EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2011)
T. Hammond and A. Nealen (Editors)

Single-View Sketch Based Modeling

Alexis Andre[†1] and Suguru Saito[2]

[1]Sony Computer Science Laboratories, Inc.
[2]Tokyo Institute of Technology

Abstract

This paper presents a new sketch modeling system that is able to generate complex objects drawn from a unique viewpoint. The user draws the model in an iterative manner, adding simple parts to the existing object until completion. Each part is constructed from two construction lines (lines on the surface of the object that are planar and perpendicular to each other) whose orientation in the 3D space is uniquely determined by the system, and an optional silhouette. The system is then able to produce rough 3D reconstructions of drawings very easily by tracing over a sketch for example. Such models are perfectly suited to investigate their shade or shadow and they can be used as substitutes for more detailed models when the need for quick models is present. The user can also explore shapes directly on the system, refining the shape on the go in a oversketching way. The creation of models is very efficient, as the user models the shapes directly in the correct pose and orientation. Results show that the system is able to create complex objects without ever having to change the viewpoint.

1. Introduction

Since the beginning of 3D computer graphics, modeling has been a cornerstone of the field. The task of converting design sketches, drawings, concepts (in 2D) into 3D sets of coordinates for the computer to render with a given amount of realism on a screen has always been tedious. The traditional design workflow is often rooted in paper-based tasks: the process of creation is actually performed on paper [PG98], where the designer is limited by their imagination only. Once the shape has been fixed, the 3D modeling can start. This work aims at allowing paper-like exploration within a 3D modeling system, where the designer draws as if it were on paper, but is immediately proposed with a 3D shape for visual feedback. The possibility to draw directly on the image is appealing and is a characteristic of sketches (and sketch-looking rendering [SSLR96]).

Drawing on a piece of paper reflects the mental image the artist has of the object under study. By choosing a particular viewpoint and by drawing its corresponding representation, the artist gives us his vision of the object: it should be enough to understand the shape, and most important features should be present and visible from that viewpoint. Traditional modeling tools target complete 3D objects, where no particular view direction is chosen: it is natural to use multiple 2D views of the object to describe and interact with the model. A design sketch is however enough to get a global idea of its shape, and this is the target of this paper: create 3D models from a 2D sketch drawn from a fixed viewpoint: **reconstruct the geometry from a unique sketch.** As we are only using one view of our objects, the obvious problems due to self-occlusion will influence the amount of detail we can achieve with this approach. However for applications such as rough exploration of shapes, automatic shading or previsualization, the system is very efficient since the user need not rotate the viewpoint between different operations.

The core problem we have can be stated as follows: every line drawn on the paper is the 2D projection of an unknown 3D curve. From the pioneer work Teddy [IMT99] to more recent work on sketch-based modeling, for example [NISA07], the traditional way of dealing with this problem is to consider that the user is drawing on a plane, and that the user is expected to change the viewpoint and the virtual drawing plane along to draw strokes in various orientations. We strongly argue that this process is not natural for people used to draw on paper, as well as restricting the range of shapes that can be drawn with a small number of strokes. The main limitation is that the silhouette of shapes

† alexis@csl.sony.co.jp, suguru@img.cs.titech.ac.jp

SBIM 2011, Vancouver, British Columbia, Canada, August 5 – 7, 2011.
© 2011 ACM 978-1-4503-0906-6/11/0008 $10.00

DOI: 10.2312/SBM/SBM11/133-140

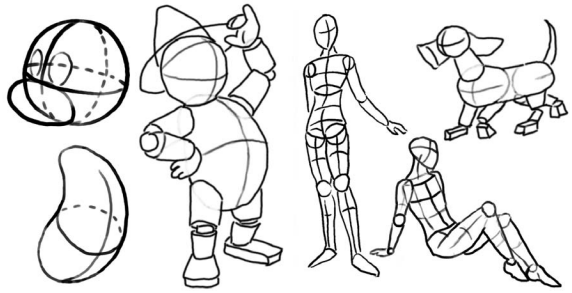


Figure 1: *Inspiration for this work. Any complex object can be decomposed into a set of simple geometric shapes, that are drawn with some profile curves or construction lines. Inspired from Preston Blair.*

has to lie on one plane, a very restricting assumption. In our system, we allow the user to draw strokes outside the plane perpendicular to the view direction, to closely match the way people draw on paper.

