Structured Annotations for 2D-to-3D Modeling

Yotam Gingold[*]
New York University / JST ERATO

Takeo Igarashi
University of Tokyo / JST ERATO

Denis Zorin
New York University

Abstract

We present a system for 3D modeling of free-form surfaces from 2D sketches. Our system frees users to create 2D sketches from arbitrary angles using their preferred tool, which may include pencil and paper. A 3D model is created by placing primitives and annotations on the 2D image. Our primitives are based on commonly used sketching conventions and allow users to maintain a single view of the model. This eliminates the frequent view changes inherent to existing 3D modeling tools, both traditional and sketch-based, and enables users to match input to the 2D guide image. Our annotations—same-lengths and angles, alignment, mirror symmetry, and connection curves—allow the user to communicate higher-level semantic information; through them our system builds a consistent model even in cases where the original image is inconsistent. We present the results of a user study comparing our approach to a conventional "sketch-rotate-sketch" workflow.

CR Categories: I.3.6 [Methodology and Techniques]:Interaction techniques I.3.5 [Computational Geometry and Object Modeling]: Geometric algorithms, languages, and systems

Keywords: user interfaces, sketch-based modeling, annotations, interactive modeling, image-based modeling

1 Introduction

Traditional 3D modeling tools (e.g. [Autodesk 2009]) require users to learn an interface wholly different from drawing or sculpting in the real world. 2D drawing remains much easier than 3D modeling, for professionals and amateurs alike. Professionals continue to create 2D drawings before 3D modeling and desire to use them to facilitate the modeling process ([Thormählen and Seidel 2008; Tsang et al. 2004; Eggli et al. 1997; Kallio 2005; Dorsey et al. 2007; Bae et al. 2008]). Sketch-based modeling systems, such as Teddy [Igarashi et al. 1999] and its descendants, approach the 3D modeling problem by asking users to sketch from many views, leveraging users' 2D drawing skills. In these systems, choosing 3D viewpoints remains an essential part of the workflow: most shapes can only be created by sketching from a large number of different views. The workflow of these systems can be summarized as "sketch-rotate-sketch." Because of the view changes, users cannot match their input strokes to a guide image. Moreover, finding a good view for a stroke is often difficult and time-consuming: In [Schmidt et al. 2008], a 3D manipulation experiment involving users with a range of 3D modeling experience found that novice users were unable to complete their task and became frustrated. These novice users "positioned the chair parts as if they were 2D objects." The change of views is a *major bottleneck* in these systems.

[*]e-mail: gingold@cs.nyu.edu

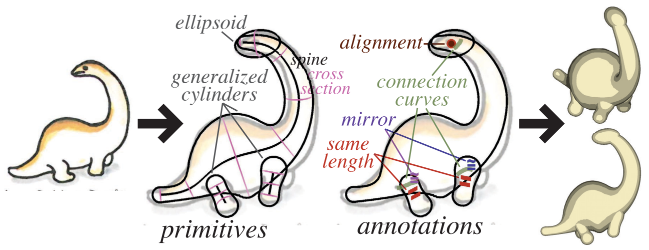

Figure 1: *Our modeling process: the user places primitives and annotations on an image, resulting in a 3D model.*

Sketching is also used in the context of traditional modeling systems: a workflow often employed by professional 3D modelers is placing axis-aligned sketches or photographs in the 3D scene for reference. This workflow could potentially allow amateurs who cannot draw well in 2D to create 3D models from sketches produced by others. Yet, paradoxically, this approach requires a *higher* level of skill despite relying on easier-to-produce 2D sketches as a modeling aid. This is because of the difficulty of using conventional tools, which require constant changes to the camera position, whereas a single view is needed to match an existing image.

The goal of our work is to design a user interface that simplifies modeling from 2D drawings and is accessible to casual users. Ideally, an algorithm could automatically convert a 2D drawing into a 3D model, allowing a conventional sketch (or several sketches) to serve as the sole input to the system. This would eliminate the need for view point selection and specialized 3D UI tools. However, many (if not most) drawings are ambiguous and contain inconsistencies, and cannot be interpreted as precise depictions of any 3D model (Section 3). This limits the applicability of techniques such as Shape-from-Shading ([Prados 2004]) and reconstruction from line drawings ([Varley and Company 2007]). Humans apparently resolve many of the ambiguities and inconsistencies of drawings with semantic knowledge. Our work provides an interface for users to convert their interpretation of a drawing into a 3D shape. Instead of asking the user to provide many sketches or sketch strokes from multiple points-of-view, we ask the user to provide all information in 2D from a *single view*, where she can match her input to the underlying sketch. In our tool, user input takes the form of (1) *primitives* (generalized cylinders and ellipsoids) with dynamic *handles*, designed to provide complete flexibility in shape, placement, and orientation, while requiring a single view only, and (2) *annotations* marking similar angles, equal-length lines, connections between primitives, and symmetries, to provide additional semantic information. Our system generates 3D models entirely from this user input and does not use the 2D image, which may be inaccurate, inconsistent, sparse, or noisy. We do not expect that users have a consistent 3D mental model of the shape and are specifying primitives precisely; we aim to create plausible, reasonable quality 3D models even if a user's input is inconsistent.

Contributions. We have designed a system of user interface elements implementing an intuitive and powerful paradigm for interactive free-form modeling from existing 2D drawings, based on the idea of "describing" an existing drawing by placing primitives and annotations.

We present the results of a small user study showing that our interface is usable by artists and non-artists after minimal training and demonstrating that the results are consistently better compared to tools using the "sketch-rotate-sketch" workflow. We demonstrate that our system makes it possible to create consistent 3D models qualitatively matching inconsistent illustrations.

Our resulting 3D models are collections of primitives containing structural information useful in applications such as animation, deformation, and further processing in traditional modeling tools. We do *not* argue that one should perform all 3D modeling operations in a 2D view. Our goal is to demonstrate that it is possible to accelerate the creation of initial, un-detailed 3D models from 2D sketches, which can be further refined and improved using other types of modeling tools.

2 Related Work

Interactive, single-view modeling. Our approach is most similar in spirit to [Zhang et al. 2001] and [Wu et al. 2007], in which users annotate a single photograph or drawing with silhouette lines and normal and positional constraints; the systems solve for height fields that match these constraints. In our system, the primitives and annotations added by a user are structured and semantic, and we are able to generate 3D models from globally inconsistent drawings. Our system rectifies the shape primitives placed by a user in order to satisfy the user's annotations (symmetries and congruencies).

[Andre et al. 2007] presented a single-view modeling system intended to mimic the process of sketching on a blank canvas. We are similarly motivated; our system allow users to preserve intact the process of sketching on a blank canvas.

Interactive, multiple-view modeling. In [Debevec et al. 1996] and [Sinha et al. 2008] and [van den Hengel et al. 2007], users mark edges or polygons in multiple photographs (or frames of video). The systems extract 3D positions for the annotations, and, in fact, textured 3D models, by aligning the multiple photographs. (In [Debevec et al. 1996], users align them to edges of a 3D model created in a traditional way). In our system, users have only a single, potentially inconsistent drawing; these computer vision-based techniques assume accurate, consistent input and hence cannot be applied to our problem.

