DOI: 10.1111/cgf.12044
EUROGRAPHICS 2013 / I. Navazo, P. Poulin
(Guest Editors)
Volume 32 (2013), Number 2

Geosemantic Snapping for Sketch-Based Modeling

A. Shtof[1] and A. Agathos[2] and Y. Gingold[3] and A. Shamir[4] and D. Cohen-Or[1]

[1]Tel Aviv University, Tel Aviv, Israel
[2]West University of Timişoara, Timişoara, Romania
[3]George Mason University, Fairfax, USA
[4]The Interdisciplinary Center, Herzliya, Israel

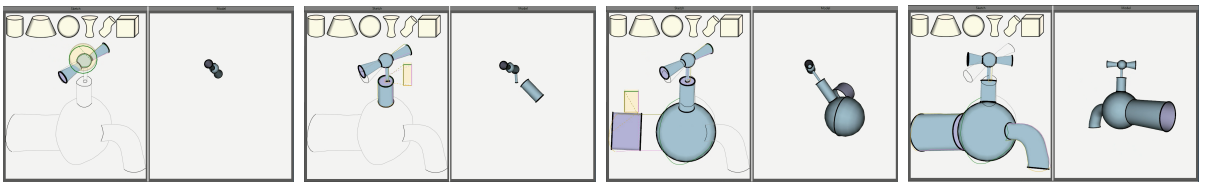

Figure 1: *Using our method, the human modeler simply drags-and-drops the shape primitives into approximate position; the system provides real-time precise geometric snapping feedback and infers geosemantic constraints between the primitives. (Total modeling time: one minute.)*

Abstract

Modeling 3D objects from sketches is a process that requires several challenging problems including segmentation, recognition and reconstruction. Some of these tasks are harder for humans and some are harder for the machine. At the core of the problem lies the need for semantic understanding of the shape's geometry from the sketch. In this paper we propose a method to model 3D objects from sketches by utilizing humans specifically for semantic tasks that are very simple for humans and extremely difficult for the machine, while utilizing the machine for tasks that are harder for humans. The user assists recognition and segmentation by choosing and placing specific geometric primitives on the relevant parts of the sketch. The machine first snaps the primitive to the sketch by fitting its projection to the sketch lines, and then improves the model globally by inferring geosemantic *constraints that link the different parts. The fitting occurs in real-time, allowing the user to be only as precise as needed to have a good starting configuration for this non-convex optimization problem. We evaluate the accessibility of our approach with a user study.*

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

Sketching in two-dimensions remains easier than 3D modeling for professional 3D modelers and novices alike. Professional 3D modelers nearly always begin the modeling process by sketching, either on paper or in a 2D sketching application. Sketching allows artists to focus on creativity rather than technical issues in the early, exploratory stages design. At the same time, novices are capable of sketching, but are unfamiliar with 3D modeling tools. Although 2D sketches are expressive, 3D models allow better understanding of the shape's structure and proportions. It would be nice if simple 3D models could be created easily from 2D sketches for quick previews and as a starting point for further editing and creation of more complex models. This would facilitate the advantages of both techniques for both experts and novices.

Creating a 3D model from a sketch is in general an ill-defined problem it is highly non-linear and non-convex. The challenge can be separated into two orthogonal tasks: Se-

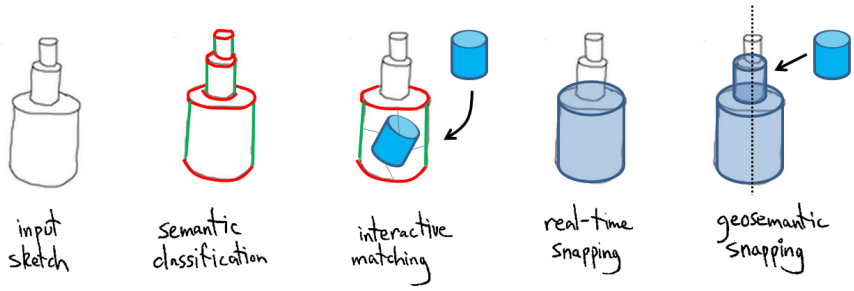

Figure 2: *An overview of our method (from left to right): a **sketch** is loaded into the system and its strokes are **classified** as feature curves or silhouette curves. Then, when a primitive is dragged onto the sketch, automatic **matching** is used to associate it with strokes in the sketch. **Real-time snapping** of the individual primitive refines its position and orientation in the sketch as the user drags. When a primitive is released and other primitives have already been placed, automatic inference of **geosemantic** constraints guides the final snapping.*