We are mainly interested in a particular drawing method used by many cartoon artists, summarized in a book by Preston Blair [Bla03], and depicted in Figure 1. The idea is that all shapes can be divided into simple geometric primitives, and that construction lines (lines that define the primitives, called perspective guidelines in [Bla03] or sometimes profile curves) help a lot to understand the shape. Our method follows this approach: we combine simple primitives whose shape is determined by construction lines and an optional silhouette until we get the desired complex shape, and **everything is performed from a unique viewpoint**. The construction lines we consider in this paper are planar lines that are orthogonal to each other: this allows us to determine the orientation of the planes the strokes lie on, thus to reconstruct the original strokes in 3D space.

This drawing style was also investigated in terms of a rendering target for completed 3D models inside a silhouette-based modeling tool [SIJ*07] in order to create a "sketchy" feeling for the models. We on the contrary start from such sketches to reconstruct the 3D geometry.

The most important difference in our approach is that we do not require the user to think in terms of a modeling software with various operations that make sense for the computer such as cut or extrude for example. We based our system on **existing drawing methods**, with a fixed view-point, in order to create directly what the user was thinking.

The contributions of this paper are as follows:

- a reconstruction method for simple shapes based on orthogonality,
- an interface for editing such shapes in a fixed view setting,
- an interface for combining simple parts into more complex objects, from a unique viewpoint.

The paper starts with a description of related work then explains the method of reconstruction for one basic geometric primitive, using two construction lines and an optional silhouette, as well as the required interface when the system needs the user's guidance to reconstruct the shape. How to combine simple parts to create complex objects is then discussed. Finally, results and discussion are presented.

The results of this paper allow artists to draw as they would have done on paper or to trace over an existing sketch and to get the corresponding 3D model directly in the correct orientation.

2. Related work

Traditional sketch modeling systems such as Teddy [IMT99] employ the sketch-rotate-sketch paradigm to generate shapes. The main difference with common modeling tools lies in the use of a silhouette to generate the basic shape. The inflation method used to construct the basic geometry from one stroke varies from system to system, see for example [KH06, SWSJ05, TZF04].

Some sketching interfaces use additional information such as shade and shadow to reconstruct the object with more details, for example [SLKM04], where the amount of shade is used to inflate more or less the volume. Narrower shadows result in more elongated shapes. While this previous work is also using cross section lines (we call them construction lines in this work), our approach is different and our method does not require additional information to reconstruct the correct shape.

Using a single 2D sketch as the input of a 3D modeling system has been of course the topic of various research. In [IOI06], the initial sketch is first decomposed into billboards forming a hierarchical structure similar to the parts decomposition we are taking in this paper. Their method is however focused on floral elements, with the use of pre-existing components that are mapped to the billboards. We on the contrary allow any shape to be created, as long as the artist is able to draw its construction line.

Another approach is to construct the shape once the user has finished his drawing. 3D Sketch [MSK00] takes the sketch of an object, maps the strokes to the edges of a cube and inflates the corresponding cubic form while preserving the style of the strokes. The system provides a nice interface to sketch a whole range of objects, as long as they can be mapped to a cube. Such limitation makes the strength and the weakness of their system. We want a system able to model any shape, without any constraint.

A single-view 3D modeling system has also been proposed by Gingolg et al. [GIZ09], where the users were able to reconstruct 3D shapes by tracing over existing sketches. While the target is similar, their approach is a refining process, starting from basic shapes then progressively editing

them until they satisfy the users' view. On the contrary we focus on a system where almost all interactions are drawing actions.

In [SKSK09], artists are able to draw curves in 3D from a single viewpoint when drawn in a particular style called *analytic drawing*. The scaffolds allow the system to infer the depth of all curves, however this requires the user to correctly draw the scaffold before drawing the actual lines. Our approach is based on a rather different drawing style, where less strokes are needed to generate simpler yet complete shapes.

In our system, we use extrusion from construction lines to reconstruct the shapes, a common operation in modeling. We can then find systems that rely heavily on extrusion, such as [LG94, OSD06, Ske]. However, none of those systems focused on the reconstruction of a sketch drawn from a single viewpoint.

Our system tries to fill the void between single-view reconstruction and sketch-based modeling by providing a tool that is able to generate complex models by combining small shapes, easy to draw, from a unique viewpoint. For more related work, we refer the reader to [CPCN05].

3. Single Part Generation

In this section, we will focus on the reconstruction of a single part, drawn with two construction lines and an optional silhouette.

