b) denote the vector perpendicular to ***ad***, and [*x*′ *y*′ *z*′ 1] is the surface normal at ***Q***. Then we have:

$$[x'\ y'\ z'\ 1]=[x\ y\ z\ 1]\begin{bmatrix}\cos\theta & 0 & -\sin\theta & 0\\ 0 & 1 & 0 & 0\\ \sin\theta & 0 & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

where $\theta$ is ***ae***'s revolving angle along *y*-axis, and is empirically approximated by $\frac{|pa|}{|ba|}\cdot\pi$.

After we get the normal, we can calculate the lighting intensity using Phong model. The light inside the sub-trapezium is approximated by the mean value of its four vertices. In case the revolution axis is not *y*-axis, geometric transformation will be performed to put it on the *y*-axis before the calculation of surface normal. Its principal regions are similar to those for the sphere primitive. But the lighting interpolation is performed bi-linearly in its spread-out trapezium.

## GRID-BASED SHADING FOR STYLIZED ILLUSTRATION

Stylized illustration takes charge of medium simulation, i.e. mimics the rendering styles, scatters/paints the pictorial units (dots, lines, etc.) into their spatial layout, and creates the resultant picture. In human's illustration, the painting is usually done region by region. Here, we propose an innovative grid-based method to simulate the visual effect of styles in technical illustration. Two major illustration styles, stippling and line-drawing, are explored.

In stippling, contours are drawn with curves in black color, and dots' distribution is processed grid by grid for each region in the spatial layout. Density of dots in each grid is determined by the lighting intensity. Some illustration samples are given in Fig.5. The pseudo codes to draw dots are given as follows:

```
Stippling_Drawing (Region in the spatial layout,
PrimitiveType)
{
  Partition the input region into grids by Primitive-
    Type;
  For each grid in the region (surface) do
  {
    Calculate light intensity according to the corre-
    sponding lighting models;
    Compute the dots' density (the number of re-
    quired dots) by the light intensity;
    For each dot to be drawn do
        Paint it inside current grid stochastically;
  }
}
```

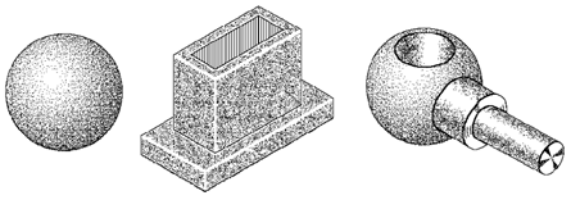

**Fig.5 Stippling samples**

In line-drawing, this process is much more complicated than that in stippling, as lines have a richer set of attributes like segment length, width, orientation, variation from baseline and line pattern (scratch and hatching, etc.), and all those attributes contribute to the resultant visual effects of line-drawing. We adopt the stroke-texture tactics (Winkenbach and Salesin, 1994) to simulate line-drawing styles. The parameters of line characteristics are pre-computed according to illustration style, CSG primitves' type and spatial occupation of principal regions, and then the system chooses the appropriate line strokes (segments) and textures (line-pattern) with the desirable attributes. These picked-out strokes or stroke-textures will be 'filled' into the corresponding region in the spatial layout via a grid-based approach like that in stippling illustration. The density of lines/segments (or line spacing) is calculated during runtime based on its changing light intensity.

For the flat surface in hexahedron, line attributes are easily extracted from its silhouette. Hatching and crosshatching are simulated as shown in Fig.2. Scratching will be applied automatically when line spacing is smaller than a predefined threshold. However, sometimes line strokes are desired to convey shape features of circles in cylinders or cones. Therefore, arc-based drawings and concentric arcs are introduced as shown in Fig.6.

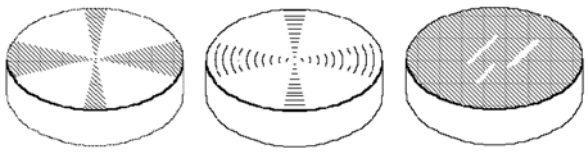

**Fig.6 Arc-based line-hatching illustration for circles**

For cylindric surface, line spacing is obtained via interpolation. The line densities in the four principal regions (shown in Fig.3a) are respectively calculated based on the four-level lighting intensities. The overall line distribution in the visible region will be computed by linear interpolation of them. Crosshatching will be applied to dark regions if applicable (Fig.7). Line-drawing for cones can be simulated similarly.

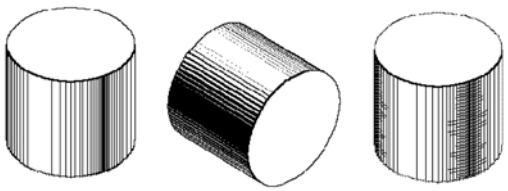

**Fig.7 Line-drawing for cylinders: uniform straight lines, lines with thickness and crosshatching**

In the line-drawing of revolving surface, we need to harmonize the appearance of line-drawing with the shape of its projected outline in image space. We employ a sub-trapezium grid-division method shown in Fig.4. In the trapezium, the top and bottom edges are further partitioned into a set of equal parts according to the density of lines. The partition-points are then correspondingly connected to form a set of candidate lines/segments. The actual drawing and connection of them are determined by the lighting in each grid. It ensures that the line-drawing somewhat reflects the shape of silhouette outline (shown in Fig.4c).

