IGEL: A Virtual Heat Cutter for 3D Shape Modeling

Hitomi Imaizumi[*]
Ochanomizu University

Takayuki Itoh[†]
Ochanomizu University

Abstract

Heat cutter is a tool for processing styrene forms. Since blades of cutters are steal wires, user can freely bend them. We think this operation is intuitive for 3D shape modeling. IGEL virtually realizes the operation of the heat cutters by a sketch input system. IGEL has two different modes, 2D and 3D modes, and users can switch them freely. In 2D mode, users can draw the shapes of cutters freely. In 3D mode, users can cut the styrene forms using their own cutters. Our implementation represents the styrene forms as triangular meshes, and cuts the meshes according to the user's operation.

Keywords: 3D Modeling, Virtual Mockup, Heat Cutter.

1 Introduction

3D Computer Graphics is a very important technology for various entertainment fields, such as movie and video games. 3D shape modeling is an important technical step for computer graphics. Most of 3D computer graphics designers use professional shape modeling software, but many of them feel that such software may be difficult for novice users, because it is not very intuitive, and may require special knowledge and training. Today we enjoy various consumer-generated contents on the Web, such as movies uploaded onto video-sharing Web sites. However, we do not have many chances to enjoy consumer-generated 3D contents. We think one of the reasons is the difficulty of 3D shape modeling environments.

On the other hand, there have been many techniques for intuitively enjoying 3D shape modeling processes. Some of famous techniques applied sketch interfaces [Igarashi et al. 1999] [Karpenko and Hughes 2006] [Cherlin et al. 2005] [Ijiri et al. 2005], and some others mimicked real manufacturing processes such as cutting and engraving [Mizuno et al. 1999] [Mitani and Suzuki 2004] [Owada et al. 2004]. We think such kinds of enjoyable techniques may bring us motivation of 3D shape modeling for non-professional people. Also, such technique may contribute to reduce design duration and costs for professional users. The technique presented in this paper is also categorized to such kind of techniques.

This paper proposes IGEL, a virtual heat cutter for 3D shape modeling applying a sketch interface. Our study supposes the following conditions during the development of IGEL:

- We attempt to implement all the functions by intuitive operations of pointing devices, without selection of menu or keyboard operations.
- We attempt to mimics real manufacturing operations such as cutting or engraving.
- We do not suppose to use high-spec input devices or graphics systems.
- We do not suppose highly accurate modeling such as design of industrial products. Instead, we aim to satisfy high-level hobby users.

[*]e-mail: hitomi@itolab.is.ocha.ac.jp
[†]e-mail:itot@is.ocha.ac.jp

Figure 1 shows a snapshot of a heat cutter, which features an electrically-heated steal wire to process styrene forms. IGEL mimics the heat cutters to process 3D shapes. It supposes that users design both shapes of steal wires and its trajectory by using a sketch interface. Users can freely cut and process the initial shapes of virtual styrene forms, by firstly designing the shape of the virtual steal wire, and then moving it in a virtual 3D space. We think the process is enjoyable since we can design the shapes of the cutter: we can generate wave-shaped cross-sections by designing wave-like cutters or star-shaped holes by designing star-like cutters. IGEL enables such kinds of enjoyable shape modeling intuitively, by only several stroke inputs.

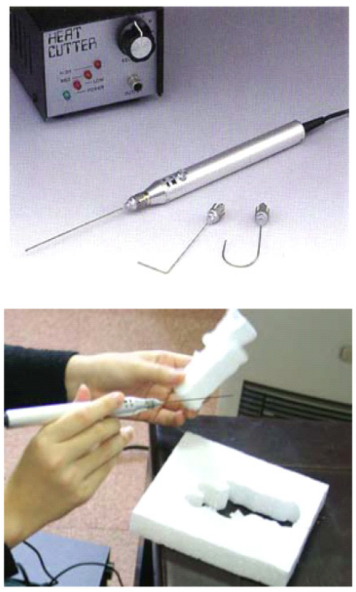


Figure 1: *A heat cutter.*

We implemented IGEL without supposing high-spec input devices: we only used a mouse or a pen tablet to implement and test IGEL. Also, we implemented virtual styrene forms as simple triangular meshes, not volume or point models, because we did not suppose to use high-spec graphics system. We named this technique "IGEL" as an abbreviation of "Interactive Graphics Enabling Light-modeling". Remark that "igel" means a hedgehog in German. We likened the cutters in our system to stings of a hedgehog.

