Sketch-based Modeling with Few Strokes

Joseph Jacob Cherlin [†] Faramarz Samavati [†] Mario Costa Sousa [†] Joaquim A. Jorge [‡]

[†] University of Calgary
{cherlin,samavati,mario}@cpsc.ucalgary.ca

[‡] Technical University of Lisbon
jaj@inesc-id.pt

Abstract

We present a novel sketch-based system for the interactive modeling of a variety of free-form 3D objects using just a few strokes. Our technique is inspired by the traditional illustration strategy for depicting 3D forms where the basic geometric forms of the subjects are identified, sketched and progressively refined using few key strokes. We introduce two parametric surfaces, rotational and cross sectional blending, that are inspired by this illustration technique. We also describe orthogonal deformation and cross sectional oversketching as editing tools to complement our modeling techniques. Examples with models ranging from cartoon style to botanical illustration demonstrate the capabilities of our system.

CR Categories: I.3.5 Computer Graphics Geometric algorithms, languages, and systems I.3.6 Computer Graphics Interaction techniques

Keywords: sketch-based interfaces/modeling, free-form surfaces

1 Introduction

In traditional illustration, the depiction of 3D forms is usually achieved by a series of drawing steps using few strokes. The artist initially draws the outline of the subject to depict its basic masses and boundaries. This initial outline is known as *constructive curves* and usually results in very simple geometric forms. Outline details and internal lines are then progressively added to suggest features such as curvatures, wrinkles, slopes, folds, etc [Dease et al. 1999; Goldstein 1999; Guptill 1977]. Three examples of methods typically used to achieve such drawing progression are shown in Figure 1:

The **spiral method**, where shape depiction is achieved by the use of quickly formed spiral strokes connecting the constructive curves, creating a visual "blend" of the overall volume between the constructive curves. Spiral strokes are helpful when irregular rounded forms are involved, such as fruits, vegetables, animals, and people.

The **scribble method**, involves the use of continuous stroke(s) placed between constructive curves. The scribbled strokes are typically used to depict specific folds, bumps, etc., across the subject. In Figure 1, a single scribbled stroke defines the fold pattern at the boundary end of the skirt.

The **bending (or distortion) method**, adds unique variations to the initial sketch of the subject. These variations aid on depicting the overall shape of subjects which naturally present a large variety of twists, turns, and growth patterns, such as botanical and anatomical parts.

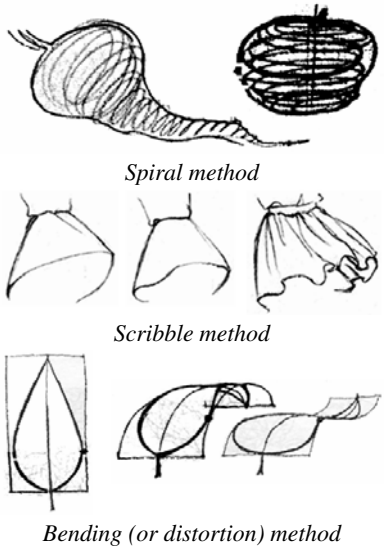

Figure 1: Traditional hand-drawn techniques for progressive shape depiction [Dease et al. 1999; Goldstein 1999; Guptill 1977]. Skirt drawing, Copyright 1998-2004 Rio Aucena. Used with permission.

In this paper we present a sketch-based modeling system inspired by the three traditional illustration methods above. We have developed new algorithms to facilitate the rapid modeling of a wide variety of free-form 3D objects, constructed and edited from just a few freely sketched 2D line segments, without imposing any constraints regarding the order in which lines are entered as well as their spatial relationship.

Our system takes as input strokes sketched by the user using a mouse or tablet. Each stroke is then captured and properly filtered (Section 3) to allow efficient and robust modeling of 3D objects, represented as parametric surfaces, in two phases, creation and editing. The techniques we developed, at each phase, were inspired by the traditional shape sketching methods of spiral, scribble and bending (Figure 1). In the *creation phase* (Section 4), surfaces can be generated using techniques for two types of parametric surfaces, rotational blending surface (Section 4.1) – approximating the spiral method –, and cross sectional blending surface (Section 4.2) – approximating the scribble method. In the *editing phase* (Section 5), subtle or drastic variations to the surfaces can be added by using a single deformation stroke (Section 5.1)– approximating the bending method. Surfaces can also be modified by oversketching cross-sections of the model (Section 5.2) – also related to the scribble method.

The rest of this paper is organized as follows: related research is reviewed in Section 2. Details of our approach are provided in Sections 3, 4 and 5. Results are discussed in Section 6, and conclusions presented in Section 7. An appendix is also provided with mathematical details of the techniques presented in Sections 4 and 5.

2 Related Work

Sketch-based systems are a relatively new area in modeling, in particular for 3D content creation of free-form objects in design, production and science. The main goal of sketch-based systems is to allow the creation of 3D models by using strokes extracted from user input and/or existing drawing scans. Refer to [Naya et al. 2002] for a complete classification of sketch-based systems. Our system fits in the sketch-based category of *gestural modeling* where hand gestures are used as commands for generating and editing 3D shapes from 2D line segments.

SKETCH [Zeleznik et al. 1996] combines mouse gestures and simple geometric recognition to create and modify 3D models. To this end, SKETCH uses a gesture grammar to create simple extrusion-like primitives in orthogonal view. It is also possible to specify CSG operations and defining quasi- free-form shapes such as ducts in a limited manner.

Quick-Sketch [Eggli et al. 1997] is based on parametric surfaces. The system creates extrusion primitives from sketched curves, which are segmented into line and circle primitives with the help of constraints. They also consider surfaces of revolution and ruled surfaces for creating more free-form shapes. In all cases, a combination of line segments, arcs and B-Spline curves are used for strokes. Although this system can be used for sketching engineering parts and some simple free-form objects, it is hard to sketch more complicated free-form objects such as in Figure 1.