Automatic, single-view sketch recognition. Sketch recognition techniques convert a 2D line drawing into a 3D solid model. These approaches also typically assume a simple projection into the image plane. Furthermore, a variety of restrictions are placed on the line drawings, such as the maximum number of lines meeting at single point, and the implied 3D models are assumed to be, for example, polyhedral surfaces. For a recent survey of line-drawing interpretation algorithms, see [Varley and Company 2007]. One notable recent work in this direction is [Chen et al. 2008], which allows for imprecise, sketched input by matching input to a domain-specific database of architectural geometry. More relevant to free-form modeling, the recent works of [Karpenko and Hughes 2006] and [Cordier and Seo 2007] generate 3D models from a single view's free-form visible silhouette contours. These works are primarily concerned with generating surfaces ([Karpenko and Hughes 2006]) or skeletons ([Cordier and Seo 2007]) correctly embedded in $\mathbb{R}^3$ with visible contours matching the user's input. They do not represent modeling systems per se, but rather a necessary component for any system taking silhouette contours as input. Our approach can be viewed as a form of user-assisted 2D-to-3D interpretation. Because a human uses our tool to annotate the 2D image, we are able to receive user input that eliminates ambiguity and rectifies inconsistencies in the image.

Interactive 3D modeling. There are a variety of sketch-based modeling tools based on the concept of sketching curves from various angles (sketch-rotate-sketch). The earliest of these is [Igarashi et al. 1999], and this direction has been explored in a variety of later works. A good overview can be found in the recent survey of [Olsen et al. 2008]. These works assume users are capable of sketching a model from multiple points-of-view, and that users can find good views for sketching and manipulating the model. As such, users cannot trace a guide image. We do not assume such skill, and believe that that rotation and sketching from novel views is the most difficult aspect of these systems.

The work of [Cherlin et al. 2005] deserves further mention. The goal of this work is minimizing the number of strokes a user must draw to create a surface. They introduce "rotational blending surfaces," which are similar to our generalized cylinders. However, these surfaces' "spines" are planar *xy* curves unless over-sketched from a rotated view.

Generalized cylinders have been used extensively in the blobby modeling literature ([Bloomenthal 1997]). The MetaReyes 3D modeling system ([Infografica 2009]) makes extensive use of a generalized cylinder primitive with non-circular cross sections and is based on implicit surfaces; our generalized cylinder primitive could be implemented similarly. MetaReyes is not designed to be a single-view modeler; as a result, users cannot easily match a guide image.

Two early sketch-based modeling systems relevant to our work are [Zeleznik et al. 1996] and [Eggli et al. 1997]. [Zeleznik et al. 1996] introduced SKETCH, a gestural interface for 3D modeling. In SKETCH, users are capable of performing most modeling operations from a single view. For this reason, SKETCH could almost be re-purposed as a tool for annotating existing 2D drawings. However, inconsistencies commonly present in 2D drawings preclude this application of SKETCH. In addition, SKETCH only supports a subset of CAD primitives and cannot be used for free-form modeling. [Eggli et al. 1997] introduced constraints for beautifying users' imprecise, sketched input. [Cabral et al. 2009] presented an interactive system for deforming and reshaping existing architectural models with length and angle constraints. We, too, use constraints, although our constraints are also designed to reconcile globally inconsistent user input.

The systems presented in [Kallio 2005; Dorsey et al. 2007; Bae et al. 2008] allow users to sketch freely with 2D input devices, projecting users' strokes onto various planes in space. These systems are designed to support the early conceptual design phase of a project; as such, they do not create surfaces, but rather a collection of strokes in space. Our interface can be used to create a 3D surface from a sketch drawn during a traditional 2D conceptual design phase.

3 Motivation

3D shapes in 2D sketches. 2D sketches are remarkably efficient at conveying the information sufficient to perceive a 3D object from a single point of view. Many 2D drawing approaches are based on composing (or decomposing, if drawing reality) a model out of primitive shapes (Figure 2, bottom row). The 3D shape of each primitive is relatively simple; primitives are primarily depicted with outlines and additional "scaffolding" ([Schmidt et al. 2007]), typically lines indicating the shape of several *cross-sections*. In [Vilppu 1997], these primitives are sphere-, cylinder-, and cube-like shapes. Similarly, our interface is based on the idea of users placing primitives over a 2D drawing and modifying primitives' cross-sections to control their shape and match their appearance to the drawing.

Global inconsistency and ambiguity. Many 2D drawings are globally inconsistent and contain various ambiguities. It is well known that drawings and paintings, from quick sketches to classical works of art, contain elements drawn from different perspectives ([Agrawala et al. 2000]); in many cases, one cannot unambiguously infer the 3D view used for the 2D image.

Due to inconsistencies and inaccuracies, intended or unintended, one cannot in general take a drawing to be a precise projection of a 3D model (e.g. Figure 2, top row). For this reason, we cannot hope to reconstruct a good-quality 3D model without additional informa-

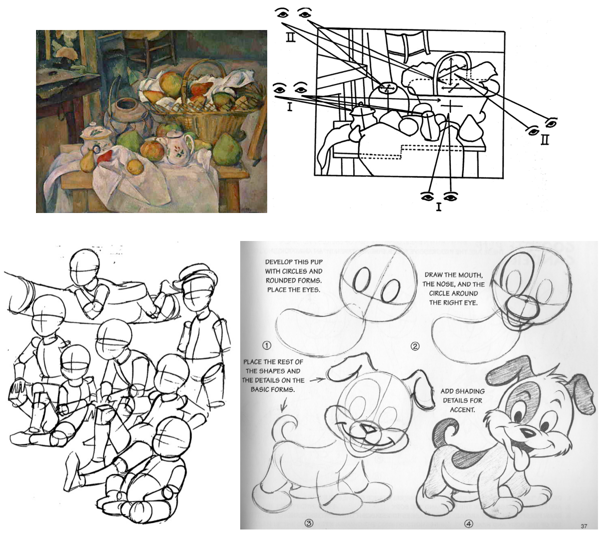
II
I
II
I
DEVELOP THIS PUP WITH CIRCLES AND ROUNDED FORMS. PLACE THE EYES.
DRAW THE MOUTH, THE NOSE, AND THE CIRCLE AROUND THE RIGHT EYE.
PLACE THE REST OF THE SHAPES AND THE DETAILS ON THE BASIC FORMS.
ADD SHADING DETAILS FOR ACCENT.
①
②
③
④

Figure 2: *Top row: Cezanne's* Still Life with a Fruit Basket *(diagram from [Loran 1943], reproduced from [Agrawala et al. 2000]). Bottom row: Drawing using primitive shapes from [Vilppu 1997] (left) and [Blair 1994] (right).*

tion. Nevertheless, even poor quality drawings are understandable by humans; they are able to convey local shape information and are usually sufficient to recognize the object. Our system provides several types of *annotations* for users to input essential semantic information. We combine the primitives' local shape information with the annotations' semantic information to create a plausible, globally consistent 3D model.