## RESULTS AND DISCUSSION

Automatic generation of technical illustration from 3D object is highly desired by CAD/CAM systems. We implemented this CSG-based illustration approach in a prototype system, RETOUCH, using Visual C++ 6.0. It supports six types of CSG primitives: hexahedron, sphere, cylinder, cone, prism and revolving solid. A visual evaluation of their resultant illustration is given in Fig.8.

The major contributions of our work are:

1. The NPR lighting models in designer's way. We extended Phong's lighting model by discretizing its lighting calculation. A set of multiple level lighting parameters is associated with principal regions of concern to the designer, and each lighting level is subject to the user's adjustment. The lighting calculation over the whole primitive is done by interpolat-

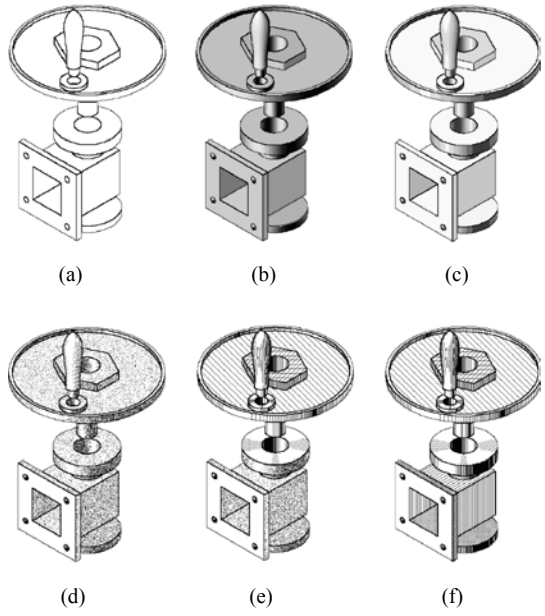


**Fig.8 Comparing with Phong shading (b) and Gooch's NPR lighting models (c), illustrations [(d), (e), (f)] produced by RETOUCH, more engineering information are indicated, for instance in (d) all surfaces are rough, maybe casting by mould. (e) & (f): some machine-tooled processing(buffing, grinding, rasping, etc.) have been done**

ing lighting intensities of its principal regions. The major benefits it offers are: (1) the user can artistically and flexibly modulate the lighting distribution by locally changing the lighting parameter in each level, and the system automatically re-computes the concrete lighting distribution dynamically; (2) it provides more information about the layout of lighting regions of designer's interest, and makes it possible to incorporate more artistic conventions into the illustration and media simulation.

2. Grid-based shading for stylized illustration. In photorealistic rendering, the resulting image is usually painted pixel by pixel. It shares the same principle of real photo-taking exposure–point by point. However, pixel/point tends to focus more on details rather than conveying high-level meaningful information, and therefore it works well with photorealistic rendering, but fails in expressive rendering. We somewhat make a breakthrough in this aspect by proposing a grid-based method to 'fill' pictorial units into regions, and accordingly accomplish the illustration region by region. Illustration samples in Fig.9 and Fig.10 demonstrate its effectiveness and convenience in simulating the NPR illustration styles.