3.1. Framework

The system consists of a canvas where the user draws with a 2D input device (a pen tablet was used for all the examples in this paper) and some clickable buttons available to the right for various operations that are not sketch-oriented (shifting or copying strokes for example). We use an orthogonal projection so the problem is now how to find the relative depth of every stroke on the drawing, where all drawing operations are done from a unique viewpoint.

The system also presents the user with a side view in one corner of the drawing plane after each operation that results in a change of the global shape, so that the user can immediately see the effects of his last operation. This greatly reduces the need to change the general viewpoint. The system has two rendering modes: the user draws strokes in the 2D mode, and can switch to the 3D mode to get a preview of the model (with flat shading) at any time by holding a button on the input device (right button on the mouse, or one button on the stylus of a pen tablet). The companion video shows examples of interaction with the system.

3.2. User input

The user draws ordered sets of 3D points (the coordinates in the drawing plane and time) that we consider as strokes for our system. As our method is able to deal with sharp features in the input strokes, we preprocess the strokes to remove input noise while keeping the corners. We beautify the user input by detecting corners at the maximum of curvature and minimum of drawing speed, and by smoothing the segments between the corners [SSD01].

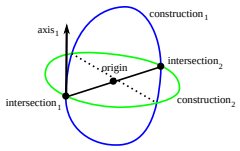


Figure 2: *The naming conventions used in this paper. For the two colored construction lines, there exist two possible intersection lines shown in black (solid and dotted lines). The solid line represents the chosen possibility.*

We consider for now that the user is drawing three strokes: a silhouette, and two construction lines. We first identify the silhouette by computing the coverage each stroke has on the two others (the ratio between the number of points located inside one stroke and the total number of points of that stroke). The silhouette is the stroke than encloses the most the two other ones.

The strokes required by our system are not different from what artists draw in the early stages of a sketch. Two construction lines have however four intersection points: in the mind of the user, the orientation of the strokes in space is established, resulting in two real intersection points, and two fake points due to the projection. Figure 2 shows an example and gives the terminology used throughout this paper.

When the user has drawn two construction lines (and maybe a silhouette), he needs to specify which intersection point is a true intersection point by clicking on one of the four points, giving at the same time an order to the system to reconstruct the shape from the available information (i.e. the system shall reconstruct a part whether the user has drawn a silhouette or not). This simple step strays however from pure shape depiction.

3.3. Cubic Corner

We now have two construction lines on the virtual canvas, and we want to orient them in the 3D space. The properties of the construction lines we are considering in this paper can be reformulated: at one real intersection point, the two construction lines' tangents plus the intersection line form the projection of the corner of a cube, whose reverse projection is known [Per68] under orthographic projection. We can not however expect the user to draw strokes that are exactly perpendicular in space, so we consider that the strokes are perpendicular at the intersection point specified by the user.

In fact, the orientation of the three strokes in space is

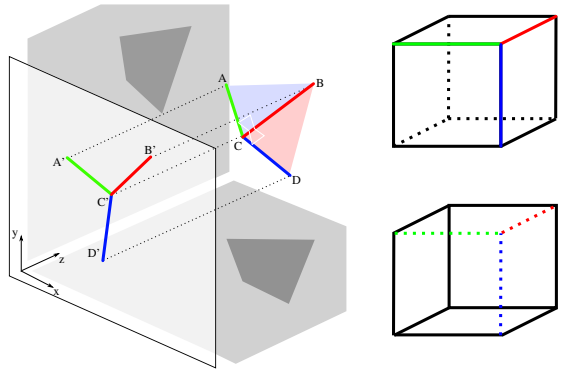


Figure 3: *Cubic corner. The reverse projection of three orthogonal vectors is known, with respect to Necker's inversion shown on the right: the same angles on the drawing plane are the projections of two different corners (depending on which face of the cube is facing the user). The two possible choices consist of one corner and its mirror image along the projection direction.*

