

# 3D Sketch: Sketch-Based Model Reconstruction and Rendering

Jun Mitani,<sup>†</sup> Hiromasa Suzuki,<sup>◊</sup> Fumihiko Kimura<sup>◊</sup>

<sup>†</sup> *Department of Information Engineering, The University of Tokyo, JAPAN*

<sup>◊</sup> *Department of Precision Machinery Engineering, The University of Tokyo, JAPAN*

**Abstract:** We propose a new modeling and rendering system that enables users to construct 3D models with an interface that seems no different from sketching by hand, and that displays models in a sketch-like style, preserving the features of the user's strokes. We call this system *3D SKETCH*. To reconstruct 3D objects from sketches, we limit the domain of renderable sketches and prepare a *template* for interpreting sketches. As long as a sketch can be matched to such a template, the system can reconstruct a mesh model from the sketch. The system collects information about strokes made, and uses that information for our rendering scheme.

**Key words:** Sketch input, Modeling, Non-photorealistic rendering

## 1. INTRODUCTION

Although the use of CAD systems has become commonplace, traditional sketching is often more efficient in the early stages of concept design. One reason is that CAD systems are too tedious to allow the quick expression of design ideas, thus interfering with the designer's creative process. Then, too, some designers do not like CAD images, which have a very different style from rough sketches. We propose a new modeling and rendering system that enables users to construct a 3D model with an interface equivalent to sketching, and that displays the constructed model in sketch-like style that preserves the features of the user's strokes. We call this system *3D SKETCH*. *3D SKETCH* is intended to support the early stages of the product design process, when design sketches are often drawn

quickly in order to visualize the rough geometric shapes of prospective products. The paper is organized as follows. Section 2 describes existing modeling and rendering methods. Section 3 describes a system for capturing the designer's input. Section 4 describes the 3D SKETCH modeling system for building mesh models. This system uses a template to interpret a sketch. Section 5 describes the 3D SKETCH rendering capability, which captures not only the fundamental shapes of strokes but also the essence of the user's strokes. Section 6 describes the results of our method. Section 7 summarizes our conclusions and presents plans for future work.

## 2. MODELING AND RENDERING

### 2.1 Modeling

In industrial applications, the WIMP (Window Icon Menu Pointing) device is popular. But because this interface is unintuitive and hard to learn, it is not well suited to the early stages of design. We are interested in system that enables the user to construct 3D models using a sketch interface.

The *SKETCH* system proposed by Zeleznik (1996) is based on *gestures*, each of which is a specific sequence of strokes. A gesture corresponds to some solid modeling operation. Though it is easy to create angular objects, it is hard to reflect the features of a designer's strokes in 3D models.

On the other hand, Igarashi (1999) proposed the *Teddy* system for generating *Teddy bears*, or rounded shapes. The system assumes that a sketched curve is a silhouette of an axis-symmetric object. The *Teddy* system provides a highly intuitive user interface and can reflect the features of a designer's strokes. However, it is difficult to use for modeling prismatic objects. Varley (2000a) proposed *RIBALD*, which can convert a sketch to a B-Reps solid model. However, several assumptions must be made about sketches in order for *RIBALD* to perform this conversion, and thus it cannot directly accept sketch input by designers.

Our system applies mirror symmetry to reconstruction. It is based on the method of Furushima et al. (1993), who applied mirror symmetry to reconstruct a surface model from a curved network. However, with that method a user must draw every curve even though some curves may be hidden behind the object. This hinders the natural sketch interface.

### 2.2 Rendering

Now, thanks to many studies, we can obtain computer graphics images rendered in almost photographic quality. Likewise, non-photorealistic

rendering, in which models appear as if sketched by hand or drawn as cartoons, has also been well studied.

In the early stages of concept design, it is desirable to render a model in the form of a sketch and to do so in a way that leaves the designer's creative process undisturbed. Markosian (1997) proposed a method to render outlines and feature lines speedily and to make real-time NPR possible. Both Zeleznik (1996) and Igarashi (1999) use NPR and display models as sketches. But these systems both use fixed algorithms to render models, and neither makes good use of the designer's strokes. Our system uses information about strokes to render the features both the look and the "feel" of those strokes in the resulting model.