Teddy [Igarashi et al. 1999] is a sketch based system that allows the user to easily create free-form 3D models. The system allows creating a surface, by inflating regions defined by closed strokes. Strokes are inflated, using *chordal axis* transform, so that portions of the mesh are elevated based on their distance from the stroke's chordal axis. Teddy also allows users to create extrusions, pockets and cuts to edit the models in quite flexible ways. Teddy's main limitation lies in that it is not possible to introduce sharp features or creases directly on the models except through *cuts*. Furthermore, the system only allows editing a single object at a time.

Owada et al. [2003] proposed a sketch-based interface similar to Teddy that can model 3D solid objects and their internal structures. Sketch-based operations similar to those in Teddy are used to define volume data. The authors take advantage of spatially-enumerated representation for performing volume editing operations including extrude and sweep. Extruding connects a volumetric surface to a new branch, or to punch holes through the surface. Sweep allows creating a second surface on top of the original. This is accomplished by drawing the cross section of where the two surfaces are to meet, and a sweep path to define the place of the second surface. Using an ingenious command to hide portions of the surface, the system makes it possible to "see inside" the surface. By hiding portions of a model, and then using extrusions, the user can specify hollow regions inside an object. However, this system is also not suitable for editing sharp features or creases.

Karpenko et al. [2002] use *variational implicit surfaces* [Turk and O'Brien 1999] for modeling blobs. They organize the scene in a tree hierarchy thus allowing users to edit more than one object at a time. Also, their system allows constrained move operations between tree nodes. Another interesting feature is using guidance strokes for merging shapes. Like Teddy, this system is not clearly suited to editing sharp features or creases into objects.

BlobMaker [de Araujo and Jorge 2003] also uses variational implicit surfaces as a geometrical representation for free-form shapes. Shapes are created and manipulated using sketches on a perspective or parallel view. The main operations are inflate, which creates 3D forms from a 2D stroke, merge which creates a 3D shape from two implicit surface primitives, and oversketch which allows redefining shapes by using a single stroke to change their boundaries or to modify a surface by an implicit extrusion. This system improves on Karpenko's and Igarashi's by performing inflation independently of screen coordinates and a better approach to merging blobs. Like other systems previously reviewed, BlobMaker does not provide tools to create sharp features.

Ijiri et al. [2004] present a sketch–based system for specialized editing of leaf-like objects combining free-form modeling with bending operations. However, the interface is limited to model floral features such as leaves or petals using specific interaction idioms.

Duncan and Swain [2004] present SketchPose, a system that allows designers to quickly sketch control points using a pen. Like our approach, they have derived novel techniques drawn from conventional pencil-and-paper cartooning methods. Moreover, their technique also stresses sketching on the view plane. This greatly simplifies positioning and deforming objects, thus expediting the definition of poses for animated characters.

Varley et al. [2000] present a method to generate a 3D mesh based upon user-input strokes. Their method assumes that the output mesh is geometrically similar to a pre-defined template. The camera position and orientation are estimated based upon the spatial layout of the strokes and the template. The authors discuss various methods of extracting and reconstructing meshes using the camera and stroke information. They again use the template to determine where each of these reconstruction methods is applicable. The mesh is ultimately constructed as a collection of Coon's patches or by a method of B-Rep reconstruction using similar stroke data. Varley et al. also present a novel stroke capturing algorithm which they use in both their B-Rep and 3D mesh methods. This algorithm allows the user to draw strokes in a style similar to the one many artists and engineers use when they sketch on paper. In this style, many shorter sub-strokes are used to compose each stroke. They refer to the group of sub-strokes as a "bundle of strokes." Their algorithm interprets each "bundle of strokes" as a single stroke. In their method, only intersecting strokes are considered to be part of the same bundle.

There are several commercial packages which use sketching for geometric modeling. SketchUp [2005] is a direct manipulation package with a very well thought-out interface that allows architects to quickly sketch 3D drawings of buildings using plane faces and extrusion, with no support for curved surfaces. VRMesh [2005] is able to create triangular meshes from sketches using an inflation approach similar to Teddy. Curvy 3D [2004] models surfaces using 2D sketches.

Another approach for creating 3D objects and adding surface details is by painting with height, or depth [Curvy 3D 2004; Lawrence and Funkhouser 2003; Z-Brush 2005].

The systems we have reviewed fall roughly into three categories. **(1) Extrusion-based systems** such as Sketch, GIDeS [Pereira et al. 2004] and Quick-Sketch are able to create simple ideal solids and duct-like shapes, but are not suited for editing free-form objects. **(2) Blob editing systems** allow users to create soft blobby surfaces, but are not very good of creating blade-like shapes or patches. Finally, **(3) reconstruction-based systems** [Naya et al. 2002; Varley et al. 2000] are better suited for creating 3D solids from wire-frame drawings and thus better at creating mechanical engineering parts, but not free-form objects.

In contrast to previous approaches, our system provides the means to (1) generating a large variety of 3D parametric surface objects with curved and creased features. Furthermore, (2) few strokes are

required to create and edit the surfaces. Also, by following traditional methods gleaned from pen and paper, (3) we have devised simple interaction idioms to allow efficient and robust stroke capturing. Finally, (4) our boundary and surface editing paradigm is quite flexible to support the application of both subtle and drastic variations to instances of sketched objects. In the next sections we discuss the methods that support these contributions.

3 Stroke Capture

Stroke capture is the act of recording strokes from an input device such as a pen or mouse. It is necessary in any sketch-based system because strokes are the elementary data type of such systems. In contrast, other forms of modeling use points, polygons or density values as the defining units.