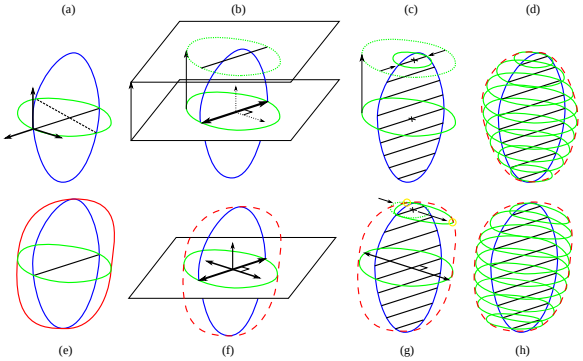


Figure 4: *Reconstruction algorithm. (a) input: two construction lines in green and blue, with the two possible intersection lines in black. The three arrows form a cubic corner. (b) Sweeping of one construction line along the other one. Without the silhouette, the scaling factor corresponds to the length of the intersection line. (c) The scaling is performed in the corresponding plane, on both axis with the same factor. (d) Resulting shape, with the resulting silhouette shown in red. (e) User input: two construction lines and a silhouette in red. (f) Reconstruction with a silhouette. Similar to (b), we first scale the stroke using the length of the intersection line, but we use the two axis of the corresponding plane this time. (g) We scale the stroke in the direction perpendicular to the intersection line until it gets tangential to the silhouette at two and only two points (yellow circles). (h) The resulting shape corresponds to the given silhouette and two construction lines.*

a function of the relative angles between the projected lines [Per68]. With the notation of Figure 3, we have:

$$\theta = \sqrt{\cot A'\widehat{C'D'}\cot B'\widehat{C'D'}} \quad (1)$$

$$z_C = z_D \pm L_{C'D'}\tan(\arcsin(\theta)), \quad (2)$$

where z_C represents the z-coordinate of the point C (z_D for D), $L_{C'D'}$ is the length of the segment $C'D'$. The two possible results come from *Necker*'s reversion (see Figure 3). This reverse projection of the corner of a cube has already been applied to sketch-based modeling [ASN07], and for the reconstruction of CAD drawings [VTMS04]. The reverse projection of sets of angles has also been studied under perspective projection in the depth-from-image field, for example [WK01].

Once we determine the orientation of the planes the construction lines are lying on, we project the drawn lines back to the 3D space. The formula gives however two possible choices: the system presents the user with the first result, and if the user disagrees, he needs to click again on the intersection point to ask for the other possible result.

3.4. Reconstruction by sweeping

We now have the two construction lines in 3D, but the depth along the silhouette (if it has been drawn) is still unknown (see Figure 4). We now distinguish the construction lines into the first and second construction lines, the first one being the one with the biggest diameter (the user can specifically ask to change the sweeping direction by dragging in the desired direction when selecting the true intersection point). We reconstruct the shape by sweeping the first construction line along the direction of the second one, with some scaling so that it matches the following conditions:

1. the resulting silhouette matches the user input silhouette, and
2. the second construction line intersects the swept lines.

Since the construction line we are sweeping is planar, we have two dimensions available to scale it. We define the first one as the direction of the intersection line, and this dimension is used to satisfy condition 1. The direction perpendicular to that one is used to satisfy condition 2, when the user has drawn a silhouette. In the absence of silhouette, the scaling factor needed to satisfy condition 1 is also applied to the second dimension.

Since we consider that the silhouette is not lying on a plane, and since the silhouette is dependent on the view direction, we need to find the scaling factors (one for each side of the intersection line, for each extrusion step) that make the new line tangential to the silhouette direction in one and only point for each side. We solve this problem using a binary search, computing the number of intersection points between the silhouette and the new line.

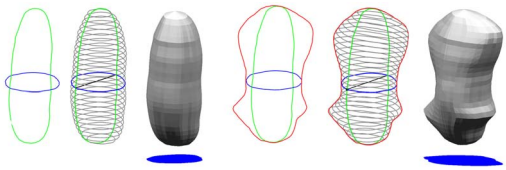


Figure 5: *Two simple shapes generated with our system. The construction lines for both shapes are the same, but the shape on the right has a silhouette that changes the scaling during the extrusion. From left to right, we have: user input strokes, extruded construction lines with the intersection line shown in black, and final shape with flat shading and the pseudo shadow on the bottom.*

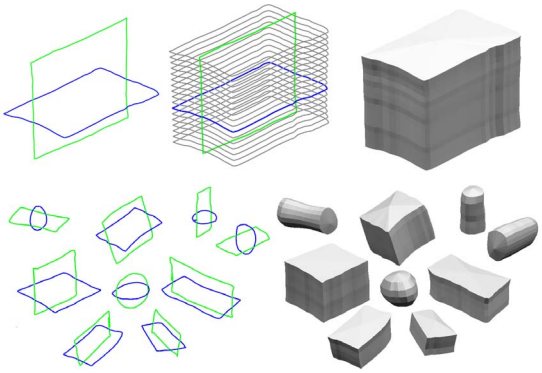


Figure 6: *Simple shapes that show the possibilities of our system. From a simple cube (top row), we can directly orient the shapes as shown in the middle row.*

Figure 5 shows the resulting shape for simple construction lines and the influence of the silhouette on the scaling process. Figure 6 shows how it is possible to orient directly various shapes in space by drawing the construction lines in the correct position.