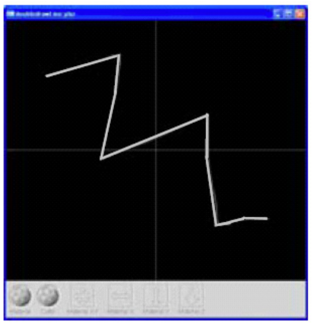
2 Related Work

Sketch-based 3D shape modeling is an active research topic [Igarashi et al. 1999] [Karpenko and Hughes 2006] to realize easy and friendly 3D shape modeling environments. These techniques enable easy operations based on 3D shape generation from 2D shape inputs. However, most of the techniques automatically decide the geometric features in the depth direction. This limitation may sometimes make sophisticated 3D shape modeling more difficult using such techniques. We think it is a trade-off between easiness and sophistication of 3D shape modeling.

Recent sketch-based 3D shape modeling techniques attempt to strike a good balance between the easiness and sophistication. Combination of parametric surface modeling and sketch interface is a good solution to balance them [Cherlin et al. 2005]. Also, several excellent 3D shape modeling tools for specific objects (e.g. flowers [Ijiri et al. 2005]) have been presented.

On the other hand, several intuitive 3D shape modeling techniques mimic real manufacturing processes and construct virtual tools as software. Virtual sculpture [Mizuno et al. 1999] and paper crafts [Mitani and Suzuki 2004] are typical examples of such kind of the works. These techniques are not only intuitive but also enjoyable because it realizes virtual experiences of real manufacturing processes. We think this is one of the values of virtual tool techniques.

IGEL is also a kind of virtual tools mimicking a heat cutter, featuring a sketch interface. Owada et al. presented a similar work [Owada et al. 2004] which cuts virtual objects by one stroke using a sketch interface, and maps natural-looking texture images onto the cross-section. Against this work, IGEL realizes more flexible shape modeling featuring two kinds of sketch interface: inputs of shapes and trajectory of cutters.



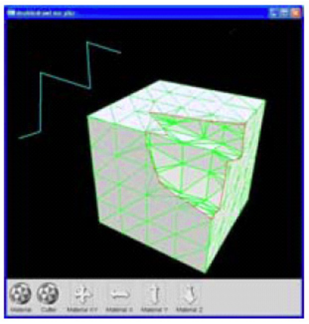


Figure 2: *(Upper) 2D mode for designing the shapes of cutters. (Lower) 3D mode for inputing the trajectory of the cutters.*

3 Processing Flow of IGEL

As mentioned in Section 1, IGEL mimics a heat cutter. It is not easy to freely deform the wires of real heat cutters due to these strengths. On the other hand, we implemented a sketch interface to freely deform it in our virtual heat cutter, so that we can flexibly and enjoyably realize 3D shape modeling.

3.1 Technical Overview

IGEL provides the following two modes to enable 3D shape modeling, as shown in Figure 2:

2D mode: a sketch interface to input the shapes of cutters.

3D mode: another sketch interface to input the trajectory of the cutters.

We did not suppose high-spec input devices while the development of IGEL, and therefore we implemented the both modes only with drag operations of popular pointing devices (mouse or pen tablet).

Also, we did not suppose high-spec graphics systems including GPU while the development, and therefore we implemented IGEL by only linking OpenGL and its extensions, without using GPU-based languages or video game development kits. Consequently, we applied simple triangular polygon models, not volumetric or point models, as shape representation of the objects.