We want our strokes to be in parametric form, because this will naturally lead to parametric surfaces. Parametric surfaces are desirable for many reasons. We can evaluate them an arbitrary number of times to create a mesh with many or few polygons, as needed. Also, it is easy to generate texture coordinates for parametric surfaces. Finally, many parametric surfaces, including ours, can trivially be rendered as triangle strips. Stripped geometry can render about three times as quickly as its non-stripped counterpart.

In our system, sketched strokes start as raw data from the mouse or pen (Figure 2, left). This data is an ordered set of points. Although this raw data could be interpreted as a parametric curve, this interpretation causes three problems. First, the points are very noisy due to the shaky nature of handling the input devices. Second, the points are irregularly distributed along the drawing path due to variations in drawing speed. Third, there will be a very large number of points because the input device sends data many times per second.

Classical approaches to stroke filtering reduce the noise and complexity in separate steps. The first pass applies point reduction and dehooking while the second step uses line segment approximation [Douglas and Peucker 1973]. We chose to implement a single pass technique to yield a smooth and compact approximation to the input stroke using a simpler technique.

We would like to fit a B-Spline curve with a low number of control points to our stroke data. B-Splines have a guaranteed degree of continuity, which resolves the difficulty with noise. The problem of point distribution is easily solved with B-Splines because it is straightforward to evenly distribute points along the B-Spline by evaluating it at evenly spaced intervals.

To find the B-Spline curve, one may use least squares to obtain the optimal curve [Eggli et al. 1997; Samavati and Mahdavi-Amiri 2000]. However, even in the best case scenario, the least squares model must be converted to a linear system. In our application real-time feedback is essential and solving a system of linear equations is simply not fast enough. We have used reverse Chaikin subdivision to efficiently create a denoised B-Spline with evenly spaced control points [Bartels and Samavati 2000; Samavati and Bartels 2004].

A reverse subdivision scheme decomposes the fine resolution data to a coarse approximation and a set of details. These details usually show the high frequency information of the data. In our case, the high frequency data consist mostly of noise and can be discarded. Since Chaikin subdivision is based on a quadratic B-Spline, we can interpret the coarse information as control points of a quadratic B-Spline curve.

If we denote the fine points by $p_0, p_1 ... p_n$ to and the coarse points by $q_0, q_1 ... q_m$ then the general case of the reverse Chaikin scheme is:

$$q_j = -\frac{1}{4}p_{i-1} + \frac{3}{4}p_i + \frac{3}{4}p_{i+1} - \frac{1}{4}p_{i+2} \qquad (1)$$

where the step size of i is two. The cardinality of the coarse points is almost half that of the fine points. Notice that reverse Chaikin subdivision contains only very simple operations. This is the source of its efficiency.

Figure 2: Stroke capture: unfiltered stroke (left), after applying the reverse Chaikin filter (middle) and the final stroke showing its control points (right).

Each time the reverse subdivision is applied to the control points, the resulting curve becomes smoother, although it deviates further from the original stroke. We have found in our experiments that running the subdivision three times provides sufficient denoising while the deviation from the input stroke is not noticeable. Figure 2 shows this process.

4 Creation Phase

As we discussed in Section 1, sketching few numbers of strokes is a natural way of making surfaces. There are several types of surfaces, called *common surfaces*, that are defined based on a very small number of simple curves. Surface of revolution, ruled surface, general cylinder and Coons surface are important examples of common surfaces. Although they have very simple and efficient parametric descriptions, they are not sufficient for modeling various objects.

Particularly, the surface of revolution is a perfect model from the sketch-based interaction point of view [Eggli et al. 1997]. The user just needs to draw a curve and identify an axis of rotation. Furthermore, extruded objects, as a special case of ruled surfaces, have been broadly used in sketch-based systems because again it can be easily defined by one stroke [Eggli et al. 1997]. However, the scope of these surfaces is too limited. It is possible to enlarge the modeling capacity by using free-form surfaces such as general B-Splines and NURBS, but this reduces the capability of making surfaces with "small" number of strokes.

It is possible to arrange the control points of a NURBS or similar patch using a sketch-based interface, but this would require many strokes to accomplish (at least one for each row of points in the patch). In this work, we introduce two kinds of new common surfaces that approximates the artistic description of objects (Section 1, Figure 1). Using these surfaces, we are able to model many objects around us with few numbers of strokes.

4.1 Rotational Blending Surface

This first technique of the creation phase was inspired by the traditional spiral method for preliminary sketching (Section 1, Figure 1). In this method, the artist depicts the basic 3D form of the subject by quickly sketching spirals or any ring-shaped curves to visually "blend" the 3D masses between the two constructive curves [Guptill 1977].

Our approach to approximate the traditional spiral method was to combine the surface of revolution and the ruled surface to find the parametric description of a rotational blending surface. Let $q_l(u)$ and $q_r(u)$ be the co-planner 2D curves (strokes) defined by the user (Section 3). We would like to use $q_l(u)$ and $q_r(u)$ as the constructive curves (outlined form) of the rotational blending surface. Let $\wp$ denote the plane of the curves and $c(u)$ be the curve formed by the midpoint of $q_l(u)$ and $q_r(u)$ at each u (Figure 3). Assume that $t_u(v)$, for fixed u, parameterizes the circle perpendicular to $\wp$ with the center $c(u)$ and passing through $q_l(u)$ and $q_r(u)$ at each u, as follows:

$$\begin{aligned} t_u(0) &= q_l(u) \\ t_u(\pi) &= q_r(u) \\ t_u(2\pi) &= q_l(u). \end{aligned}$$

Figure 3 illustrates how the curve $t_u(v)$ is generated from the constructive curves. The desired surface is formed by moving this circle along $c(u)$ by changing u

$$S(u,v) = t_u(v). \tag{2}$$

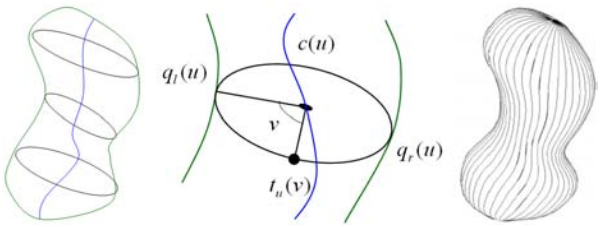

Figure 3: A rotational blending surface. The left and middle images show the constructive curves (green), $q_l(u)$ and $q_r(u)$, the center curve $c(u)$ (blue), and a circular slice of the surface, denoted $t_u(v)$. The right image shows a completed surface overlayed with the blending curves formed by holding v constant.