mantic and Geometric. Modeling from a sketch requires interpreting and understanding the semantics of the sketch including the segmentation and recognition of its parts. This includes both understanding individual strokes and their combinations representing 3D structures and parts. Second, modeling involves the complex task of fitting and reconstructing the geometry that matches the intended shape and its semantics. The need to interpret the sketch before reconstructing the geometry requires a high-level cognitive endeavor that machines are still weak at compared to humans. Furthermore, fitting primitives is a very difficult optimization problem, as it is non-convex, and strongly depends on a good starting configuration for the optimization. On the other hand, reconstructing a geometry that admits external and internal constraints is a meticulous task that is hard and tedious for humans, and better fits the type of computations machines are designed for.

In this paper we present a semi-automatic interactive system for modeling *simple* 3D models from a sketch, that is carefully designed to minimize user efforts. We do not intend to replace precise tools for editing and modeling 3D geometry; rather, we offer a way to create a 3D preview where the user can rotate the "sketch" and quickly grasp its 3D properties. Our approach is based on identification and decoupling of the semantic and geometric tasks, and converting semantic relations to constraints that are *geosemantic*. We believe that a large number of these can be implied automatically using a simple set of rules. This allows the user to *explicitly* perform only the minimal semantic tasks that the machine is weak at or that cannot be implied. These rules automatically bridge the gap between what the user explicitly provides and what is actually needed to solve the fitting problem and preserve internal structural relations.

The basic interactive modeling operation in our system is an easy-to-use 'drag-and-drop' operations, where primitive shapes are selected and dragged over the sketch to their approximate intended position. The user first loads a sketch, such as one created in illustrator or even a pencil-and-paper scan (see Figure 1). The user then identifies and chooses appropriate 3D primitives that compose the shape, such as cylinders and cones, and drags them to approximate locations on the sketch, while roughly orienting them. Throughout the drag, the system automatically matches curves on the primitive to lines in the sketch and snaps the primitive in real-time for preview. The real-time preview allows the user to be only as precise as is necessary to provide a good starting configuration for the optimization process to correctly fit the primitive.

After dropping the primitive, the system performs the final snapping by fitting the primitive to the sketch while adhering to geosemantic relations with other 3D parts that have already been modeled. For instance, it can detect and impose almost-relations to be precise-relations such as parallelism, collinearity, perpendicularity, and more. Using this procedure the user assists the recognition and segmentation by explicitly defining only *what* approximately goes *where*, while the rest is performed by an automatic, computational process. Despite the non-linearity and non-convexity of the problem, both the preview and the final snapping are performed by a very fast optimization technique using an augmented Lagrangian method.

2. Related Work

Image-Based Modeling. The method we present in this paper shares similarities with the approach taken in [GIZ09] and [LSMI10], in which a shape is modeled by the user explicitly placing primitives and specifying constraints in the form of semantic "annotations." Unlike our work, these techniques require manual placement of primitives and annotations.

Tsang et al. [TBSR04] presented a system that assists users in tracing curves from guide images by snapping user input and suggestively completing user input curves with curves

from a database. In our system, users operate at a higher level of abstraction, by dragging and dropping primitives, and our system infers geosemantic constraints.

Given a photograph of an object and a database of 3D models in the object class, Xu et al. [XZZ*11] presented a system for creating a 3D model to match the photograph requiring only minimal user input. In contrast, our system takes a sketch as input—users are modeling rather than reconstructing—and does not require a database of similar models, but just a set of prescribed primitives.

Several systems have explored interactive modeling from multiple photographs or video sequences. In [DTM96, SSS*08, vdHDT*07] users mark edges or polygons or place 3D primitives in multiple photographs or frames. The systems align them and extract 3D positions for the geometry to create textured 3D models. In our approach, there is only a single image of an imprecise sketch, so common vision techniques that assume accurate and consistent input cannot be directly applied.

Primitives and constraints. Modeling with geometric constraints has been a part of graphical design since Sketchpad [Sut63], the first graphical CAD program, and continues to be a part of professional CAD systems (such as AutoCAD, SolidWorks, CATIA, etc.). These systems require substantial training, since users are required to explicitly specify the constraints manually. (Constraints have also been used effectively in more accessible sketch-based modeling systems; see below.) Moreover, these systems provide no direct way to use sketches as part of the creative process. In contrast, our system is accessible to novices, is designed to allow users to model from a 2D sketch, and automatically infers constraints.