4 User Interface

Users begin a modeling session by choosing an arbitrary image file containing their drawing (or other source material). Our interface presents users with a window displaying the image and a palette of primitive and annotation tools (Figure 3). *Primitives* (generalized cylinders and ellipsoids) are used to create simple object parts; they are manipulated with several types of handles for single-view shape editing. *Annotations* describe geometric relationships between these shapes, such as equality of lengths and angles. Users proceed to place primitives and annotations over the image, which is visible as an underlay; only primitives' silhouettes and cross-sections are visible in the interaction view. (In case the 2D view becomes cluttered, the user can "lock" shapes. Locked shapes cannot be selected and are drawn with a hairline outline. This is similar to the approach taken in Illustrator ([Adobe 2007]).)

Primitives (e.g., a character's body) are initially created with their spines flat in the image plane. The main tools allowing implicit out-of-image-plane shape deformation and 3D positioning are cross-section tilt handles and connection curves attaching two primitives together. By tilting cross-sections, the user "lifts" a shape out of the plane, while preserving its outline. This process is similar to the way artists suggest the 3D shape and orientation of an object by sketching cross-sections, even if these are erased in the final image. Primitives are positioned with respect to each other (e.g., legs with respect to the body) using connection curves. Connection curve annotations connect two primitives where they overlap, and determine the depth ordering. Tilting cross sections and attaching primitives together make it possible to define a 3D shape matching a sketch using single-view interactions.

Additional relationships between primitives can be specified by other annotations: alignment, same-length and angle, and mirroring. Alignment and mirror annotations place primitives with respect to the *symmetry sheet* of another primitive. Symmetry sheets are controlled by the user and generalize symmetry planes; unlike

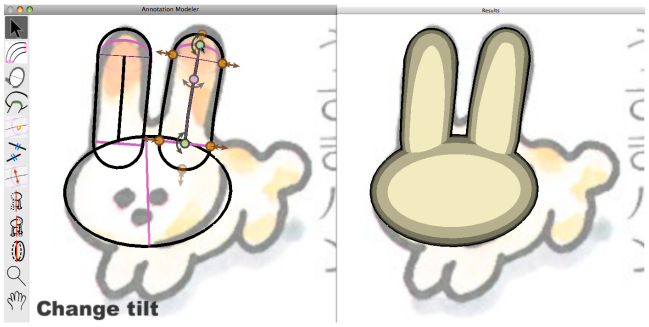

Figure 3: *A screenshot of our interface.*

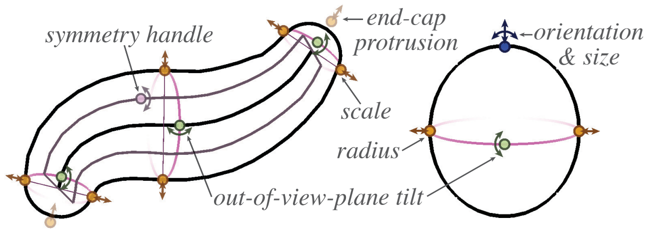

Figure 4: *Primitives: A generalized cylinder (left) and an ellipsoid (right).*

a mathematically defined symmetry plane or axis, symmetry sheets do not necessarily capture a precise geometric property of a shape; rather, it provides a means to communicate to the system the semantics of the object. For example, consider a cylinder deformed to a snake-like shape. Before the deformation, the cylinder had well-defined symmetry planes which became non-planar sheets; the sheets retain the semantics of symmetry planes.

A 3D preview window is visible at all times, allowing the user to view and rotate the 3D model. This window is only used for verifying the model, not for editing. For inconsistent sketches, the 3D model cannot match the 2D sketch and user input exactly, and the ability to evaluate the 3D model directly is essential.

4.1 Primitives

In the interaction view, primitives are depicted as 2D outlines with handles for manipulating their degrees of freedom. The primitives resemble Autoshapes ([Microsoft 2003]), with an important difference that the number of handles is not predefined and can be altered by the user. Following our observations in Section 3, we provide two kinds of primitives, generalized cylinders and ellipsoids (Figure 4). While an ellipsoid can be treated as a special case of a generalized cylinder, we have found it important to consider it a separate primitive type, as the set of suitable handles is different. The guiding design principle for all controls is direct manipulation of the appearance of 2D projections of curves associated with a primitive, for example, a cross-section or a silhouette. In standard 3D modeling systems, the user controls parameters such as 3D rotations, translations, and spatial dimensions. In our system, those parameters are affected indirectly when users match a primitives' curves to curves in a 2D sketch. This indirect manipulation of 3D parameters by direct manipulation in 2D enables the single-view manipulation necessary for easy construction of 3D models from sketches.

Generalized Cylinders. The first primitive is the generalized cylinder (Figure 4, left). It is defined by a 2D spine curve and attributes associated with cylinder cross-sections: cross-section shape (a closed curve), out-of-image-plane cross-section tilt, cross-section scale, and the symmetry director used as a reference for orienting cross-sections and to define the symmetry sheet as explained below. Most of these attributes are directly controlled by handles, excluding the cross-section shape, for which a separate 2D sketching mode is used. These attributes are smoothly interpolated along the spine. At either end, the user can choose the end-cap protrusion as well. The 3D shape is defined as the union of scaled cross-sections (with tangent-continuous end-caps); the exact definition is presented in Section 5.1. To create a generalized cylinder, the user draws a free-form spine in 2D (Figure 5). A newly created generalized cylinder has a circular cross-section, a symmetry director parallel to the image plane, no out-of-image-plane tilt, a scale 5% of the screen width, and hemispherical end-caps. The newly created primitive has scale handles and tilt handles at both ends of the spine. The user can add and remove handles freely along the spine, provided that there is at least one handle of each type.

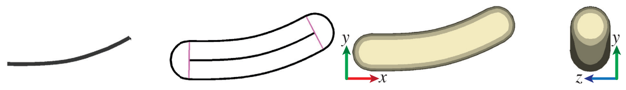

Figure 5: *Creating a generalized cylinder from a stroke.*

Out-of-image-plane tilt handle. This is one of the most important shape manipulation tools we provide. Out-of-image-plane tilt handles make it possible to bend a primitive out of the image plane while preserving its silhouette, which typically matches a silhouette in the 2D guide image. With no tilt, the cross-section plane is perpendicular to the image plane and so appears as a straight line. By dragging the handle, the user tilts the cross-section, rotating it about the axis perpendicular to the spine and parallel to the image plane; the cross-section's 2D projection changes to a closed curve (Figure 6). The 3D preview window is often important for cross-section tilt manipulation. While specifying a tilt in the 2D view allows the user to maintain consistency with the sketch, small changes in the cross-section tilt can sometimes result in large changes in the 3D shape; the ability to evaluate the result from an additional viewpoint allows for more reliable control of orientation.

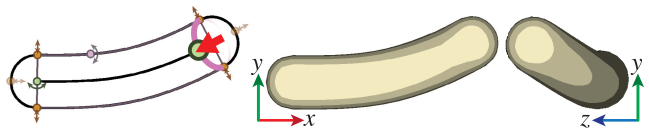

Figure 6: *Tilting a circular cross-section out of the image plane. (Editing the generalized cylinder from Figure 5.)*

Cross-section scale handle. A scale handle is used to change a cross-section's size. A scale handle is added anytime a user clicks and drags on the primitive's silhouette. As the user drags, the opposite point on the silhouette remains fixed, while the silhouette point under the mouse follows (Figure 7, top); the corresponding point on the spine is adjusted to remain in the center of the cross-section. Optionally, the user can scale the cross-section symmetrically, keeping the spine fixed (Figure 7, bottom).