For fixed v and variable u, a set of rotational blending curves from $q_l(u)$ to $q_r(v)$ and vice versa are generated (Figure 3, middle). Note that surfaces of revolution can be generated by rotational blending surfaces; for this, the second curve should be a rotated version of the first curve. A more formal mathematical description for the construction of rotational blending surfaces is presented in Appendix A.

The rotational blending surface can create a variety of models, as shown in Figure 4. This shows a good flexibility in comparison with other common surfaces. In addition, as an important advantage, the surface follows the input strokes. This shows that the surface is acting in a predicted way and respects the user's intention. Furthermore, when the constructive curves have corner points or sharp feature, the final surface will also have sharp features and rotational creases, as shown in the candle in Figure 4.

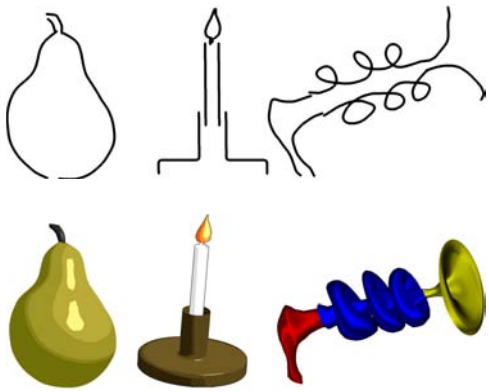

Figure 4: A variety of shapes is possible using few strokes (top row) by using rotational blending surfaces. We created the pear in four strokes, the candle in eight and the laser gun in six strokes.

4.2 Cross Sectional Blending Surfaces

This second technique of the creation phase was inspired by the traditional scribble method (Section 1, Figure 1). In this method, the artist adds details to the overall 3D shape of the subject by freely sketching one or few strokes within its outlined form (constructive curves) [Goldstein 1999; Guptill 1977]. Next, we will describe our approach to approximate the traditional scribble method.

Although the rotational blending surface is the default surface generator in our system, and it is more flexible than other common surfaces, still it can not make every free-form surface by its nature. In order to increase the flexibility of our surface generator as well as keeping the number of input strokes very small, we have introduced our second type of common surface. It is a simple modification of the rotational blending surface that allows the user to change the shape of the cross section from circle to an arbitrary 2D curve. Mathematically, this means that $t_u(v)$, for a fixed u, does not necessarily parameterize a circle anymore but the curve which is provided by the user (see Figures 5, 6). Again, the surface can be defined by changing u, or equivalently by moving $t_u(v)$ along $c(u)$. A more formal mathematical description for the construction of cross sectional blending surfaces is presented in Appendix A.

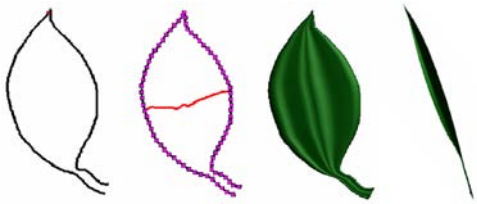

Figure 5: *From left to right*: sketching two constructive strokes (black), one cross sectional stroke (red) and the resulting leaf model in front and side views.

Figures 5 and 6 shows the model of a leaf and a sword blade, respectively, created using cross sectional blending surface. We may have given an option to the user to have control over several cross sections. However, we found this less interesting because it increases the number of strokes. In addition, when we have more than two sections we needed to use a kind of smooth transition between them and this is well fitted to free-form surface modeling with B-Spline and NURBS.

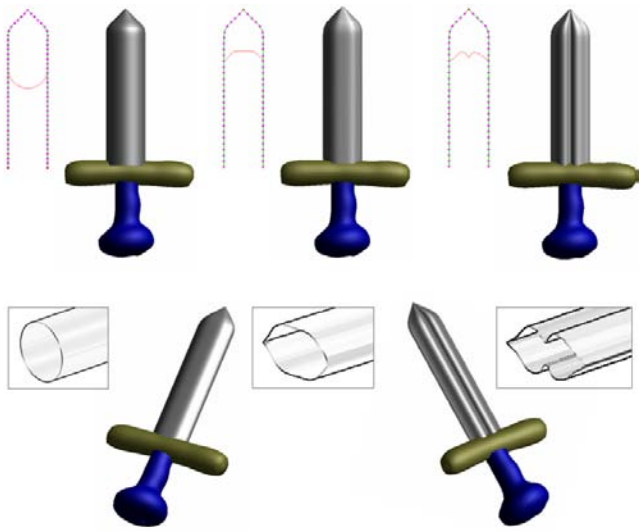

Figure 6: Modeling a sword blade using cross sectional blending surfaces. *Top row, left to right*: drawing the constructive curves only (dotted lines) results in a perfectly rounded object. Sketching the cross sectional outline (red line) results in a better, sharper, faceted blade. *Bottom row*: the sword in a different view. Notice the final surfaces, with sharper features

5 Editing Phase

In the editing phase, the user can modify the model that he/she constructed in the creation phase. This fits very well with the artistic design process of progressive drawing refinement, which is particularly prevalent in various sketching techniques (Section 1, Figure 1). We are free to use parametric or mesh representation at the editing stage. Consequently, any technique for mesh editing can be also employed here [Lawrence and Funkhouser 2003; Varley et al. 2000; Zorin et al. 1997]. However, we prefer to use the stroke based parametric representation again, which is consistent with our creation phase. This allows us to design editing operations that complement our creation phase. In addition, it enables us to keep the advantages of parametric representation, as we discussed in Section 3. The next subsections focus on describing two major parametric editing methods which are crucial for our system.