### 3. 3D SKETCH

Our 3D SKETCH system consists of three parts. The first captures the sketched input and generates an edge graph from it. The second uses this graph to reconstruct a mesh model. The third renders the constructed model in a sketch-like style that includes the features of the hand-drawn strokes.

The flow of our 3D SKETCH system is shown in Figure 1. The boxes at the left of the figure are the stages of user interface, and the boxes at right are the system transaction stages, which are not seen by users. The system transacts the sketch analysis, model reconstruction, and rendering without any intervention by the user. Thus, from the user's standpoint a 2D sketch becomes a 3D model automatically.

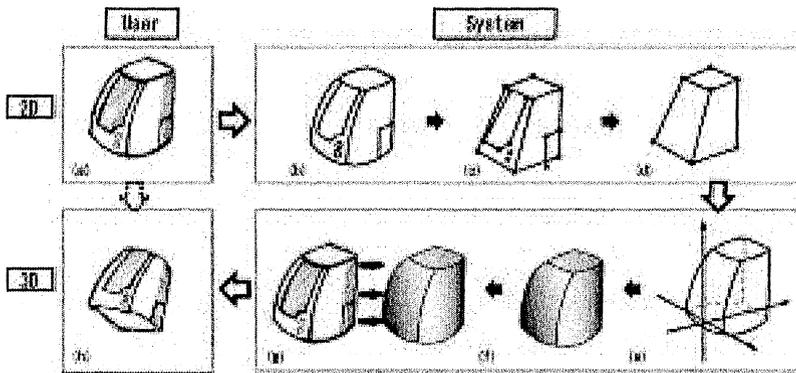


Figure 1. The flow of 3D SKETCH (Example of a speaker) (a) The user draws a sketch. (b) The core curves that represent the features of the model are extracted. (c) An edge graph of the core curves is constructed. (d) An edge graph needed for reconstruction is extracted. (e) 3D information is obtained. (f) A 3D solid model is reconstructed. (g) Sketch-style NPR is applied to the generated model. (h) User can operate the generated model.

### 3.1 Sketch Input

Various kinds of design sketches are used in design processes. In 3D SKETCH, we are particularly interested in sketches like those shown in Figure 2. All of these sketches are drawn with lines (which may be curved), so we call them *line sketches* hereafter. A designer draws a line sketch stroke by stroke; the strokes represent characteristic curves to define geometry, or hatching to represent surface texture.

We use a locator device, such as a tablet, to capture every stroke as a designer draws it. A stroke is defined as a series of locations generated by the locator device within a certain period of time or within a gesture (Figure 3 (a)). In our approach, a stroke is a fundamental element of a sketch.

The system needs to extract an edge graph, which represents the major lines (curves) of the sketch and their connectivity.

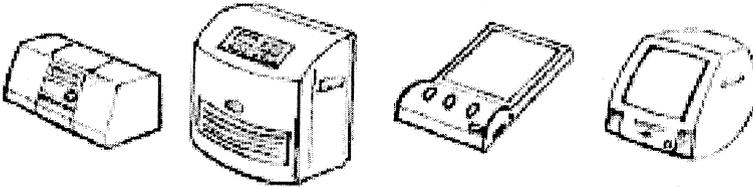


Figure 2. Examples.

### 3.2 Stroke Bundles

In a line sketch, the designer represents the shape of a product by drawing characteristic curves and hatching lines. A characteristic curve is not usually drawn as a single stroke but as a number of overlapping strokes. In this paper, a *stroke bundle* refers to such a set of strokes that collectively represent a characteristic curve (Figure 3 (b)). The stroke bundle is approximated by a *core curve*, which is extracted from the stroke bundle (see the dotted curve in Figure 3 (b)).

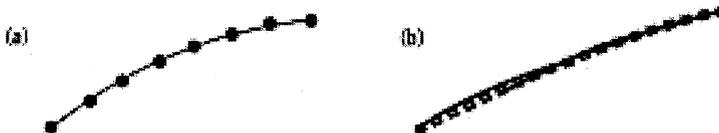


Figure 3. (a) Stroke, (b) stroke bundle (solid curve), and core curve (dotted curve).