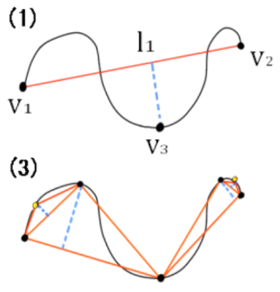
These assumptions are just based on our design policy. We do not opposite to implement IGEL-like techniques applying volumetric or point models, and using high-spec input devices or graphics systems.

3.2 2D mode

IGEL provides a 2D sketch interface to design the shapes of virtual cutters, as shown in Figure 2 (Upper). It displays the trajectory of the cursor representing the shape of the virtual cutter, when a user drags a pointing device.

IGEL interpolates the curved trajectory into a polygonal line, as shown in Figure 3. This algorithm is similar to well-known vectorization algorithms of raster images. Our implementation first generates a segment I1 connecting the start point v1 and end point v2 of the trajectory. It then finds the point v3 where is on the trajectory and the distance to I1 is maximum, and divide I1 into I2 and I3, as shown in Figure 3(1)(2). Our implementation recursively repeats the division of the segments Ij until the maximum distance between the segments and trajectory gets smaller than the predefined threshold, as shown in Figure 3(3). Figure 3(4) shows an example of the polygonal line interpolating the original shape of the virtual cutter. The algorithm drastically reduces the number of vertices constructing the curve of the virtual cutter, so that we can reduce the computation time for cutting the virtual objects.

Here, we can cancel the shape by drawing again, if we do not prefer already drawn shape.



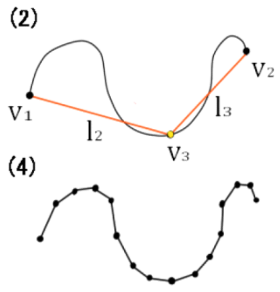


Figure 3: *Interpolation of shape of virtual wire into a polygonal line.*

3.3 3D mode

IGEL provides another sketch interface to input the trajectory of virtual cutters, as shown in Figure 2(Lower). IGEL enables it when a user finishes the input of the shape of the cutter and switches to the 3D mode. It displays the cutter in the 3D space in a different color when the user starts the dragging operation. It then calculates the intersection between a virtual object and the trajectory of the cutter according to the dragging operation. When a user finishes the dragging operation in the 3D mode, IGEL cuts the virtual object as mentioned below. Here, current our implementation only supports translation on a projection plane to operate the cutter in the 3D space.

3.3.1 Cross-Section Mesh

IGEL realizes the cutting operation by calculating the intersection between a virtual object and the curved surface generated as the trajectory of the virtual cutter. This paper defines a triangular mesh that interpolates the curved surface as "cross-section mesh". Our implementation generates the cross-section mesh while a user inputs the stroke, and completes when the user finishes the input operation. Figure 4 illustrates the generation of the cross-section mesh. It evenly generates a series of vertices on the trajectory, indicated as "sampling line 1" or "sampling line 2" in Figure 4. It then completes the generation of the cross-section mesh by connecting the vertices.