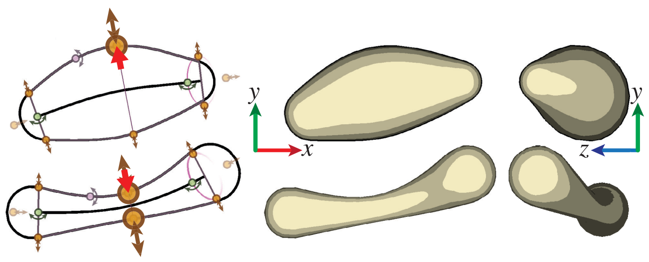

Figure 7: *Top row: Changing the scale of a cross-section; the handle opposite remains fixed. Bottom row: Changing the scale of a cross-section; the handle opposite moves synchronously. (Both rows edit the generalized cylinder from Figure 6.)*

Symmetry sheet handle. The symmetry sheet is defined by the 3D spine and the interpolated directors. The sheet is the ruled surface obtained as a union of the director lines at all points along the spine. We provide a handle to the user to select the director in any cross-section. The intersection of the sheet with the surface of the generalized cylinder is displayed in purple (Figure 8).

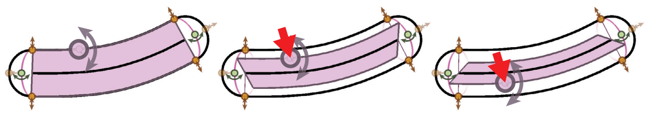

Figure 8: *Adjusting the symmetry sheet of a generalized cylinder.*

Cross-section shape adjustment mode. Arbitrary cross-section shapes can be drawn at any point along the spine, via a separate 2D mode accessible in a contextual menu (Figure 9). In this mode, a canvas is displayed and the user may draw the arbitrary cross-section curve or choose from a palette of common cross-sections. Cross-section curves are oriented so that the *x* direction of the 2D cross-section curve is aligned with the symmetry director and normalized to fit inside a unit circle. They are then scaled by the corresponding scale along the spine.

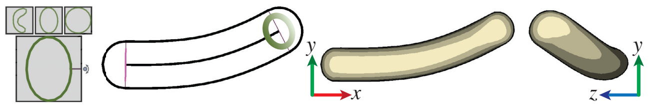

Figure 9: *Editing a generalized cylinder's cross-section curve. The edited cross-section is drawn in green. (Editing the generalized cylinder from Figure 6.)*

Spine manipulation. Finally, the user can click and drag any point on the spine to initiate a curve deformation with a peeling interface, as in [Igarashi et al. 2005]. Alternatively, the user can also over-sketch the spine, replacing all or a portion of it, depending on whether the over-sketching curve begins or ends near the spine. These operations are shown in Figure 10.

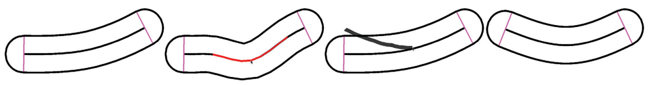

Figure 10: *From left to right: A generalized cylinder; deforming its spine by peeling; over-sketching its spine; the result of over-sketching.*

End-cap handles. End-cap handles determine the protrusion of the end-caps at either end of the generalized cylinder. These handles lie on the shape outline at the points obtained by continuing the spine to the silhouette tangentially (Figure 11).

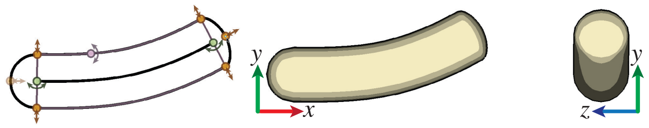

Figure 11: Adjusting the end-cap protrusion. (Editing the generalized cylinder from Figure 5.)

Ellipsoids. The second primitive is the ellipsoid (Figure 4, right). With this primitive, the user draws a free-form curve, to which a 2D ellipse is fit (in a least-squares sense). Handles are provided to change its out-of-image-plane tilt and the length and orientation of the axes of its 2D projection (Figure 12). The out-of-image-plane tilt handle appears and operates identically to that of a generalized cylinder. Ellipsoids' two smaller axes are constrained to have the same length, so all cross-sections perpendicular to its longest axis are circular. An ellipsoid's symmetry plane passes through its center and is perpendicular to its long axis; as a result, it is tied to and controlled by the the same handle as the cross-section tilt.

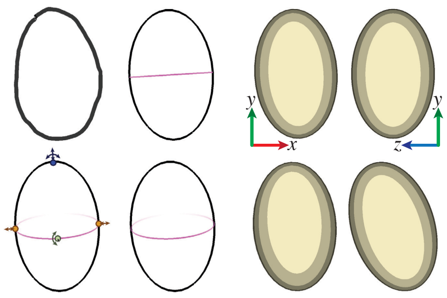

Figure 12: *Ellipsoids. Top row: Creating an ellipse from a stroke. Bottom row: Tilting a circular cross-section out of the image plane.*

4.2 Annotations

Annotations are the key to establishing a consistent 3D model that reconciles primitives placed on an inconsistent 2D image with geometric properties the user knows to be true. For example, the arms of a character should be the same length, even if there is no 3D model with arms of the same length that would project to the provided 2D sketch precisely. In addition, a character's arms should be attached to its body symmetrically opposite each other.

Connection curve annotations attach two primitives together and establish the (relative) depth between them. Users connect two overlapping primitives to each other—which places them in depth—by clicking and dragging in their 2D overlapping region. The primitives are translated in depth so that their 3D surfaces meet under the mouse; the entire curve where the two surfaces meet is drawn in green (Figure 13). Alternatively, users may draw the freeform intersection curve he or she wishes the 3D surfaces would make with each other. This stroke must begin on the overlap of exactly two primitives, which determines which two primitives are to be connected. By default, the connection curve is taken to be the intersection curve between the front faces of the 3D surfaces; this can be changed to the back faces via a menu. The interface refuses to add a connection curve annotation if doing so would create circular constraints.

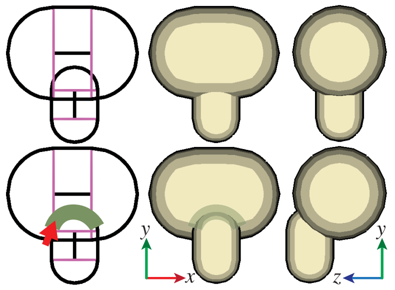

Figure 13: *Attaching two primitives with a connection curve annotation.*

Mirror annotations create a copy of a primitive (and its attached primitives) reflected across another primitive's symmetry sheet (Figure 14). Mirror annotations can be used to create characters' occluded, symmetric arms or legs.

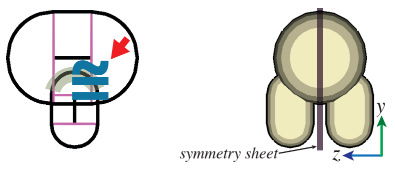

Figure 14: *Mirroring one primitive about another. (Annotating the primitives from Figure 13.)*

Alignment annotations align one or two primitives with respect to a connected primitive's symmetry sheet. If only one primitive is chosen to be aligned, it is translated so that its attachment origin (defined in Section 5) lies on the symmetry plane (Figure 15, top row). If two primitives are chosen to be aligned, they are translated so that their attachment origins are a reflection of each other with respect to the symmetry sheet (Figure 15, bottom row).