### 3.3 Core Curve Tracking

As a designer defines a characteristic curve incrementally, stroke by stroke, the corresponding stroke bundle extends. Thus we have to update the core curve each time a new stroke is added to the bundle. Figure 4 shows an example. Figure 4 (a) illustrates a bundle of two strokes (solid curves) and its core curve (dotted curve). In Figure 4 (b) a new stroke is added, and the core curve is updated as shown in Figure 4 (c). Matsuda (1999) proposed an updating method for strokes of parametric curves. Here we propose a method for polylines. The way to update a core curve is detailed in our paper (Varley 2000b).

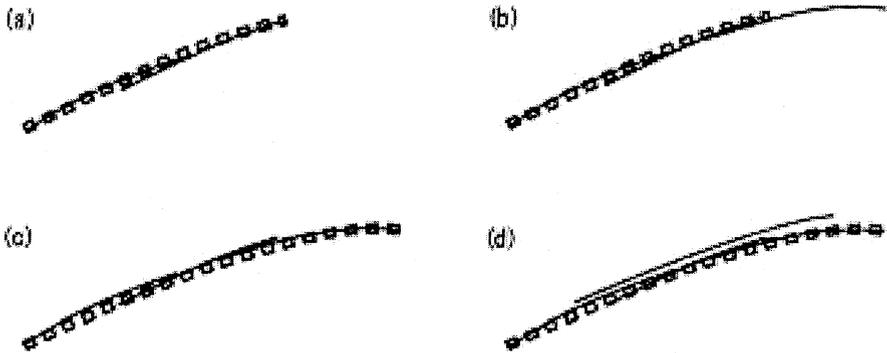


Figure 4. Updating of core curves. (a) Strokes (solid curve) and core curve (dotted curve). (b) Addition of a stroke. (c) Extending the core curve. (d) Not updating the core curve for an inclusive stroke.

### 3.4 Extracting an Edge Graph from Core Curves

After all the strokes are made, we have a set of core curves and their stroke bundles. The next thing that we have to do is to connect these core curves to extract an edge graph. The edge graph represents a network of characteristic curves of an object--more specifically, it represents the edges of a polyhedral object.

Core curves correspond to the arcs of the graph, and the end points of core curves correspond to the nodes of the graph. Thus the graph is extracted by identifying a node with end points of core curves that lie within a threshold distance of each other. (As mentioned earlier, some strokes represent surface hatching and so are not considered part of the edge graph.) To accommodate this extraction, we assume that the graph is connected. Thus if a core curve cannot be connected to any other core curves, it is not included in the graph.

## 4. BUILDING MESH MODELS

In this section we describe a method to construct a 3D mesh model of an object from a single line sketch.

### 4.1 Domain of Input Sketch

As mentioned above, we need to make some assumptions in order to construct a 3D model from a single sketch. First of all, we limit the domain of sketches to those shown in Figure 2. More precisely, an object is required to satisfy the following conditions: it must have six faces of four sides, it must have mirror symmetry, and the back face and bottom face must each be planar. The sketch of the object must be drawn from such an angle that three particular faces (top, front, and left) can be seen.

Such conditions might be considered too restrictive, but all the examples in Figure 2 satisfy them, since we allow curved surfaces. Particularly, a wide variety of electronic appliances meet these conditions.

### 4.2 Matching Edge Graph to Template

After the strokes are made, 3D SKETCH generates an edge graph. According to the conditions described in the previous section, the graph should look like the diagram shown in Figure 5. It should have nine major characteristic curves that are connected at seven nodes,  $V_1$  to  $V_7$ . Thus, we first check to see whether or not the graph is isomorphic to this template.

The diagram in Figure 5 is used as a template for reasoning about the 3D geometry. We could use other, similar templates to apply our approach to other categories of sketches.

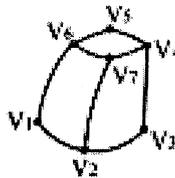


Figure 5. Matching graph structure from core curves.