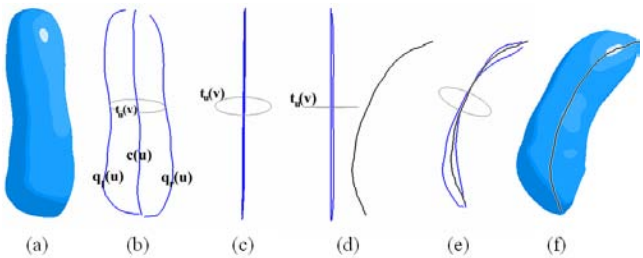

Figure 7: The orthogonal deformation stroke in action. **(a)** the surface we wish to deform; **(b)** its constructive curves $q_l(u)$, $q_r(u)$, the center curve $c(u)$, and a cross section of the surface $t_u(v)$; **(c)** the surface as viewed from the side, notice that $q_l(u)$, $q_r(u)$ and $c(u)$ are all the same, straight line when viewed from this angle; **(d)** the deformation stroke; **(e)** the cross section and the strokes morphing to the deformation stroke (the camera angle has been slightly altered for clarity); **(f)** the final, deformed surface.

Figure 8: *Left*: perspective view of three deformation strokes (in white) applied to the single leaf model of Figure 5. *Right*: artistic composition using our system illustrating the shape and color progression of autumn leaf. The stem was modeled as a rotational blending surface. The leaf of Figure 5 is placed at the top of the stem. The other three leaves are deformations of this top leaf using three different orthogonal deformation strokes.

5.1 Orthogonal Deformation Stroke

This first technique of the editing phase was inspired by the traditional bending (or distortion) method (Section 1, Figure 1). In this method, the artist adds variations to the overall 3D shape of the subject by distorting with few strokes its outlined form [Dease et al. 1999]. Next, we describe how our approach approximates the traditional bending method.

In the creation phase, the user specifies the surface with strokes by 2D drawing operations in the xy plane. This helps the user to have a natural drawing feeling while using our system, very similar to traditional pen-and-paper drawing. However, the drawback of this advantage is on the lack of flexibility for editing our models in the third dimension. In order to solve this problem, we propose a mechanism to allow the user to deform the model in a direction orthogonal to the drawing plane.

We begin by rotating the model so that we can see it from the side. Let $\mathcal{P}$ be the new view plane. Note that all three curves $q_l(u)$, $q_r(u)$ and $c(u)$ form an identical vertical line segment $l(u)$ in this plane. The user enters the deformation stroke $d(u)$ in $\mathcal{P}$ as illustrated in Figures 7 and 8. From the user perspective, this stroke shows the skeleton (the axis) of the deformed surface.

Based on our discussion in Section 4, the original surface $S(u,v)$ is formed by moving the cross section curve $t_u(v)$ along $c(u)$ (see Figures 3 and 7). Note that this is true for both surfaces of the cre-

Figure 9: A skirt modeled with cross sectional oversketch. *Top row, left to right*: starting with a simple tube, the user selects a cross section of the surface. Next, the user redraws a section of it and this edits the surface. *Bottom row*: the surface is rotated and drawn upon further, and the results are shown with both toon and Gouraud shading for clarity.

ation phase. We use the deformation stroke $d(u)$ to transform cross sections of $S(u,v)$ to a new set of curves that create the deformed surface $\hat{S}(u,v)$. More specifically, for every fixed u, we transform the cross section curve $t_u(v)$ to a new curve $\hat{t}_u(v)$. The appropriate transformation is determined by the relative situation of $c(u)$ (or $l(u)$) and $d(u)$ in the new view plane $\wp'$. For this, let $(l(u), T_l(u), V)$ denote the source frame formed at $l(u)$. In this notation, $T_l(u)$ is the unit tangent vector of $l(u)$ and V is the view vector. The destination frame is $(d(u), T_d(u), V)$, where $T_d(u)$ is the unit tangent vector of $d(u)$. Let $M(u)$ be the the transformation that maps the source frame to the the destination frame for every u. Consequently, for each u, $M(u)$ is a 4x4 transformation matrix consisted of a translation and a rotation. If we assume that both $\hat{t}_u(v)$ and $t_u(v)$ are represented in the homogenous coordinate system, then we have

$$\hat{t}_u(v) = M(u)t_u(v) \tag{3}$$

Again by changing u, the resulting curves $\hat{t}_u(v)$ construct the deformed surface $\hat{S}(u,v)$ as illustrated in Figures 7 and 8.

5.2 Cross Sectional Oversketch

This second technique of the editing phase is related to the traditional scribble method (Section 1, Figure 1), in a similar way as described for cross sectional blending surfaces (Section 4.2).

In our system, the user defines the cross section strokes in the drawing plane $\wp$. Again, this fits to the artistic approach for depicting 3D forms (Section 1, Figure 1). This assumption is a good selection for the default objects. However, there is a certain limitation due to the 2D mode of interaction. For example, it is hard to control the behavior of the cross section curve near to the intersections.