The use of the silhouette was motivated by two reasons: the first one was to mimic the drawing style described in the introduction, and the second one was that it is difficult to draw thin shapes with accuracy with only the construction lines, as shown in Figure 7. With a silhouette, the shapes can get as thin as the user wants, as shown in Figure 8.

4. Interactions and Multiple Parts

4.1. Operations on strokes

While oversketching (or redrawing) is obvious when using pen and paper, the first sketch-based modeling systems were not considering this feature as important. We had to wait for more recent systems to include a way to redraw the shapes, for example in [SIJ*07]. For our system, as we are considering only strokes that define shapes, we just need to recreate the shapes using the new information.

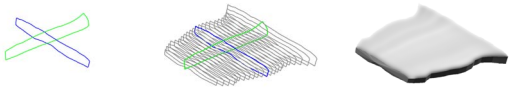


Figure 7: *Typical failure when trying to model a square plate, as the length of the intersection line scales the construction line in two dimensions so a small variation in width (globally small but relatively important) results in unwanted changes in the other dimension.*

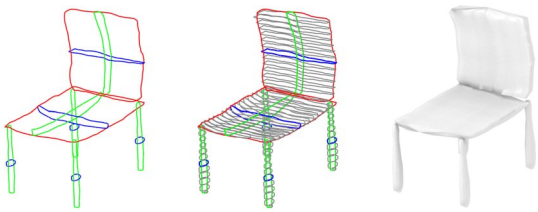
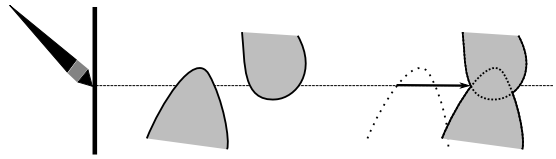


Figure 8: *Using a silhouette, the user can however draw straight and thin parts, whose inner shape is defined by construction lines. Here, a chair, whose back and seat are drawn using silhouettes on top of the construction lines.*

The user is then able to redraw a stroke locally, and the changes to the resulting shapes are immediately shown on the display. The implementation is straightforward: when the user is drawing a stroke on the canvas and when it is not a closed loop (otherwise we start a new part), we find the closest existing stroke to this newly input correction stroke, we replace the corresponding part of the original input points by the new ones and we perform the smoothing again. The discarded input points are stored to show exploration history, in a similar way to [SIJ*07]. See the video for examples.

The resulting feature is very important from the point of view of usability. Allowing the user to redraw some parts of the shape he is working on opens the gate for exploration of new shapes: exactly how a designer would draw more than one stroke on paper before being satisfied. Moreover, the user is always sketching strokes to create shapes. A very different approach was taken in [NISA07], where some strokes drawn on the surface of the object were pulled to modify the global shape. We argue that our approach better fits the pencil-and-paper workflow.

Aside from the oversketching feature, we added some basic operations to our system, in order to assist the user with common tasks. We implemented a copy feature able to copy strokes over the canvas, a shift feature that allows to move strokes on the drawing plane. Finally, we allowed the user to scale one part of the object to help him draw small parts, first using bigger strokes on the side of the canvas then scaling them down and shifting them to the correct position.



(a) (b)

Figure 9: *Depth setting for two parts. At first, the two parts are disconnected (a), but when the user specifies the junction, the newer part of the two is shifted to create the junction at the desired position (b).*

4.2. Relative Depth for Multiple Parts

The creation of objects with multiple parts raises the question of the relative depth. When the user has drawn two parts, in our orthographic projection, we have no clue on the depth difference between those two parts.

We ask the user to position the parts in space, with the help of the following operations. In the 3D display mode, the system shows a pseudo shadow of the current parts on the bottom part of the screen. The user is able to drag the shadow to specify the depth of that part, similar to what has been done in [ZHH96].

One other available operation is to specify the junction line between two parts, more specifically a point where two parts overlap. The depth of the two parts at the specified point should then be the same for the junction line to pass through this point. For implementation purposes, as we need to decide which part of the two we ought to translate, we store for each part the last time its depth was changed, and we translate the part whose depth is the oldest, to allow to set depths for objects with more than two parts. The method is shown graphically in Figure 9.