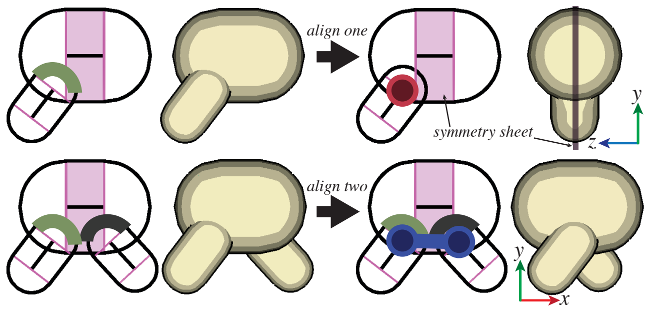

Figure 15: *Top row: Aligning one primitive on another's symmetry plane. Bottom row: Aligning two primitives with respect to another's symmetry plane. (The dark connection curve connects back faces.)*

Several annotations mark equal geometric measurements. **Same-length** annotations mark two or more primitives as having the same long axis length, for an ellipsoid, or 3D spine length, for a generalized cylinder. Users mark primitives, and same length markings, familiar from geometry textbooks, appear along the primitives' lengths (Figure 16). Note that the primitives in the interaction view do not change; in general, primitives are placed to match a guide image.

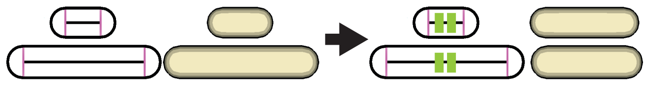

Figure 16: *A same-length annotation.*

Same-tilt annotations mark two or more cross-sections as having planes whose tilt angles with respect to the image plane are the same. As users mark circular cross-sections, angle markings, familiar from geometry textbooks, appear near the center of the titled cross-sections (Figure 17).

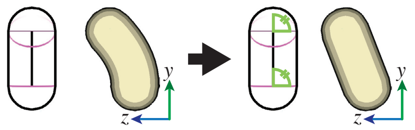

Figure 17: *A same-tilt annotation.*

Same-scale annotations mark two or more cross-sections as having the same scale. The arrows spanning the diameter of the chosen cross-sections are marked with familiar same length markings (Figure 18).

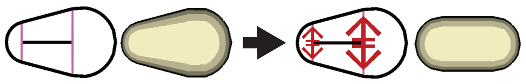

Figure 18: *A same-scale annotation.*

5 Implementation

In this section, we describe how we build a 3D model from user input. Because we assume that images are globally inconsistent, some primitives' parameters may contradict annotations: for example, the user may indicate that two legs have equal length, yet the primitives defining the legs do not. Because annotations provide

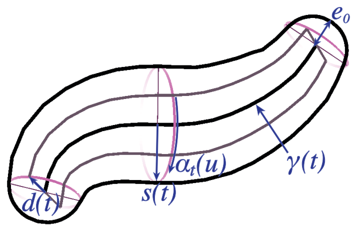

Figure 19: *Notation introduced in Section 5.1.*

semantic information the user knows to be true, annotations take precedence over primitive parameters set using handles.

Annotations are applied in the following order: (1) same-scale (2) same-tilt (3) connection curves (4) same-length (5) alignment (6) mirror. Same-scale and same-tilt annotations modify the shape of primitives; connection curves position primitives; same-length and alignment annotations adjust the shape and position of primitives; and mirror annotations create additional, aligned instances of primitives.

The annotations marking equal geometric measurements commute with respect to each other, but must be applied before the symmetry-related annotations (alignment and mirror). Connection curve annotations are drawn on primitives' original 2D shapes; they should be applied as early as possible, once primitives' 3D shapes are known. Same-scale and same-tilt annotations cause complex changes to primitives' 3D shapes which would invalidate a previously applied connection curve. Same-length annotations cause comparatively larger changes to primitives' 2D shapes and simpler changes to primitives' 3D shapes. These simpler changes straightforwardly also apply to the depth placement induced by previously applied connection curves.

5.1 Primitives

Generalized cylinders. A generalized cylinder is defined by a 3D spine curve $\gamma(t)$ ($t \in [0,1]$), a cross-section scale function $s(t)$, symmetry directors $d(t)$, and closed, 2D cross-section curves $\alpha_t(u) = (\alpha_t^1(u), \alpha_t^2(u))$ ($u \in [0,1]$). The simplest definition of a generalized cylinder's surface is $c(u,t) = \gamma(t) + s(t)\left(\alpha_t^1(u)d(t) + \alpha_t^2(u)d^{\perp}(t)\right)$, where $d^{\perp}(t)$ is the vector perpendicular to both the spine's tangent and the symmetry director (Figure 19). Unfortunately, if the curvature of $\gamma(t)$ is high compared to the cross-section scale, this definition will cause the surface to pinch and self-intersect. To eliminate this problem, we use integral curves along a *smoothed* distance field [Peng et al. 2004] to offset the cross-sections, instead of straight lines perpendicular to $\gamma(t)$. To obtain the 3D position for a cross-section point $\alpha_t(u)$, we move along the (curved) integral line of the smoothed distance function starting at point $\gamma(t)$, with an initial direction along $\alpha_t(u)^1 d(t) + \alpha_t(u) d^{\perp}(t)$, by distance $|\alpha_t(u)|$. (This computation is described in more detail in [Peng et al. 2004].) Conceptually, this process corresponds to bending and compressing the planes of cross-sections to avoid self-intersections.

The user's out-of-view-plane-tilted cross-sections imply sparse tangent constraints on $\gamma(t)$. Given these constraints and the user-specified 2D spine curve, we find smoothly varying z coordinates for γ, minimizing $\int (\Delta\gamma(t))^2 dt$ with respect to the z coordinate of γ. Similarly, we find a smoothly varying scale function $s(t)$ given the user's sparse scale constraints, as well as smoothly varying symmetry directors $d(t)$ and cross-section curves $\alpha_t(u)$.

Finally, we attach end-caps in a tangent continuous manner. Let e_0 be the end-cap protrusion at $\gamma(0)$. We extend the surface by sweeping the cross-section at the end in the direction tangent to (without loss of generality) $\gamma(0)$; we scale these cross-sections by the y value of a cubic Bézier curve whose tangent at its start is parallel to $(1, s'(0))$ and whose y value falls to 0 at $x = e_0$. The four control points are $(0, s(0)), (\frac{e_0}{2}, s(0) + \frac{s'(0)e_0}{2}), (e_0, \frac{s(0)+s'(0)e_0}{2}), (e_0, 0)$.

Spine deformation is implemented using Laplacian curve editing [Sorkine et al. 2004]. The spine is either deformed directly or, more commonly, during cross-section scale manipulation, when it is constrained to pass through the center of the cross-section. (During cross-section scale manipulation, one of the cross section's silhouette points remains fixed while the other follows the mouse.)