### 4.3 Calibration of Camera Parameters

A sketch can be regarded as a projected view of an object, although the projection is not accurate. In order to generate 3D information, we have to

estimate camera parameters of the projection that correspond to the viewpoint of the designer.

Consider two parallel lines in 3D space. Their projections are also straight lines in the 2D drawing space. If these lines intersect, the intersection is called a *vanishing point*. If we can find three vanishing points for the sketch, we can estimate the COP (center of projection) using a method proposed by Kondoh (Kondoh 1988).

So we first find the vanishing points on the sketch by using a least-squares approximation.

The calibration of the camera parameters is detailed in our paper (Varley 2000b).

#### 4.4 Generating 3D Edge Curves

We can get 3D information about curves by using the camera parameters and some constraints. The reconstruction of 3D positions using mirror symmetry is based on the method of Furushima et al. (1993). We apply this method to our template to determine the 3D edge curves.

For our template, the object has a 3D structure of the type shown in Figure 6. It has six faces and 12 edges. The connectivity of these faces and edges is also defined. We first compute the 3D polylines for those 12 edges, and then generate polygonal meshes for those faces surrounded by those edges.

The edge graph of a sketch is assumed to have core curves corresponding to the nine edges numbered (1) to (9) in the figure. The edges marked (1) through (12) can be computed as follows.

**(1) & (9), (6) & (8): Pairs at mirror-symmetrical positions:** (1) & (9) and (6) & (8) have mirror symmetry, so they can be determined. First, the curves are resampled so that both curves of a pair have the same number of uniformly spaced points. As each point on one core curve has a corresponding counterpart on the other core curve (at positions of mirror symmetry), their three-dimensional positions can be determined.

**(2), (5), (7): Self mirror symmetry:** By dividing these curves at their midpoints, they can be considered a pair of core curves with mirror symmetry. In addition, core curve (2) lies on the  $xy$  plane, so, after computing the 3D geometry based on the mirror symmetry, we further correct it to lie in the  $xy$  plane.

**(3), (4): Planar constraint:** Every point of core curve (3) lies on the  $xy$  plane and every point of (4) lies on the back face. This back face is parallel to the  $y$  axis and can be defined using one of the end points of core curve (5). Thus we can determine the core curves' 3D positions.

(10), (11), (12): Finally, we define the hidden edges. Edges (11) and (12) are at mirror-symmetrical positions to (4) and (3), respectively. Thus, they can be computed as the mirror images of (4) and (3), which have already been defined. Edge (10) is an intersection of the  $xy$  plane and the back face, so it can be defined by connecting the end points of (3) and (12).

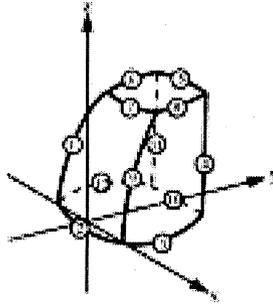


Figure 6. Determining polylines for 12 edges.

## 4.5 Surface Generation

In order to generate a mesh model, the surfaces for the six faces of the object are generated by interpolating their four boundary edges using Coons patches (Farin 1997). The polyline of an edge can be regarded as a parametric curve. All the edges except edge (10) have been generated from resampled core curves. Thus, they are also polylines having almost uniform spacing between the vertices, and therefore can be considered as a series of points uniformly sampled from a continuous curve. Edge (10) is a straight line segment, so its parameterization is trivial. Using the Coons patch, we uniformly sample the surface to generate a regular triangular mesh.

## 5. SKETCH-STYLE NPR

Our 3D SKETCH system is designed for the early stages of concept design. To preserve the designer's expression and not disturb the designer's creative process, we use real-time non-photorealistic rendering. The usual NPR methods use predefined static algorithms to generate sketch-style images from 3D model data, because the input data include only geometric information. On the other hand, our system has not only 3D geometric information but also all of the sketch stroke information the designer entered to express. The strokes include character lines and hatching that express the state of the surface. Thus, we attempted to use the stroke information to more fully render the model. Once this has been done, the model is displayed

as an image that looks as if it had been sketched directly by hand, even when viewed from different directions. Here we describe how to apply the stroke information to the reconstructed model.