5. Results and Discussion

We now present objects sketched in our system, in order to show the range of shapes that can be modeled. Sharp features can be seen in most examples, as the system uses extrusion to reconstruct shapes. Systems using inflation from a single silhouette can not produce similar features directly: most of them require additional interaction.

Simple results can be seen in Figure 11. The gears are a simple example showing the benefit of not having to draw of not the silhouette of the objects. While drawing the construction lines for such models is natural, thinking in terms of silhouette is more challenging, and as far as we know, no system can reconstruct gears from their silhouette without additional information or operations. The models are well suited to generate previews of a more complex scenes in a production setting for example: using the system to create rough shapes of all the models in a scene allows artists to work on the composition of this particular scene while the final models are created.

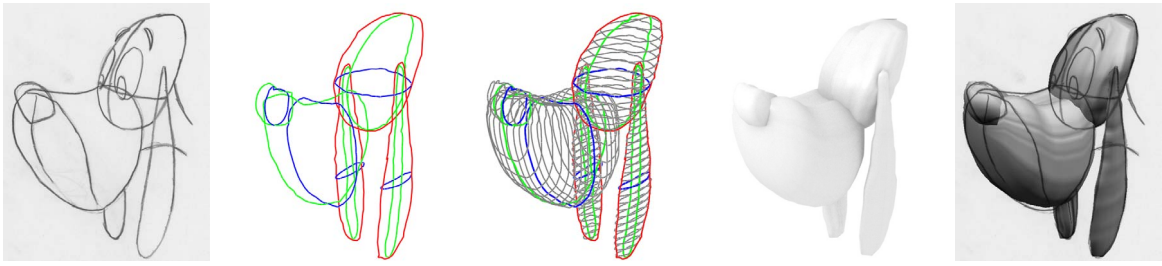
Since our first motivation was to allow users to interact with the system in the same way as they would draw objects on paper, it is possible to trace over existing sketches to instantly add lighting and shade effects on the geometry. One example is given in Figure 10.

Figure 12 shows complex models designed in our system, making use of the various possibilities of our system. While it is possible to create similar objects with other systems, we argue that our system is the closest to the actual drawing process, as we do not rotate the viewpoint, and the strokes drawn are actually close to what an artist would have drawn. Moreover, the steps needed to create such models in other systems are far greater than what our system requires. The objects in Figure 12 were created in our system by a trained user in 15 minutes (top), 5 minutes (middle) and 15 minutes (bottom). A silhouette stroke was used for some of the parts (stroke in red). Sharp features can be seen in the beard, or on the top of the spray. A complex animated scene made with various simple objects modeled with this system can be found in the companion video.

The important points to take into consideration is first that the viewpoint is fixed during the complete modeling, so all operations are performed from that unique viewpoint. Second, all strokes drawn on our virtual canvas have their equivalent when drawing on paper. Even if the style of drawing we are expecting is different from the norm in sketch-based modeling systems such as [IMT99], we argue that our approach is more suited to the trained artist (trained in a way similar to that of [Bla03]) than the beginner, and as such, is not requiring our target audience to relearn a new way to draw. Moreover, construction lines have already been used in various sketch-based systems in the form of profile curves [LGS06, SLKM04, CSSJ05], but our system is, as far as we know, the first one to apply them in a single view setting.

While the resulting models offer a level of quality that matches other sketch-based modeling systems without having to change the viewpoint between operations, self-occlusion is sometimes difficult to cope with. For the applications we are targeting in this paper, such as adding shading or rapid prototyping, the system is however able to produce enough 3D information to create the desired effect.

Another point open for discussion is the choice of our extrusion method to reconstruct one part. While the chosen method is easy to implement, not computationally expensive (we do not optimize any energy function) and produces acceptable results, interpolating multiple 3D strokes is an active research field, with various methods proposed, for example [TO99]. Another advantage of the extrusion method used in this paper is that the models created are “nice”, to quote a professional modeler when presented with the system, and that “they could have been done by hand”. Moreover, the



(a) (b) (c) (d) (e)

Figure 10: *Since all operations are performed from a unique viewpoint, we can trace over an existing sketch (a) to create 3D shapes. (b) Input lines. (c) Extruded lines. (d) Ambient occluded rendering. (e) With the original sketch as a transparent layer.*

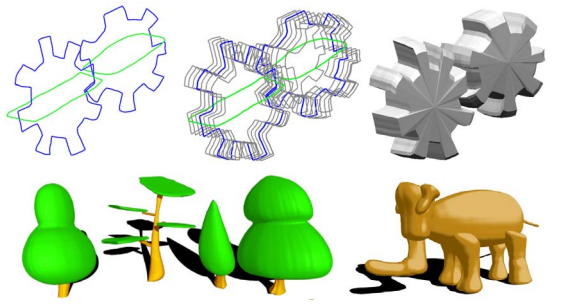


Figure 11: *Top: Some gears: while the sketchy nature of the input results in rough shapes, systems based on inflation methods can not produce such models. Bottom: Applying basic materials to the models allows for quick generation of various models.*

models are easy to edit due to their inherent extruded structure.