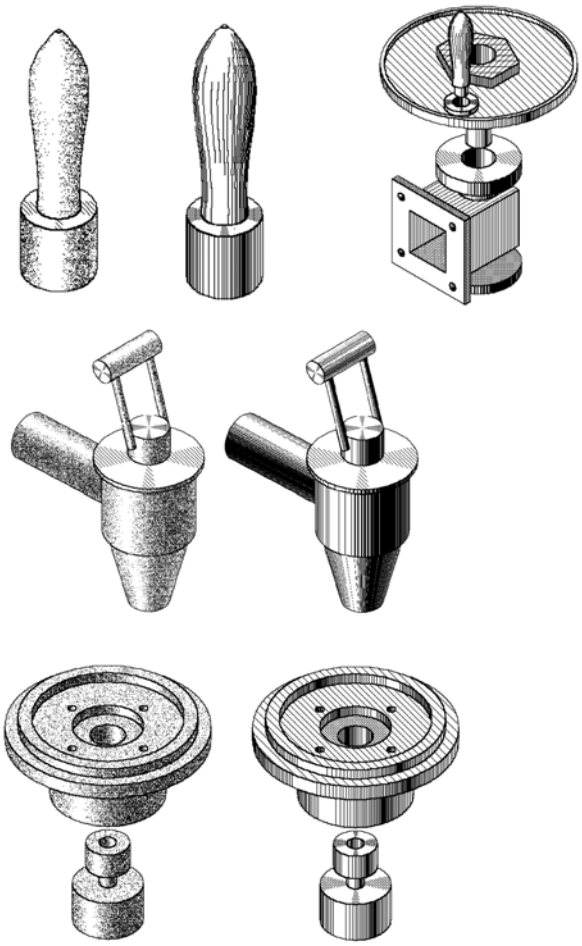

**Fig.9 Illustration samples of mechanical components**

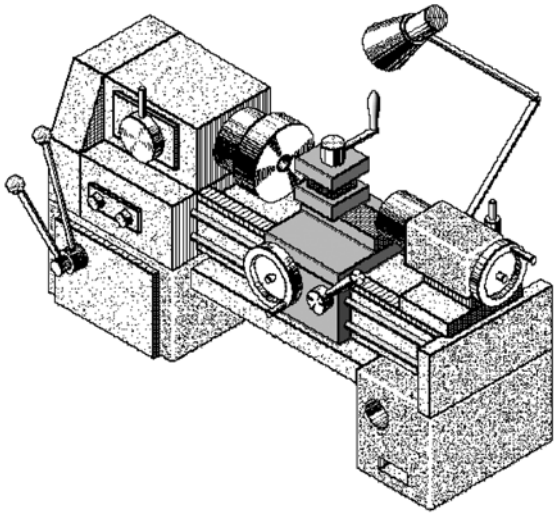

**Fig.10 An illustration instance of a lathe generated by RETOUCH. The user takes around one hour to accomplish the modeling and rendering**

The limitations of our CSG-based approach are obvious:

1. There are about 5% mechanical components that cannot be represented by CSG models (Mantyla, 1988). To alleviate this problem, we implement an interactive CSG construction tool. The user can directly input the CSG primitives and their modeling relationship, in case the CSG representation are not available, or the object to be rendered can only be approximately represented as CSG model in terms of resultant visual effect.

2. The current implementation works well only for mechanical illustration. The illustration conventions and medium simulation are represented and implemented in procedural forms, which make it difficult to extend to other domains or include new illustration styles.

## References


Agrawala, M., Stolte, C., 2001. Rendering Effective Route Maps: Improving Usability Through Generalization. SIGGRAPH'2001 Conference Proceedings, p.241-250.

Buchanan, J.W., Sousa, M.C., 2000. The Edge Buffer: A Data Structure for Easy Silhouette Rendering. Proceedings of the First International Symposium on Non Photorealistic Animation and Rendering (NPAR'00), Annecy, France, p.39-42.

Dooley, D., Cohen, M.F., 1990. Automatic illustration of 3D geometric models: surfaces. *IEEE Computer Graphics and Applications*, **13**(2):307-314.

Durand, F., 2002. An Invitation to Discuss Computer Depiction. Proceeding of Non-photorealistic Animating and Rendering 2002 (NPAR'02), Annecy, France, p.111-124.

Geng, W.D., Fleischmann, M., Yu, H.F., Pan, Y.H., 2001. Technical Illustration Based on Human-like Approach. IEEE Proceeding of Computer Graphics International 2001 (CGI'2001), IEEE Computer Society, Hong Kong, p.343-346.

Gooch, A., Gooch, B., Peter, S., Cohen, E., 1998. A Non-Photorealistic Lighting Model For Automatic Technical Illustration. SIGGRAPH 98 Conference Proceedings, ACM Press, New York, p.447-452.

Hertzmann, A., 1998. Painterly Rendering with Curved Brush Strokes of Multiple Sizes. SIGGRAPH 98 Conference Proceedings, ACM Press, New York, p.453-460.

Lansdown, J., Schofield, S., 1995. Expressive rendering: a review of nonphotorealistic techniques. *IEEE Computer Graphics & Applications*, **15**(3):29-37.

Mantyla, M., 1988. An Introduction to Solid Modelling. Computer Science Press, Maryland.

Satio, T., Takahashi, T., 1990. Comprehensible Rendering of 3-D Shapes. SIGGRAPH 90 Conference Proceedings, ACM Press, New York, p.197-206.

Schumann, J., Strothotte, T., Laser, S., 1996. Assessing the Effect of Non-photorealistic Rendered Images in CAD. Proceedings of CHI'96, Vanconver, Canada, p.35-41.

Seligmann, D.D., Feiner, S.K., 1991. Automated Generation of Intent-based 3D Illustration. SIGGRAPH 91 Conference Proceedings, p.123-132.

Willats, J., 1997. Art and Representation. Princeton University Press.

Winkenbach, G., Salesin, D.H., 1994. Computer-Generated Pen-and-Ink Illustration. SIGGRAPH 94 Conference Proceedings, ACM Press, New York, p.91-100.

Winkenbach, G., Salesin, D.H., 1996. Rendering Parametric Surfaces in Pen and Ink. SIGGRAPH 96 Conference Proceedings, ACM Press, New York, p.469-476.