## 5.1 Stroke Mapping to the Faces

The data on strokes entered by the user to express the “feel” of the model are stored as a sequence of polylines. To apply the user’s expression to the model, the polylines are mapped onto the surfaces of the reconstructed model, in what can be regarded as a texturing method. The system first projects the 3D model on a sketching screen with a camera projection matrix obtained in the previous section. Then we try to find the correspondence between the projected model and the sketched strokes (Figure 1(g)).

The model consists of triangle mesh, and the strokes (i.e. the polylines) are sequences of vertices. For each vertex, the system finds a triangle that contains the vertex and barycentric coordinates of the vertex in this triangle. If a vertex is contained in multiple triangles, only the triangle nearest the camera position is used. When a triangle is rendered, the segments mapped on the triangle are drawn together.

Though this method works well for strokes drawn on faces, strokes drawn as character lines sometimes are not displayed as we would expect, especially when the character lines form a silhouette.

For example, a portion of the strokes that compose a silhouette of a box drawn with multiple strokes (Figure 7(a)) disappears when the box is rotated (Figure 7(b)). This is because some strokes lie on the outside of the projected model, so that the system failed to find triangles that contain vertices of the strokes. Even if the triangles containing strokes are found, a similar problem occurs when the strokes become a silhouette (Figure 7(c)). This is because the triangles on which the strokes are mapped go to the back side, making strokes invisible.



Figure 7. The silhouette strokes mapped onto faces disappear when viewed from different directions.

## 5.2 Stroke Mapping onto Character Lines

To solve the problem described in the previous section, we propose a new method that is applied to strokes representing character lines. The system keeps the information about the relative positions of the strokes to core curves, and draws strokes using this information to display the character lines. For example, when the character line (the dotted line in Figure 8(a)) drawn with multiple strokes (solid lines in Figure 8(a)) is rotated, strokes are remapped onto the character line using the information on relative positions; thus, this feature of the strokes is preserved. It can be regarded as texturing for edges rather than for faces.

Because the reconstructed model is a mesh, the 3D character line is a polyline. So we have to find a polyline segment that corresponds to each stroke. Although it is easy to obtain information about the positions of the strokes relative to the core curves, it is not easy to obtain a 3D character line that corresponds to each stroke. We discuss how to get this correspondence.

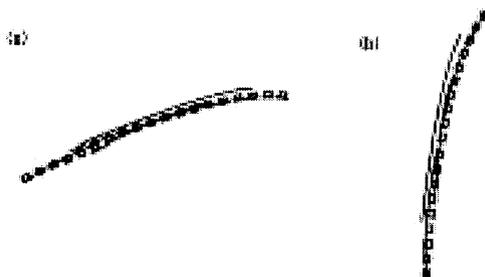


Figure 8. A character line and related strokes.

We abbreviate some terms as follows:

**PC:** Projected Character line of the reconstructed 3D model.

**PCS:** Projected Character line Segment.

**S:** Stroke entered by a user.

**SS:** Stroke Segment.

**C:** Core curve related to multiple strokes.

**CS:** Core curve Segment.

**$A \in B$ :** A is an element of B.

**$A \xleftrightarrow{1:1} B$ :** A has a 1:1 relationship to B.

**$A \xleftrightarrow{1:n} B$ :** A has a 1:n relationship to B.

At first, we obtain the information on the relative position of each SS to a CS. This information consists of the values of the relative position parameters  $l$  and  $t$ , and the references to the nearest CS from the SS (Figure 9(c)).  $l$  is the distance between the end point of SS and the CS, and  $t$  is the

value of the foot of SS on the CS that is between 0 and 1, where the start position of the CS corresponds to 0 and the end corresponds to 1.