Of course, our system presents some limitations due to the drawing style we are expecting. First, it is impossible to create holes in a part. While it is possible to create an enclosing shape (similar to a donut) with a lot of parts, it is troublesome and not intuitive. Second, the system can not cope with construction lines that present complex profiles such as forks or huge variation in orientation as the extrusion method fails for those cases. Parts that split into two or more branches can however be created by drawing separately each branch. While this might sound restrictive, we argue that it fits better our approach of drawing simple primitives. The relative depth setting is also limited since it cannot cope with the cases where disconnect parts are wanted or where the junction between parts is occluded. For locations where three or more parts overlap, it is also difficult to target the correct part. In those cases, it is then difficult to specify where the junction should be between two of them. While it is possible to drag the shadows to overcome these problems, we would rather have a constant way of interacting with the system.

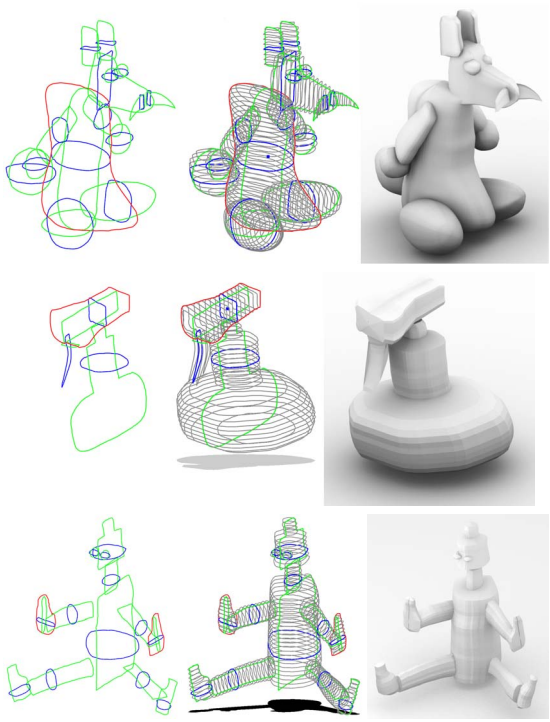


Figure 12: *Modeling results using our system. All operations were performed from the same viewpoint. For each model, we show the input lines, the extruded geometry automatically generated, and a shaded rendering using ambient occlusion.*

6. Conclusion

We proposed a new system that is able to produce complex objects from strokes drawn from a unique viewpoint. Each object is decomposed into smaller primitives that can be specified by construction lines. In our system, two or three strokes are enough to generate one of such primitives. Moreover, the strokes the system is expecting are no different from what artists are familiar with drawing: silhouettes,

and construction lines. While some interaction is needed to remove ambiguities or to specify the relative depths, the system produces 3D reconstructions of 2D sketches in a natural way.

We show with this system that single-view input is enough to create a wide variety of shapes. By providing a direct bridge between 3D models and 2D sketches, we propose a interactive system to explore shapes, and to design objects directly at the model level, while only considering drawing operations as input.

We are now considering the various challenges raised by this system. We hope to combine this approach with ways to add detail to shapes, again in a single-view setting, for example in a similar way to [ASN07]. We are also interested in ways to create smooth junctions between the parts while keeping the system simple and authentic to the original strokes.

References

[ASN07] ANDRE A., SAITO S., NAKAJIMA M.: Crosssketch: Freeform surface modeling with details. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2007), pp. 50–57.

[Bla03] BLAIR P. J.: *Animation 1: Learn to Animate Cartoons Step by Step*. Walter Foster, January 2003.

[CPCN05] COMPANY P., PIQUER A., CONTERO M., NAYA F.: A survey on geometrical reconstruction as a core technology to sketch-based modeling. *Computers & Graphics 29*, 6 (2005), 892–904.

[CSSJ05] CHERLIN J. J., SAMAVATI F., SOUSA M. C., JORGE J. A.: Sketch-based modeling with few strokes. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics* (New York, NY, USA, 2005), ACM Press, pp. 137–145.