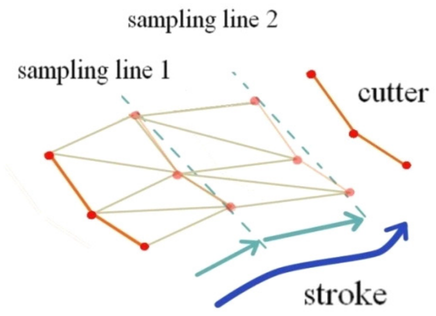


Figure 4: *Generation of cross-section mesh.*

3.3.2 Mesh-Mesh Intersection

IGEL calculates intersection between the cross-section mesh and the mesh of a virtual object. Figure 5 illustrates the process, where pink triangles are parts of the virtual object, and sky-blue triangles are parts of the cross-section mesh. The process detects intersections between a triangle of a mesh and an edge of another mesh. Traversing adjacent triangles one-by-one, and detecting the intersections of the traversed triangles, the process generates a set of segments connecting the intersections, shown as red arrows in Figure 5. This paper defines the intersection as "mesh-mesh intersection". Current our implementation completes the traverse of triangles when it arrives at the first triangle and therefore the set of segments constructs a loop. The current implementation does not support the generation of multiple loops yet. Also, it does not support the stop of cutting before constructing a closed loop, to generate incisions.

3.3.3 Mesh Division

IGEL divides the triangles so that we can cut the virtual objects, when the mesh-mesh intersection process is completed. IGEL

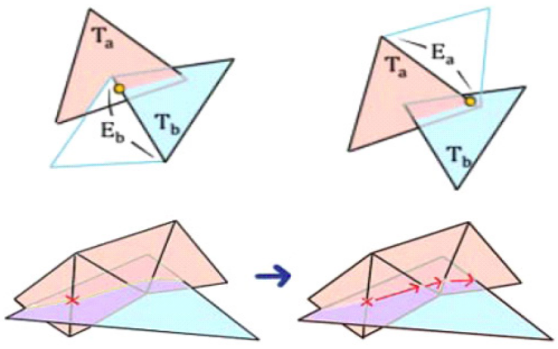


Figure 5: *Generation of mesh-mesh intersection.*

treats the segments constructing the mesh-mesh intersection as constraints, and applies the incremental constraint Delaunay triangulation algorithm [Sloan 1987] to adequately divide the triangles.

Our implementation of the mesh division consists of two steps. The first step divides the triangles by adding the new vertices on the mesh-mesh intersection. The second step modifies the triangles based on the constraints of the segments on the mesh-mesh intersection.

Figure 6 illustrates the division of triangles in the first step. The algorithm adds new vertices on the mesh-mesh intersection to the mesh, and divides the triangles so that the added vertices are adequately connected. A red point enclosed by a pink triangle, shown in Figure 6(1), is an example of the added triangle. The algorithm firstly divides the triangle enclosing the added vertex, as shown in Figure 6(2). It then checks the geometry of the divided triangles with its adjacent triangles, painted in yellow in Figure 6(3). It then swaps some edges of the triangles, if the geometry of triangles is improved by the swapping. Sky-blue triangles in Figure 6(4) are examples of the triangles generated by the edge swapping. The process adequately divides the triangles by recursively repeating the above process.

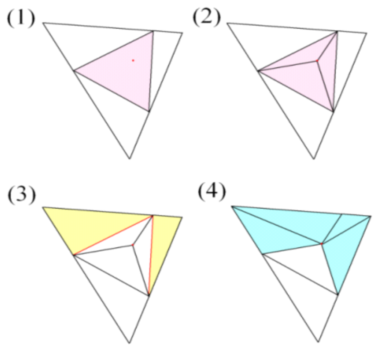


Figure 6: *Division of triangles.*

Figure 7 illustrates the swapping of triangles in the second step. Here, a blue dotted segment in Figure 7(1) is an example of the constraints. Our implementation extracts the four triangles intersected by the constraint segment, and swaps two of the triangles if the number of intersected triangles is reduced by the swapping. In this example, swapping of two triangles painted in Figure 7(2) does not reduce the number of intersected triangles, but swapping of two triangles painted in Figure 7(3) reduces it. Figure 7(4) shows after swapping of triangles painted in Figure 7(3), where the number of intersected triangles gets three. By swapping the painted triangles in Figure 7(4), the number gets two, as shown in Figure 7(5). Finally, the number gets zero, by swapping the painted triangles in Figure 7(5). Figure 7(6) is the final result of the process.