The 2D drawing program Pegasus [IKTM98] uses a "predictive" interface, which is reminiscent of our optimistic snapping procedure. The recent GlobFit technique [LWC*11] automatically infers constraints from noisy and incomplete 3D point clouds, which are then used to reconstruct a surface, but cannot be applied to model from 2D sketches. Zeleznik et al. [ZHH96] introduced SKETCH, a 3D modeling system based on the idea of placing primitives by sketching gestures. Unlike our system, SKETCH does not support modeling from a guide image or constraints. Pereira et al. [PJBF03] introduced a "calligraphic" interface for CAD modeling based on sketched commands; their gluing operation would be convenient in our system as well.

Sketch-based modeling. One approach to 3D modeling is the so-called sketch-rotate-sketch workflow. This approach was introduced in Teddy [IMT99] and inspired a large body of follow-up work; see [OSSJ09] for a survey. These approaches are not designed for modeling from an existing sketch, since the viewpoint rotates throughout the modeling process. Our approach is for modeling *from* sketches, not modeling *by* sketching.

Constraints are also employed in many sketch-based modeling systems for beautification or to "regularize" the 2D-to-3D mapping problem. An early example is VIKING [Pug91], in which users "sketch" (in a manner quite different than present sketch-based modeling approaches) and then interactively label and apply constraints to drawings. Another early system for interactively sketching 2D and 3D objects with constraints was introduced in [EHBE97]. Notably, incidence and right-angle constraints for strokes in 2D are automatically applied. More recent sketch-based modeling systems that employ constraints include iLoveSketch [BBS08] and Analytic Drawing [SKSK09]. Because the viewpoint rotates in these systems, however, users can not model from an existing sketch.

Sketch recognition techniques are designed to convert a given 2D line drawing into a 3D solid model. 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. Lipson and Shpitalni [LS96] introduced an early work in this area (as well as many later ones); SMARTPAPER [SC04] extends their algorithm in a variety of ways, including the ability to sketch over existing 3D models. For a recent survey of line-drawing interpretation algorithms, see [Coo08]. In a related approach, Chen et al. [CKX*08] allows for imprecise, sketched input by matching input to a domain-specific database of architectural geometry. (See ChemInk [OD11] and MathPad^2 [LZ04] for examples of sketch-recognition approaches applied to domains other than geometric modeling.)

3. Overview

The basic operation in our system is the snapping of 3D primitives being dragged-and-dropped over a sketch. This snapping places the primitive in 3D space, gradually constructing a 3D representation of the object one primitive at a time. We use a set of parametric 3D primitives—generalized cylinders, boxes, and spheres—whose parameter values are optimized during snapping, subject to constraints. The objective function fits the projection of the primitive to certain 2D curves of the sketch. The constraints involve both the internal structure of the primitive (*internal structure constraints*) and its relationship with other previously placed primitives (*geosemantic constraints*). With this mode of interaction, the system does not perform any recognition or segmentation of the sketch—the user chooses and places the primitives herself. However, by fitting primitives to the sketch and automating geosemantic constraint inference, the system provides an easy-to-use interface for constructing a 3D object from a sketch.

An overview of the workflow of our system is presented in Figure 2. First, a sketch is loaded into the system. We assume that the input is a 2D *orthographic* sketch defined by a set of parametric stroke curves. The conversion of a raster

sketch into a vector sketch with clean stroke curves is beyond the scope of our paper (see, for example, [NHS*12]). These curves are separated into two semantic types: *feature curves* and *silhouette curves*. Feature curves are non-view-dependent curves of the 3D object (the top and bottom circles of a cylinder, the outline of a sphere, and the 12 edges of a box), and silhouette curves are view-dependent curves corresponding to the boundaries of the object in the sketch (the sides of a cylinder; see, e.g., [CGL*08]). This semantic task is performed semi-automatically: the user marks some curves, and our system propagates tags according to the heuristic that curves likely belonging to a cylinder should alternate between silhouette and feature tags, while those of a box and sphere should not (over 80% success rate in our examples). This classification is very easy for the user to provide and greatly increases the success rate of the automatic curve matching described below.