Ellipsoids are implied by the 2D ellipse and the out-of-view-plane tilt of the circular cross-section. (The ellipsoid's two shorter axes have the same length as the ellipse's short axis and span the tilted cross-section's plane; the long axis is perpendicular to the tilted cross-section with length such that the ellipsoid's silhouette projects to the 2D ellipse.)

5.2 Annotations

Connection curves are user-drawn 2D curves $\beta(t)$ ($t \in [0,1]$) whose projection onto the surfaces of two shapes we wish to be identical; in other words, we wish for the shapes to intersect each other along the projection of β. Let P_f and Q_f be functions which project a point in the image plane to a point on the front of the first and second shapes, respectively. Allowing for translation of the surfaces in z (our depth coordinate), we minimize

$$\int_0^1 \left[P_f(\beta(t))_z - Q_f(\beta(t))_z + c\right]^2 dt$$

with respect to c, the relative offset between the two shapes. We assume that Q has already been fixed in z and translate P by c in the z direction. When dragging the intersection curve, $c = Q_f(a)_z - P_f(a)_z$, where a is the image-plane point under the mouse. A depth-first search is used to ensure that we only translate each shape in depth once. We do not support multiply-connected graphs of primitives.

Mirror annotations. To implement mirror annotations, we duplicate and then reflect the 3D shape of a primitive n (and those of its attached primitives) across the symmetry plane of another primitive m. In case of a symmetry sheet, we find the closest point on the sheet to n's attachment origin and use the tangent plane there as the symmetry plane. A primitive's *attachment origin* is either its center or, in the case of a generalized cylinder, one of the 3D endpoints of its spine, whichever is closest (in 2D) to the connection curve attaching the primitive to m.

Alignment annotations. To implement alignment annotations, described in Section 4.1, we find the smallest satisfying translations in a least squares sense. In case the alignment is with respect to a symmetry sheet, we simply average the tangent planes for each to-be-aligned primitive's attachment origin, just as with mirror annotations.

Same scale, tilt, and length annotations. The congruency annotations are all implemented similarly. A same-scale annotation marks a set of cross-sections as having the same scale. To satisfy this constraint, every cross-section in the set is simply assigned the average scale of the entire set. Same tilt angle annotations are implemented similarly, averaging angles. Same length annotations are also implemented similarly; changing the length of a 3D spine curve or ellipsoid axis simply scales the 2D spine curve or corresponding 2D ellipse axis. However, when scaling a primitive, attached primitives are also translated so as to remain connected.

6 Results

We have used our interface to generate models from a variety of found 2D images. These are shown in Figures 20 and 21 and in the accompanying video. A typical modeling session lasted less than ten minutes (longer for the more complex models shown in Figure 21). Results created by user testers are presented in Section 7. Source material included drawings from a children's book, user-created 2D drawings, concept artwork, and cartoons.

Models creating with our system contain structural information useful in a variety of applications. This information includes spines, the location of joints, and a mesh whose parts are segmented into distinct connected components. Furthermore, annotations encode symmetries in the model and lengths which should remain equal when, e.g. deforming the model. One use for symmetry information is automatic propagation of mesh refinement performed in a surface editing tool such as a 3D sculpting tool. The spines and joints can be used for animation.

7 Evaluation

We have found it easy to create a variety of models consisting of rotund and tubular areas, and, through the use of non-circular cross sections, flat shapes such as ears. When tilting cross section, we find it necessary to verify the results in the 3D view. This is consistent with [Koenderink et al. 1992], in which humans are shown to make (consistent) errors when estimating the magnitude of surface slopes. (It is perhaps worth noting that tilting circular cross sections is reminiscent of the gauge figures adjusted by subjects in the experiment described in [Koenderink et al. 1992].)

We performed a small, informal user study, consisting of seven people, three of whom had 2D artistic experience. None of the subjects had significant 3D modeling experience, but five were familiar with 3D manipulation concepts. After a 15 minute training session, users were able to create their own models, several of which are shown in Figure 22. (One user struggled with a drawing whose point-of-view aligned with primitives' spines; this limitation is discussed in Section 8, and the drawing can be found in the auxiliary materials.) Users unfamiliar with 3D manipulation concepts were significantly slower, however, but reported feeling more comfortable over time.

We also performed a comparison study, consisting of seven people, between our system and FiberMesh [Nealen et al. 2007]. FiberMesh was modified to display an underlying guide image and to have a second side-by-side view; both views had a button to reset the view parameters. Subjects were asked to create a 3D model from a 2D illustration in each system and to work until satisfied. All subjects received the same image. Half were randomly assigned to use our system first. Before using each system, subjects were given 15 minutes of training, which consisted of a brief video and hands-on, guided experimentation. Throughout the study, subjects were encouraged to "think out loud." The example illustration and data collected from the comparison study can be found in the auxiliary materials.

Most subjects had casual 2D drawing experience, were familiar with 3D manipulation concepts, and had no 3D modeling experience. On average, subjects spent 29 minutes in our system and 31 minutes in FiberMesh until task completion. When asked to state an overall preference, five subjects preferred our system, one preferred FiberMesh, and one reported no overall preference. Several subjects noted their satisfaction at being able to directly trace silhouettes in FiberMesh, but found the depth placement and orientation difficult to control for all but the initial shape. Multiple subjects remarked that the symmetry-related annotations were a great benefit and that they would have liked to have had them in FiberMesh.

Our informal and comparison studies found that, overall, users reported satisfaction at creating 3D models from their sketches or illustrations, and comfort with our primitives which resembled shapes in 2D drawing programs. Users reported understanding and liking tilting cross sections, but noted that they verified their manipulations in the 3D results window. (In the comparison study, some users did not tilt any cross sections outside of training.) Connection curves were similarly received, but many users would have benefited from a more simply mapped depth control (e.g. mapping