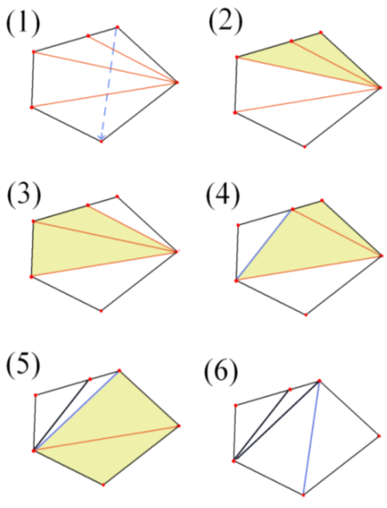


Figure 7: *Division of triangles.*

IGEL applies the process to both virtual object mesh and cross-section mesh. Here, a part of the cross-section mesh enclosed by the mesh-mesh intersection can be reused as the cross-section of the virtual object mesh.

3.3.4 Remaining Part Selection

When a mesh division process is completed, IGEL displays the two parts of the virtual object mesh divided by the cross-section mesh in two colors, as shown in Figure 8(Left). Users can click either of the two parts to specify the remaining part. When a user specifies the remaining part, IGEL maps the part of the cross-section mesh enclosed by the mesh-mesh intersection into the virtual object mesh. As a result, IGEL generates the new shape of the virtual object, as shown in Figure 8(Right).

After completing the remaining part selection, users can cut the virtual object by the same virtual cutter. Or, they can return to 2D mode to design a different shape of cutter.

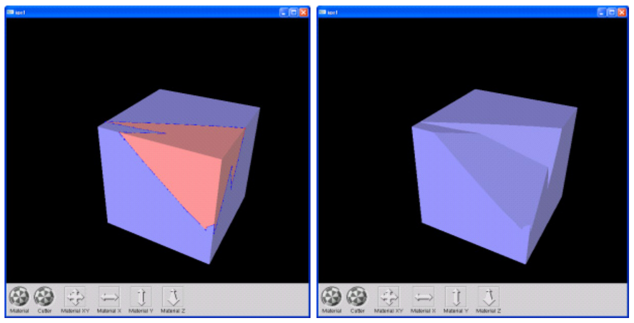


Figure 8: *Selection of remaining part.*

3.4 User Interface

Current our implementation uses GLUI, as an extension of OpenGL, for implementation of graphical user interface of viewing operations. The implementation provides gadgets for rotation and translation of virtual objects and viewpoints. The implementation still needs some keyboard-based operations such as switch between 2D or 3D modes. However, we would like to improve the user interface so that we can operate IGEL only by pointing devices.

3.5 Example

Figure 9 shows an example of shape modeling by cutting a virtual object five times.