[GIZ09] GINGOLD Y., IGARASHI T., ZORIN D.: Structured annotations for 2d-to-3d modeling. *ACM Trans. Graph. 28* (December 2009), 148:1–148:9.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 409–416.

[IOI06] IJIRI T., OWADA S., IGARASHI T.: Seamless integration of initial sketching and subsequent detail editing in flower modeling. *Comput. Graph. Forum 25*, 3 (2006), 617–624.

[KH06] KARPENKO O. A., HUGHES J. F.: Smoothsketch: 3d free-form shapes from complex sketches. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 589–598.

[LG94] LIANG J., GREEN M.: JDCAD: A highly interactive 3D modeling system. In *Computer and Graphics* (1994), vol. 18(4), pp. 499–506.

[LGS06] LEVET F., GRANIER X., SCHLICK C.: 3d sketching with profile curves. In *International Symposium on Smart Graphics* (2006), vol. 4073 of *Lecture Notes on Computer Science*.

[MSK00] MITANI J., SUZUKI H., KIMURA F.: 3d sketch: Sketch-based model reconstruction and rendering. In *IFIP Workshop Series on Geometric Modeling: Fundamentals and Applications, 7th Workshop GEO-7* (2000), pp. 85–112.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: designing freeform surfaces with 3d curves. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 41.

[OSD06] OH J.-Y., STUERZLINGER W., DANAHY J.: Sesame: towards better 3d conceptual design systems. In *DIS '06: Proceedings of the 6th conference on Designing Interactive systems* (New York, NY, USA, 2006), ACM, pp. 80–89.

[Per68] PERKINS D.: Cubic corners. *Quarterly Progress Report*, 89 (1968), MIT Research Laboratory of Electronics, 207–214.

[PG98] PURCELL A., GERO J.: Drawings and the design process: A review of protocol studies in design and other disciplines and related research in cognitive psychology. *Design Studies 19*, 4 (1998), 389 – 430.

[SIJ*07] SCHMIDT R., ISENBERG T., JEPP P., SINGH K., WYVILL B.: Sketching, scaffolding, and inking: A visual history for interactive 3d modeling. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering* (2007), pp. 23–32.

[Ske] SKETCHUP: Google sketchup software, from Google Inc., http://sketchup.google.com.

[SKSK09] SCHMIDT R., KHAN A., SINGH K., KURTENBACH G.: Analytic drawing of 3d scaffolds. *ACM Trans. Graph. 28* (December 2009), 149:1–149:10.

[SLKM04] SHIZUKA H., LIU W., KONDO K., MATSUDA K.: A sketch interpreter system with shading and cross section lines by freehand drawing. In *Proceedings of the 11th International Conference on Geometry and Graphics (ICGG2004)* (8 2004), pp. 357–362.

[SSD01] SEZGIN T. M., STAHOVICH T., DAVIS R.: Sketch based interfaces: Early processing for sketch understanding. *Workshop on Perceptive User Interfaces, Orlando FL* (2001).

[SSLR96] SCHUMANN J., STROTHOTTE T., LASER S., RAAB A.: Assessing the effect of non-photorealistic rendered images in cad. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1996), ACM, pp. 35–41.

[SWSJ05] SCHMIDT R., WYVILL B., SOUSA M. C., JORGE J. A.: Shapeshop: Sketch-based solid modeling with blobtrees. In *Proceedings of the 2nd Eurographics workshop on Sketch-Based Interfaces and Modeling,* (8 2005), pp. 53–62.

[TO99] TURK G., O'BRIEN J. F.: Shape transformation using variational implicit functions. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 335–342.

[TZF04] TAI C.-L., ZHANG H., FONG J. C.-K.: Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces. *Computer Graphics Forum 23*, 1 (Mar. 2004), 71–83.

[VTMS04] VARLEY P. A. C., TAKAHASHI Y., MITANI J., SUZUKI H.: A two-stage approach for interpreting line drawings of curved objects. In *ed. J. F. Hughes and J. A. Jorge, Sketch-Based Interfaces and Modelling, Eurographics Symposium Proceedings* (2004), pp. 117–126.

[WK01] WONG K. C., KITTLER J.: Recognition of polyhedral objects using triplets of projected spatial edges based on a single perspective image. *Pattern Recognition 34*, 3 (2001), 561 – 586.

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: Sketch: an interface for sketching 3d scenes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 163–170.