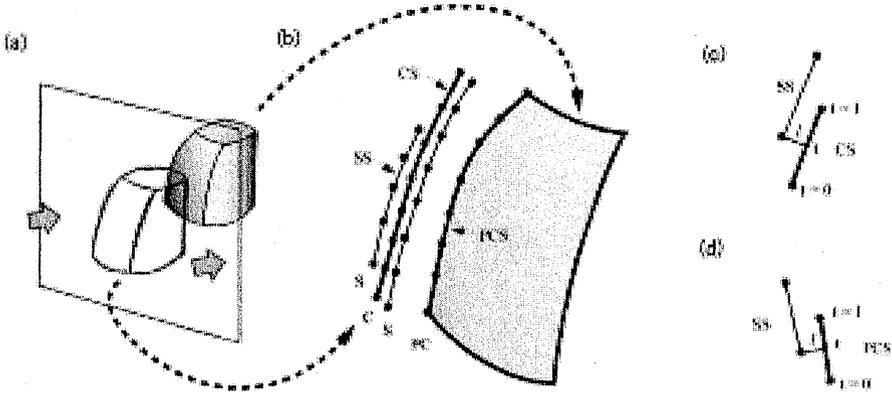


Figure 9. Stroke mapping onto character lines.

Then the system finds the correspondence between each PCS and each SS. When the system renders a character line, the SSs instead of the PCS are drawn, applying the relative parameters  $l$  and  $t$  (Figure 9(d)).

When a model is generated from a sketch, the following relation is known.

$$\begin{aligned}
 PCS &\in PC \\
 SS &\in S \\
 CS &\in C \\
 C &\xrightarrow{H} PC \\
 S &\xrightarrow{H} C
 \end{aligned}
 \tag{1}$$

The questions are how to obtain the correspondence between SS and PCS, and how to decide the relative position parameters. That is, we want to know the relation of

$$SS \xrightarrow{H} PCS$$

and the relative position parameters  $l$  and  $t$  (Figure 9(d)).

When the number of PCSs in a PC is  $m$ , we subdivide  $C$  to  $m$  CSs. Then we get the relation:

$$CS_i \xrightarrow{H} PCS_i \quad (0 \leq i < m) \tag{2}$$

After this subdivision, we calculate the parameters  $l$  and  $t$ . Now, we get

$$SS \xleftarrow{n:l} CS_i \quad (3)$$

from (1) and (2) (Figure 9(c)).

After that, we change the reference of the SS from  $CS_i$  to the  $PCS_i$  that corresponds to the  $CS_i$ . Then we get

$$SS \xleftarrow{n:l} PCS \quad (4)$$

and parameters  $l$  and  $t$ .

Figure 10 is an example of this method applied to a box. Even when the box is rotated, edges are displayed, thus keeping the features of the entered strokes. (There are no bottom side edges, because no strokes are mapped (Figure 10(c))).

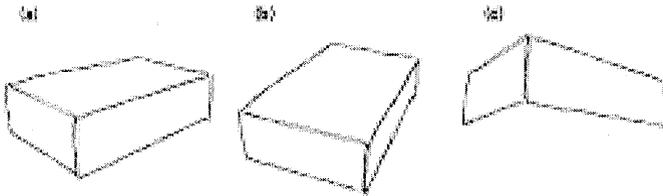


Figure 10. Strokes mapped on character lines.

## 6. RESULTS

Our prototype 3D SKETCH system has been implemented in Java on a personal computer (Pentium III 600MHz, memory 512MB). Figure 11 shows examples of an electronic gadget. Figure (a) shows the inputted sketch of 43 strokes. Figure (b) shows the core curves extracted from (a). In Figure (c), dots show the seven recognized vertices of the template, while the box shows a unit cube corresponding to the extracted world reference frame. Figure (d) is the reconstructed mesh model with 1304 vertices. The total CPU time needed to process the sketch data to produce each mesh model is about 20 seconds. Figure (e) shows the image our rendering methods produced using a user's strokes. Figure (f) is a smoothly shaded image from a different perspective. Figure (g) shows the images viewed from different perspectives. The rendering is swift enough to handle the model interactively.