We have designed a simple method that allows the user to change the cross section stroke for any view. In this method, the user can rotate the object and change the view, and then he/she can select a cross section on the surface. This is done by setting the parameter u (Section 4.1) proportional to the mouse position. Then we highlight the corresponding cross section $t_u(v)$ on the surface that forms a visible interaction. At this stage, we map the changes given by the user to the cross section. This is an operation that allows the user to edit a surface by *oversketching*. As shown in Figure 9, we simply insert the new portion of the stroke, and delete the old one.

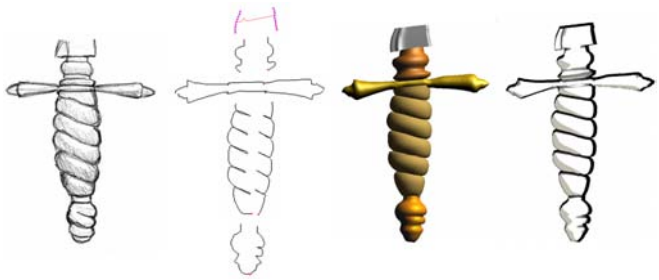

Figure 10: Modeling a dagger handle. *From left to right*: starting with the existing drawing *Gunner's Dagger* (Copyright 1998-2004 Rio Aucena. Used with permission), the user sketches 11 strokes for each of five specific parts of the original drawing. The blade is modeled as a cross sectional blending surface (3 strokes) and the other four parts are modeled as rotational blending surfaces (2 strokes per surface). The final model is then rendered with both Gouraud and non-photorealistic shading.

6 Results and Discussion

All the results were generated on an AMD Anthlon 2800 with a GeForce 5900 XT, 512 MB card, with quad meshes from our parametric representation rendered in OpenGL. We created 3D models using few strokes representing subjects of cartoon styles (Figures 4, 6, 9, 10, 11) and botanical illustrations (Figures 5, 8, 12, 13). We were able to construct models with sharp corners (i.e. candle in Figure 4), facets (i.e. sword in Figure 6), and bumps (i.e. pumpkin in Figure 11).

We observed that, in many cases, some models such as the pear, candle, laser gun, sword, leafy stalk, and pumpkin are particularly fast to create (around less than a minute). More complex models took longer because they relied more upon the assembly of parts (fitting each surface together). However, our research focused on steps of the modeling rather than the assembly process. Therefore, we implemented standard techniques for assembling 3D parts, in which the user directs translation and rotation by clicking and dragging with the mouse. We found this kind of assembly interface to be the major bottleneck of our creative process. When creating models of the wizard, Vulpix, and the bad guy (Figure 11), and for the yellow berries (Figure 13), over 60% of the time was spent on assembling the parts. The wizard took about an hour to create, of which about 35 minutes were spent assembling the parts. The yellow berries took around two hours to create with approximately 80 minutes spent on assembling the leaves and berries to the stem. This shows that the assembly process requires further streamlining and is in need of a new kind of interface, perhaps sketch- and/or rule-based.

We also noticed that we were able to create many of our models *more quickly than we could have hand-drawn and shade the same objects*. For instance, the constructive lines (contour) of a pear took the same amount of time on computer as on paper, but the computer-generated pear is quicker to create because it is automatically shaded. This is notable because it usually takes longer to model something on a computer than to sketch it on paper. This is one of the reasons artists draw "concept sketches" on paper rather than "concept models" on a computer.

7 Conclusions and Future Work

This paper presented a novel sketch-based system that allows interactive modeling of many free-form 3D objects with few numbers of strokes. Our technique draws on conventional drawing methods and workflow to derive interaction idioms that should be familiar to illustrators. We have developed algorithms for parametric surfaces using rotational and cross sectional blending. Although we were inspired by traditional pencil and paper drawing techniques, our methods allow either subtle or incremental (as with paper) or large-scale changes to the objects. Our results with cartoon-like features show that it is possible to model quite convoluted shapes with a small number of strokes. This differs from classical animation systems that require lots of menu selections and data entry to accomplish even the simplest tasks.

There are interesting directions for future work. As we mentioned in Section 6, the assembly of parts needs a more streamlined form of user interaction. A sketch- and/or rule-based solution might be appropriate, and perhaps the best solution would combine translation and rotation into a single operation. Another important feature we are considering is to support parametric representations with branching structures, to simplify modeling of more complex objects. We also plan on conducting a formal system evaluation with users with no-artistic background as well as with trained artists and illustrators in various domains (i.e. 3D content creation, cartoon, technical/scientific illustration).