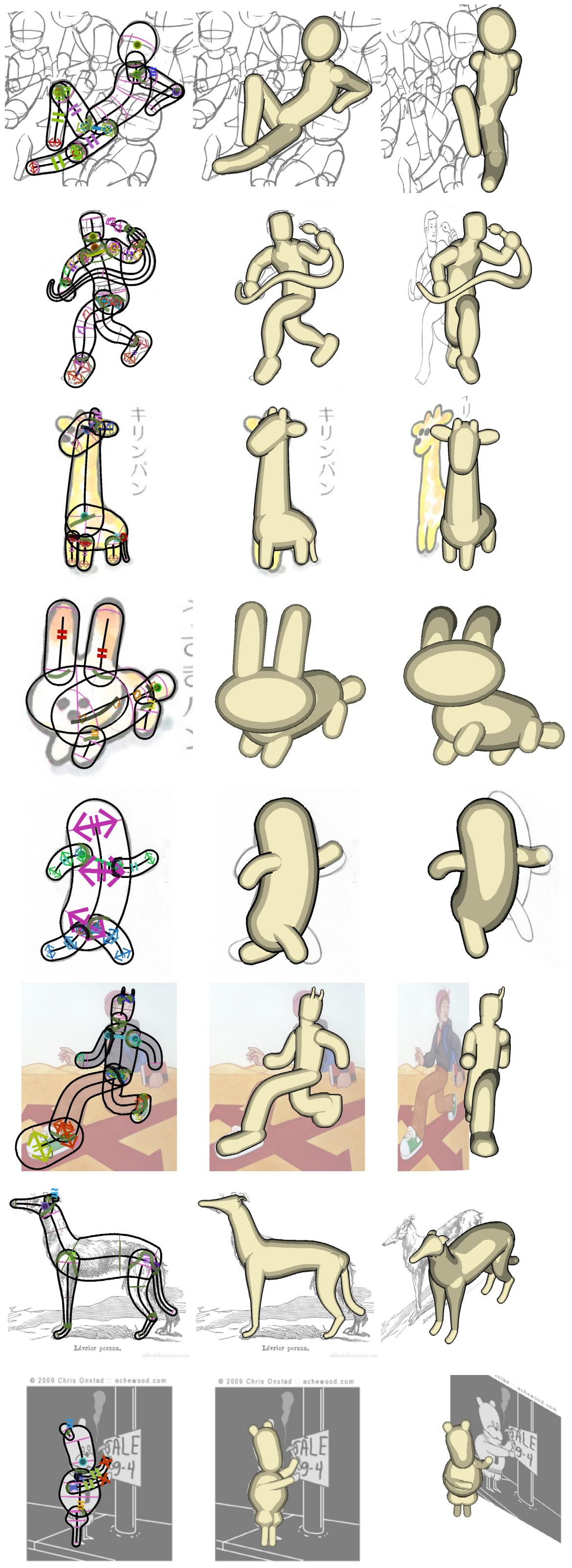

Figure 20: *Models created using our interface. These models took an average of 15 minutes to create. Primitives and annotations are shown on the left, and the resulting model from the same angle and a different angle is shown in the middle and on the right. The first six source images are [Vilppu 1997], © Alex Rosmarin, © Satoshi Kako, © Satoshi Kako, © Alex Rosmarin, and © Alex Rosmarin. The last source image is © Chris Onstad.*

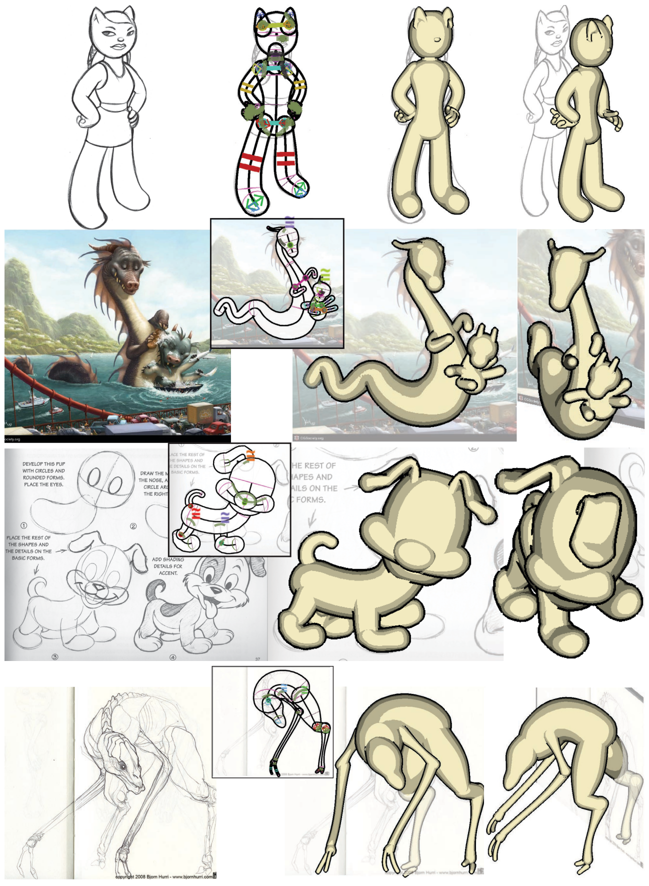

Figure 21: *More complex models created using our interface. These models took on the order of 30 minutes to create. Generalized cylinders were used everywhere except the characters' noses in rows 1 and 3, where ellipsoids were used. In each row: the guide image; the primitives and annotations; the resulting model from the same and a different angle. The guide images in each row are © Alex Rosmarin; © Kei Acedera, Imaginism Studios 2008; [Blair 1994]; and © Björn Hurri,* `www.bjornhurri.com`.

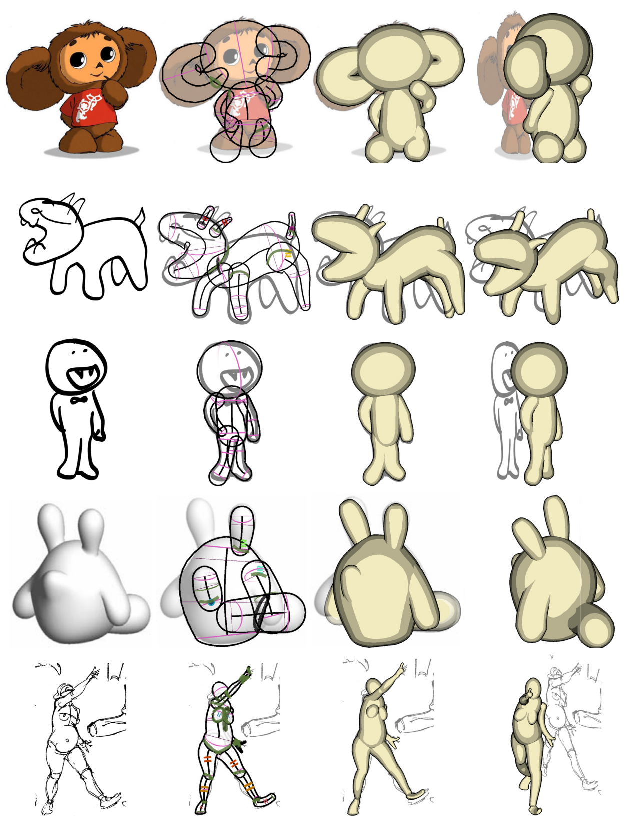

Figure 22: *Models created by first-time users. The primitives and annotations are shown on the left, and the resulting model from the same angle and a different angle is shown in the middle and on the right. From top to bottom: a cartoon character (20 minutes); a monster (10 minutes); a vampire (10 minutes); a cartoon character from [Nealen et al. 2007] (20 minutes); a figure from [Vilppu 1997] (30 minutes). The monster and vampire images were drawn by users.*

the mouse's y position to depth). All annotations were used during testing, but not uniformly. Some users preferred to adjust primitives manually until they achieved the desired result. Most users appreciated the symmetry-related annotations (mirroring and alignment) for their efficiency (relative to duplicating effort and manual tweaking).

8 Conclusion, Limitations, and Future Work

We have presented an interface for 3D modeling based on the idea of annotating an existing 2D sketch. Although many 2D drawings have no consistent 3D representation (Section 3), with a small degree of training, even novices are able to create 3D models from 2D drawings. Our interface eliminates the need for constant rotation inherent to many previous sketch-based modeling tools, which precludes users from matching their input to a guide image. Our primitives and annotations are structured and persistent, and provide semantic information about the output models useful in a variety of applications. An additional benefit of our approach is that the entire modeling process is visible in a single, static 2D image, which makes it easy to explain and learn how to create a given model.