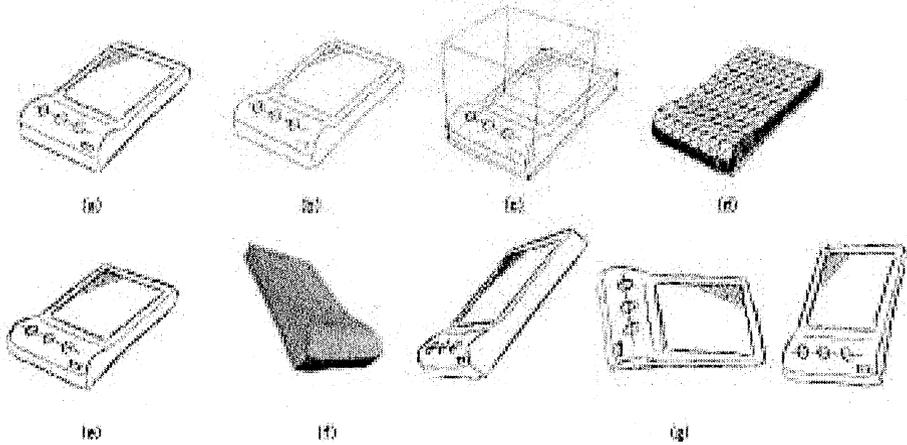


Figure 11. The result of a 3D SKETCH of an electronic gadget.

## 7. CONCLUSIONS

This paper has described the 3D SKETCH system as it targets the early stages of concept design. The system generated a model by interpreting a graph of a sketch. As long as the sketch is compatible with the prepared template, the user can construct a model in a natural sketching manner without being burdened by any computational operation.

The system renders the generated model with sketch-style NPR using information obtained about the user's strokes. The model looks like a hand-drawn sketch even when viewed from different directions. The rendering was swift enough to handle the model interactively.

For future work, we propose the following subjects:

- More-flexible templates should be developed, to enable the system to cover a wider variety of sketches.
- Although we generate surfaces as Coons patches from character lines, a designer may draw strokes to express the appearance of a surface. It is desirable for the system to generate the faces that reflect the strokes.
- With our system, only a single model can be constructed. We should make the system able to handle multiple models in one sketch to make it possible to construct more-complicated models.

## 8. ACKNOWLEDGEMENTS

We would like to thank Prof. Kunio Kondoh, Saitama University, Japan, for his helpful comments. We also thank Mr. Nihei of Konika Inc. He is an industrial designer who tested our prototype system and gave us important evaluations. Our special thanks also go to Prof. Ralph Martin, Cardiff University, England, for his valuable comments.

## REFERENCES

Farin, G. (1997). *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press.

Furushima, S., Kanai, S., and Takahashi, H. (1993). Generation of 3 Dimensional Free-Form Curve Models by Automatic Recognition of Rough Sketches (in Japanese). In *J. Japan Society for Precision Engineers*. pp. 105-110.

Igarashi, T., Matsuoka, S., and Tanaka, H. (1999). Teddy: A Sketching Interface for 3D Freeform Design. In *Proceedings of ACM SIGGRAPH '99*, pp. 409-416.

Kondoh, K., Kimura, F., and Tajima, T. (1988). Estimation of Viewpoint in Perspective Drawings and its Application (in Japanese). In *J. Information Society of Japan*. pp. 686-693.

Markosian, L., Kowawlski, M. A., Trychin, S. J., Boudev, L. D., Goldstein, D., Hughes, J. F. (1997). Real-Time Nonphotorealistic Rendering. In *Proceedings of ACM SIGGRAPH '97*. pp. 415-420.

Matsuda, K., Kondoh, K. (1999). Iterative Sequential Drawing for Freehand Sketch Input (in Japanese). In *J. Information Society of Japan*. pp. 594-601.

Varley, P. A. C. and Martin, R. R. (2000a). Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch -Frontal Geometry and Sketch Categorisation, *1st Korea-UK Joint Workshop on Geometric Modeling and Computer Graphics*, Kyung Moon Publishers, pp.113-128.

P.A.C. Varley, H. Suzuki, J. Mitani and R.R. Martin (2000b), Interpretation of Single Sketch Input for Mesh and Solid Models, *Int. J. Shape Modeling (in print)*.

Zeleznik, R. C., Herndon, K. P., and Hughes, J. (1996). SKETCH: An Interface for Sketching 3D Scene. In *Proceedings of ACM SIGGRAPH '96*, pp.163-170.