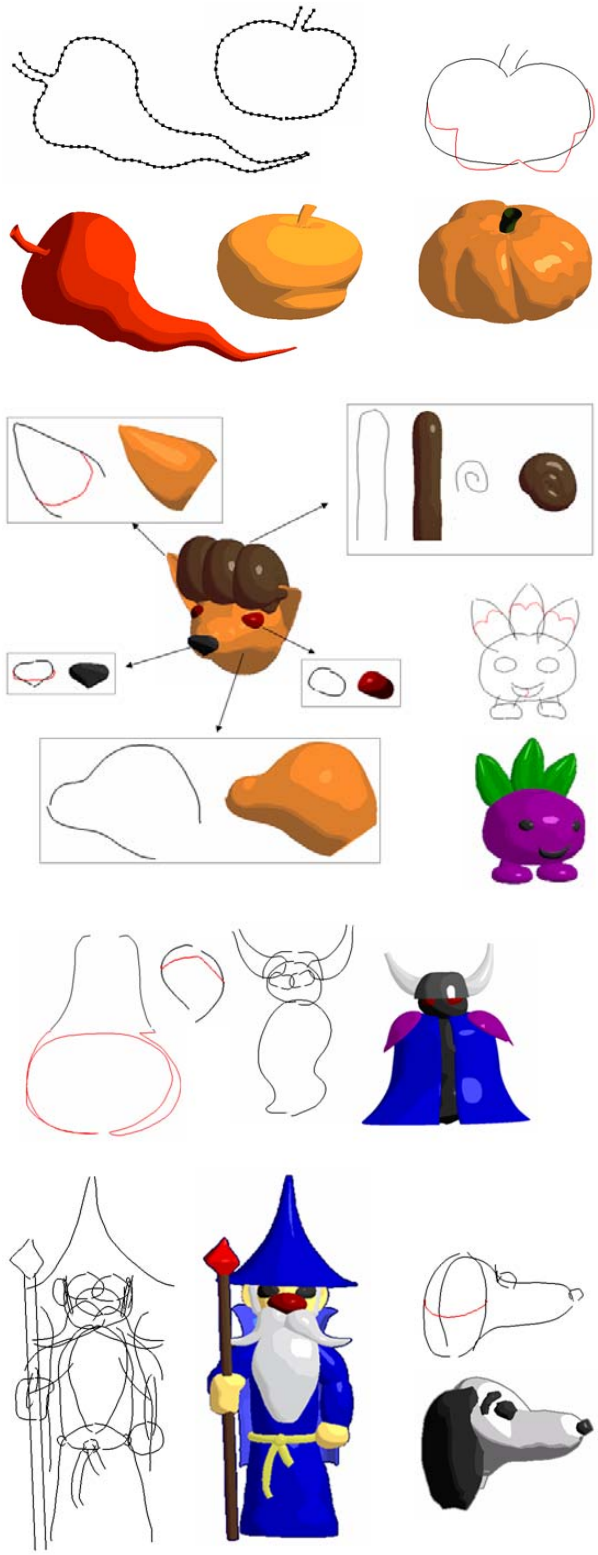

Figure 11: Cartoon-like shapes. *From top to bottom* with total number of strokes sketched by the user in parenthesis: paprika (2) and tangerine (2) (both sketched directly over the spiral method drawings in Figure 1), pumpkin (5), Vulpix (13), Oddish (15), bad guy (19), Wizard (57), and snoopy (10).

Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions. We also thank Rio Aucena and Siriol Sherlock for providing their drawings and paintings used in our experiments, and Patricia Rebolo Medici for her useful artistic advice and proofreading. This research was supported by Discovery Grants from the Natural Sciences and Engineering Research Council of Canada and by the Portuguese Science Foundation (FCT) BSAB-458-2004.

References

ANGEL, E. 2002. *Interactive Computer Graphics: A Top-Down Approach Using OpenGL, 3rd Edition*. Addison Wesley Professional.

BARTELS, R. H., AND SAMAVATI, F. F. 2000. Reversing subdivision rules: Local linear conditions and observations on inner products. *Journal of Computational and Applied Mathematics 119*, 1-2, 29–67.

CURVY 3D. 2004. Aartform. *http://www.curvy3d.com*.

DE ARAUJO, B., AND JORGE, J. 2003. Blobmaker: Free-form modelling with variational implicit surfaces. In *Proc. of the 12th Portuguese Computer Graphics Meeting*, 17–26.

DEASE, C., GRINT, D., AND KENNEDY, D. 1999. *Complete drawing course (the diagram group)*. Sterling Publishing Co., Inc.

DOUGLAS, D., AND PEUCKER, T. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer 10*, 2, 112–122.

EGGLI, L., HSU, C., BRUDERLIN, B., AND ELBER, G. 1997. Inferring 3d models from freehand sketches and constraints. *Computer-Aided Design 29*, 2, 101–112.

GOLDSTEIN, N. 1999. *The Art of Responsive Drawing*. Prentice-Hall, Inc.

GUPTILL, A. 1977. *Rendering in Pencil*. Watson-Guptill Publications.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freeform design. In *Proc. of SIGGRAPH '99*, 409–416.

LAWRENCE, J., AND FUNKHOUSER, T. 2003. A painting interface for interactive surface deformations. In *Proc. of Pacific Graphics '03*, 141–150.

NAYA, F., JORGE, J. A., CONESA, J., CONTERO, M., AND GOMIS, J. M. 2002. Direct modeling: from sketches to 3d models. In *Proc. of the 1st Ibero-American Symposium in Computer Graphics*, 109–117.

PEREIRA, J. P., BRANCO, V. A., JORGE, J. A., SILVA, N. F., CARDOSO, T. D., AND FERREIRA, F. N. 2004. Cascading recognizers for ambiguous calligraphic interaction. In *Proc. of the Eurographics Workshop on Sketch-Based Modeling (SBM'04)*.

SAMAVATI, F. F., AND BARTELS, R. H. 2004. Local filters of b-spline wavelets. In *Proceedings of International Workshop on Biometric Technologies (BT 2004)*, University of Calgary, Canada.

SAMAVATI, F., AND MAHDAVI-AMIRI, N. 2000. A filtered b-spline models of scanned digital images. *Journal of Science 10*, 4, 258–264.

SKETCHUP. 2005. Last Software, Inc. *http://www.sketchup.com*.

TURK, G., AND O'BRIEN, J. 1999. Shape transformation using variational implicit surfaces. In *Proc. of SIGGRAPH '99*, 335–342.

VARLEY, P., SUZUKI, H., MITANI, J., AND MARTIN, R. 2000. Interpretation of single sketch input for mesh and solid models. *International Journal of Shape Modeling 6*, 2, 207–240.

VRMESH. 2005. VirtualGrid Company. *http://www.vrmesh.com*.

WEST, K. 1983. *How to draw plants: the techniques of botanical illustration*. The Herbert Press Limited.

Z-BRUSH. 2005. Pixologic. *http://www.pixologic.com*.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 1996. SKETCH: An interface for sketching 3D scenes. In *Proc. of SIGGRAPH '96*, 163–170.