While we are able to create a variety of models using our existing primitives and annotations, our approach is not without its limitations. First, our existing primitives are limited to free-form surfaces. We cannot model surfaces with edges or relatively flat surfaces. Second, our interface cannot be used when a drawing's point-of-view aligns with primitives' spines or long axes, projecting them to (or nearly to) a single point. Third, our interface is not appropriate for adding fine-scale details. For this task, models created in our system can be refined with a displacement editor. Fourth, it is not possible to do away with a 3D view altogether. The 3D view is necessary for verification; it is consulted interactively when manipulating out-of-image-plane tilt handles and connection curve annotations. Fifth, our interface has operations which feel like modeling rather than sketching, and it takes some training to learn the 2D-to-3D mapping. Sixth, we provide no way to color or texture 3D models, even though drawings may have been colored. Seventh, we do not allow cycles of connection curves, which are useful in some cases (for example, Figures 20, second row, and 21, top row).

We see our current interface as an initial step in the direction of structured, two-dimensional 3D modeling. In the future, we plan to create primitives and annotations for precise CAD modeling, for other commonly used free-hand drawing primitives, for relatively flat shapes, and for additional geometric relationships. We also foresee annotations for specific applications, such as annotating primitives' motions for animation and rigidity for deformation. We would like to assist users matching curves in an image, as in [Tsang et al. 2004]. We would also like to color or texture primitives based on the underlying image. Our system could be extended to add geometry to an existing 3D model, by sketching over the model on a 2D overlay plane and then placing primitives and annotations on the overlaid sketch. In some cases, support for oblique projections would allow for alternative, possibly less surprising 3D interpretations of user input. Finally, a comprehensive user study comparing our system to many other systems, with a large number of users at multiple skill levels, remains to be performed.

References

ADOBE, 2007. Illustrator. http://www.adobe.com/products/illustrator/.

AGRAWALA, M., ZORIN, D., AND MUNZNER, T. 2000. Artistic multiprojection rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques*, 125–136.

ANDRE, A., SAITO, S., AND NAKAJIMA, M. 2007. CrossSketch: Freeform surface modeling with details. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBM)*, 45–52.

AUTODESK, 2009. Maya. http://www.autodesk.com/maya.

BAE, S.-H., BALAKRISHNAN, R., AND SINGH, K. 2008. ILoveSketch: As-natural-as-possible sketching system for creating 3D curve models. In *Proceedings of the ACM symposium on User interface software and technology (UIST)*, 151–160.

BLAIR, P. 1994. *Cartoon Animation*. Walter Foster, Laguna Hills, California.

BLOOMENTHAL, J., Ed. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann, San Francisco, California.

CABRAL, M., LEFEBVRE, S., DACHSBACHER, C., AND DRETTAKIS, G. 2009. Structure preserving reshape for textured architectural scenes. *Computer Graphics Forum 28*, 2, 469–480.

CHEN, X., KANG, S. B., XU, Y.-Q., DORSEY, J., AND SHUM, H.-Y. 2008. Sketching reality: Realistic interpretation of architectural designs. *ACM Transactions on Graphics 27*, 2, 11.

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

CORDIER, F., AND SEO, H. 2007. Free-form sketching of self-occluding objects. *IEEE Computer Graphics and Applications 27*, 1, 50–59.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of ACM SIGGRAPH*, 11–20.

DORSEY, J., XU, S., SMEDRESMAN, G., RUSHMEIER, H., AND MCMILLAN, L. 2007. The mental canvas: A tool for conceptual architectural design and analysis. In *Proceedings of Pacific Graphics*, 201–210.

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

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3D freeform design. In *Proceedings of ACM SIGGRAPH*, 409–416.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics 24*, 3, 1134–1141.

INFOGRAFICA, R., 2009. MetaReyes. http://www.reyes-infografica.com/plugins/meta.php.

KALLIO, K. 2005. 3D6B editor: Projective 3D sketching with line-based rendering. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBM)*, 73–80.

KARPENKO, O. A., AND HUGHES, J. F. 2006. SmoothSketch: 3D free-form shapes from complex sketches. *ACM Transactions on Graphics 25*, 3, 589–598.

KOENDERINK, J. J., VAN DOORN, A. J., AND KAPPERS, A. M. L. 1992. Surface perception in pictures. *Perception & Psychophysics 52*, 5, 487–496.

LORAN, E. 1943. *Cezanne's Composition*. University of California Press.

MICROSOFT, 2003. Office. http://office.microsoft.com.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. FiberMesh: Designing freeform surfaces with 3D curves. *ACM Transactions on Graphics 26*, 3, 41.

OLSEN, L., SAMAVATI, F. F., COSTA SOUSA, M., AND JORGE, J. 2008. A taxonomy of modeling techniques using sketch-based interfaces. In *Eurographics State of the Art Reports*.

PENG, J., KRISTJANSSON, D., AND ZORIN, D. 2004. Interactive modeling of topologically complex geometric detail. *ACM Transactions on Graphics 23*, 3, 635–643.

PRADOS, E. 2004. *Application of the theory of the viscosity solutions to the Shape From Shading problem*. PhD thesis, University of Nice-Sophia Antipolis.

SCHMIDT, R., ISENBERG, T., JEPP, P., SINGH, K., AND WYVILL, B. 2007. Sketching, scaffolding, and inking: A visual history for interactive 3D modeling. In *Proceedings of NPAR*, 23–32.

SCHMIDT, R., SINGH, K., AND BALAKRISHNAN, R. 2008. Sketching and composing widgets for 3D manipulation. *Computer Graphics Forum 27*, 2, 301–310.

SINHA, S. N., STEEDLY, D., SZELISKI, R., AGRAWALA, M., AND POLLEFEYS, M. 2008. Interactive 3D architectural modeling from unordered photo collections. *ACM Transactions on Graphics 27*, 5, 159.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of Eurographics/ACM SGP*, 175–184.

THORMÄHLEN, T., AND SEIDEL, H.-P. 2008. 3D-modeling by ortho-image generation from image sequences. *ACM Transactions on Graphics 27*, 3, 86.

TSANG, S., BALAKRISHNAN, R., SINGH, K., AND RANJAN, A. 2004. A suggestive interface for image guided 3D sketching. In *Proceedings of ACM SIGCHI*, 591–598.

VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., WARD, B., AND TORR, P. H. S. 2007. VideoTrace: Rapid interactive scene modelling from video. *ACM Transactions on Graphics 26*, 3, 86.

VARLEY, P., AND COMPANY, P. 2007. Sketch input of 3D models: Current directions. In *VISAPP 2007: 2nd International Conference on Computer Vision Theory and Applications*, 85–91.

VILPPU, G. 1997. *Vilppu Drawing Manual*. Vilppu Studio, Acton, California.

WU, T.-P., TANG, C.-K., BROWN, M. S., AND SHUM, H.-Y. 2007. ShapePalettes: Interactive normal transfer via sketching. *ACM Transactions on Graphics 26*, 3, 44.

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

ZHANG, L., DUGAS-PHOCION, G., SAMSON, J.-S., AND SEITZ, S. M. 2001. Single view modeling of free-form scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 990–997.