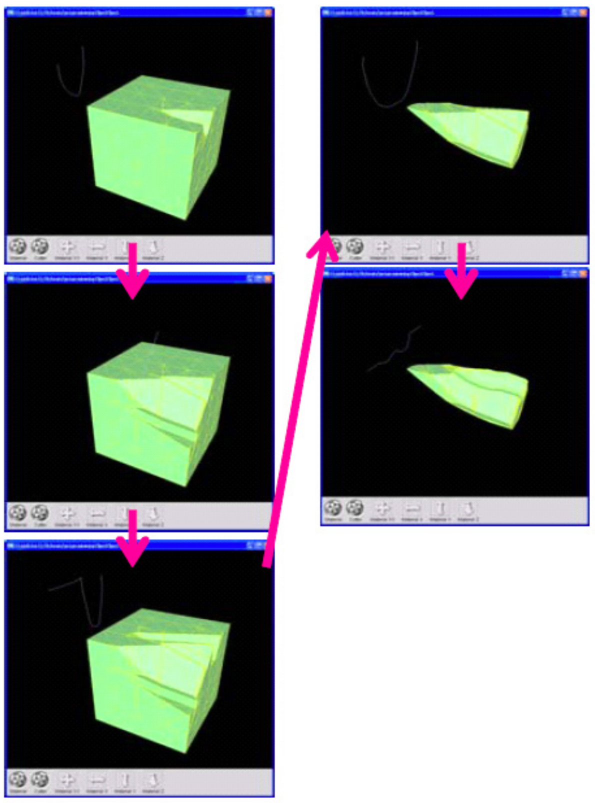


Figure 9: *An example.*

4 Conclusion and Future Work

The paper presented IGEL, a virtual heat cutter for 3D shape modeling. It provides two modes of the sketch interface. One of the modes enables to design the shapes of the virtual cutters, and the other enables to manipulate the cutters.

The following are our on-going and future works for the extension and improvement of IGEL.

[Extension of cutting operations] As described in Section 3.3.2, current our implementation does not support the situation that mesh-mesh intersection forms multiple loops. However, it is natural that we generate multiple loops while cutting bumpy objects. We would like to extend our implementation so that we can cut multiple parts in one stroke. Also, we would like to support the stop of the virtual cutter before the mesh-mesh intersection constructs a loop, so that we can generate incisions in virtual objects.

[Hole generation] Current our implementation does not support generation of holes. We need to extend our implementation so that it can recognize two loops of mesh-mesh intersection are topologically connected and they form a hole. We would like to implement it because generation of holes is an important operation for virtual heat cutters.

[Non-cutting operations] Current our algorithm does not take the effect of heat into account. Therefore, it does not support melt of objects. We would like to implement such effect to mimic more real heat cutters.

[Evaluation] We have not had any evaluation with IGEL. We plan two kinds of evaluations at this moment. Firstly we would like to compare IGEL with existing shape modeling tools or real heat cutters. Secondly we would like to user evaluation with examinees.

[Discussion on input devices] Current our algorithm does not support translation of cutters along depth direction, and rotation of cutters. It is hard to implement while we limit the input devices to mouse or pen tablet. We would like to discuss what kind of pointing devices are preferable for IGEL. Also, we would like to test IGEL with haptic pointing devices, because we think force feedback of such devices are effective for users.

[Test with virtual reality technology] Display of objects and cutters is also an important point for intuitive shape modeling. We would like to test IGEL with virtual reality technologies, such as large or stereoscopic displays.

References

CHERLIN, J. J., SAMAVATI, F., SOUSA, M. C., AND JORGE, J. A. 2005. Sketch-based modeling with few strokes. In *Proceedings of the 21st Spring Conference on Computer Graphics*, 137–145.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freedom design. In *Proceedings of ACM SIGGRAPH 99*, ACM Press / ACM SIGGRAPH, 409–416.

IJIRI, T., OWADA, S., OKABE, M., AND IGARASHI, T. 2005. Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints. In *Proceedings of ACM SIGGRAPH 2005*, ACM Press / ACM SIGGRAPH, 720–726.

KARPENKO, O. A., AND HUGHES, J. F. 2006. Smoothsketch: 3d free-form shapes from complex sketches. In *Proceedings of ACM SIGGRAPH 2006*, ACM Press / ACM SIGGRAPH, 589–598.

MITANI, J., AND SUZUKI, H. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. In *Proceedings of ACM SIGGRAPH 2004*, ACM Press / ACM SIGGRAPH, 259–263.

MIZUNO, S., OKADA, M., AND TORIWAKI, J. 1999. An interactive designing system with virtual sculpting and virtual woodcut printing. *Computer Graphics Forum: Journal of the European Association for Computer Graphics 18*, 3, 183–193.

OWADA, S., NIELSEN, F., OKABE, M., AND IGARASHI, T. 2004. Volumetric illustration: Designing 3d models with internal textures. In *Proceedings of ACM SIGGRAPH 2004*, ACM Press / ACM SIGGRAPH, 322–328.

SLOAN, S. W. 1987. A fast algorithm for constructing delaunay triangulations in the plane. *Advances in Engineering Software 9*, 1, 34–55.