ZORIN, D., SCHRODER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *Proc. of SIGGRAPH '97*, 259–268.

A Formal Construction of Blending Surfaces

Rotational

Let (x,y) be a coordinate system consisting of two orthogonal unit vectors in the plane $\wp$. Recalling Section 4.1, $\wp$ is the plane in which the left and right constructive curves $q_l(u)$ and $q_r(u)$ lie. Let z be $x \times y$ so that (x,y,z) forms a coordinate system for 3D space.

In order to define $S(u,v)$ in a more formal way, we show that $S(u,v)$ can be formed by a series of affine transformations on the circular cross sections of a cylinder. Let $Q(u,v)$ be the unit cylinder in the 3D space

$$Q(u,v) = \begin{bmatrix} cos(v) \\ u \\ sin(v) \\ 1 \end{bmatrix}, \quad \begin{matrix} 0 \le u \le 1 \\ 0 \le v \le 2\pi \end{matrix},$$

and define

$$p_l(u) = Q(u,\pi), \tag{A-1}$$

$$p_r(u) = Q(u,0), \tag{A-2}$$

$$p_c(u) = \frac{1}{2}p_l(u) + \frac{1}{2}p_r(u). \tag{A-3}$$

In our construction of $S(u,v)$, $p_l(u)$ is mapped to $q_l(u)$, $p_r(u)$ to $q_r(u)$ and $p_c(u)$ to $c(u)$. For any fixed u, we have a unit circle in $Q(u,v)$ and a general circle in $S(u,v)$ and we wish to map the unit circle to the general one. This can be done by applying an affine transformation $M_s(u)$ to $Q(u,v)$

$$S(u,v) = M_s(u)Q(u,v). \tag{A-4}$$

Notice $M_s(u)$ consists of two affine transformations

$$M_s(u) = M_2(u)M_1(u)$$

where $M_1(u)$ is a scaling about $p_c(u)$ with the following parameters:

$$\begin{aligned} scale_x &= \|q_r(u) - q_l(u)\| \\ scale_y &= 1 \\ scale_z &= \|q_r(u) - q_l(u)\|. \end{aligned}$$

And $M_2(u)$ is a frame transformation [Angel 2002] from $(p_c(u),x,y,z)$ to $(c(u),x'(u),y'(u),z)$ where

$$x'(u) = \frac{q_r(u) - q_l(u)}{\|q_r(u) - q_l(u)\|},$$

and

$$y'(u) = z \times x'(u).$$

Cross Sectional

Using a similar approach, we can define our cross sectional blending surface. For this surface, we replace $Q(u,v)$ by a ruled surface. Let $t(v) = \begin{bmatrix} x(v) \\ z(v) \end{bmatrix}, 0 \le v \le 2\pi$ be the cross sectional curve. Then

$$Q(u,v) = \begin{bmatrix} x(v) \\ u \\ z(v) \\ 1 \end{bmatrix} \quad \begin{matrix} 0 \le v \le 2\pi \\ 0 \le u \le 1 \end{matrix}.$$

We define $p_l(u)$, $p_r(u)$, and $p_c(u)$ exactly as in equations A-1, A-2, and A-3. Consequently, the cross sectional blending surface is resulted by the equation A-4.

Figure 12: *Rose*, modeled in 23 strokes. The petals and the three stems were generated in 2 strokes each using rotational blending surfaces. Each leaf was generated with 3 strokes using cross sectional blending surfaces (Figure 5), distorted (Section 8) and interactively placed at the stem using standard rotation/translation modeling tools. *Bottom image*: (left) the two strokes sketched for the rose petals; (middle and right) real botanical illustrations were used as templates for sketching over the leaves (two middle leaves [West 1983] and a sample from a painting).

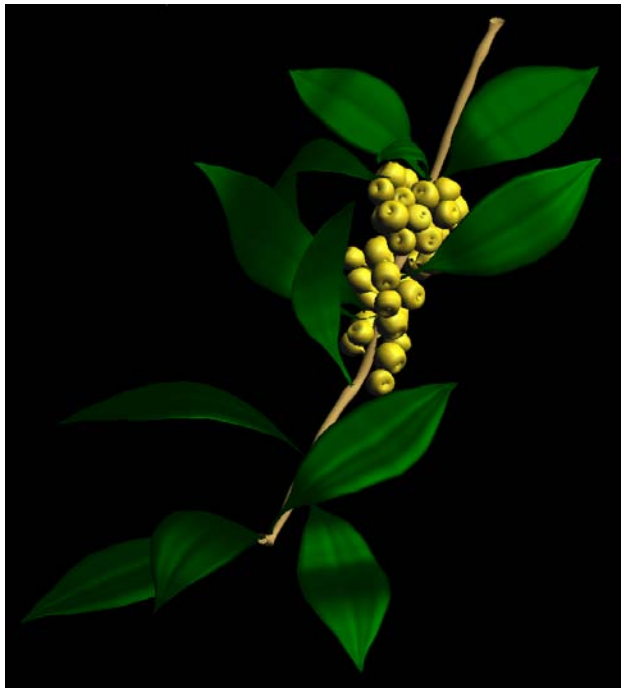

Figure 13: *Pyramidalis Fructu Luteo* (yellow berries), modeled in 33 strokes. The berries and the stem were generated in 2 strokes each using rotational blending surfaces. Each leaf was generated with 3 strokes using cross sectional blending surfaces (Figure 5). *Bottom left image*: The user sketched directly over nine specific parts of a real botanical illustration of yellow berries (Copyright 2004 Siriol Sherlock. Used with permission.): one stem, three leaves and five berries (*right image*). In the model, all leaves and berries are instances of these nine sketched-based objects. Each of the leaf instances was properly distorted (Section 8) and both leaves and berries were then placed at the stem by the user with standard modeling tools.