Once the strokes of a sketch are classified, the user picks a primitive that can be scaled, rotated, and dragged over the sketch. The system performs an automatic, real-time matching between the silhouette (resp. feature) curves of the primitive and the silhouette (resp. feature) curves of the sketch. Whenever a valid match is found, real-time snapping is performed. This real-time snapping uses only the fitting objective function and internal structure constraints. When the user is satisfied with the fit, she releases the primitive (ends the drag) and a full optimization, using the fitting objective function, internal structure constraints, and geosemantic constraints, is performed.

Primitives. Our primitives are boxes (rectangular cuboids), spheres, and various generalizations of cylinders: standard cylinders defined by an axis and radius; truncated cones defined by an axis and radii at either end; "straight" generalized cylinders defined by an axis along which the radius varies; and "bent" generalized cylinders defined by a space curve along which the radius varies. Our approach can be extended to additional primitive types; doing so requires defining the new primitive type's feature curves, which are used to infer and impose geosemantic constraints, and a suitable objective function for fitting the new primitive type to matched sketch curves (Section 5).

4. Curve Matching

Recall that the input to our system is a 2D sketch defined by a set of parametric stroke curves. Prior to modeling, the user classifies these strokes into silhouette and feature curves. At this time, our system also computes the per-pixel distance to each stroke curve, (Figure 3, left) resamples every stroke curve uniformly in arc length, and ignores feature curves to which a reasonable ellipse fit cannot be found.

As the user drags a primitive over the sketch, the system matches silhouette and feature curves from the primitive to silhouette and feature curves of the sketch. This matching

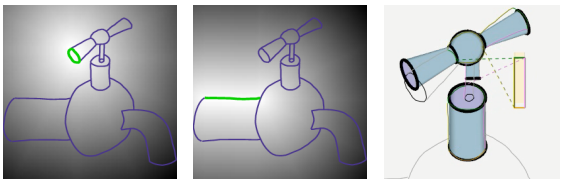

Figure 3: *Left: examples of the per-pixel distance field to the two highlighted strokes. Right: an example of bad snapping when primitive curves are matched to incorrect sketch curves.*

occurs in real-time and is described as follows. We begin by solving the bipartite graph matching problem between silhouette (resp. feature) primitive curves as the nodes of one side of the bipartite graph and silhouette (resp. feature) sketch curves as the nodes of the other side; the weight on each edge is the integral of the per-pixel distances from the primitive curve to the sketch curve [Kuh55]. However, there are many cases where this matches primitive curves to inconsistent sketch curves, which leads to an erroneous snap (see Figure 3, right). Although this could be resolved by the user positioning the primitive more accurately, our goal is to minimize manual and tedious human operations. So, we perform additional, primitive-specific filtering on the bipartite match. (Spheres contain only a single curve, so there is no further processing to be performed.)

For cylindrical primitives, which are composed of two (circular) feature curves and two silhouette curves, we select the better (lower weight) silhouette curve match and search, in the sketch, for the closest feature curve near each end-point. If nearby feature curves are found, we search for the closest additional silhouette curve near either feature curve and use as our resulting match the two silhouette curves and one or two feature curves. Otherwise, our resulting match is the two silhouette curves found by the bipartite matching. (Bent generalized cylinders cannot snap unless both silhouette and feature curves are matched.)

The box primitive is composed of twelve feature curves (the edges) that meet in triplets at three corners. We evaluate whether each such triplet of feature curves matches to three feature curves in the sketch forming a "cubic corner" [Coo08]. If multiple such cubic corners are found, the resulting match is the one where the distance from the cubic corner to the corresponding box corner is smallest.

5. Primitive Fitting

Once matching sketch curves are found, we perform an optimization procedure to fit the primitive to the sketch. This produces an initial 3D fit (up to placement in depth) at interactive rates as the user drags the primitive.

We use parametric representations for our primitives: a sphere is defined by a center point and radius; a box by a cor-

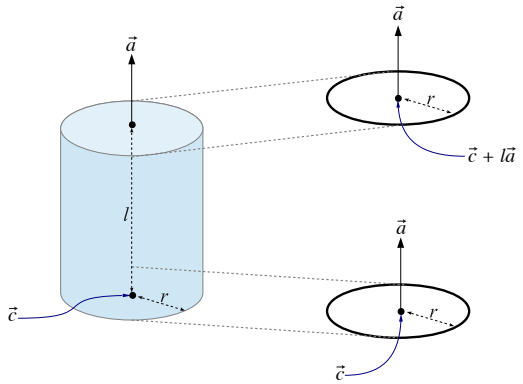

Figure 4: *The feature curves of a cylinder are defined by the cylinder's parameters. When the feature curves participate in geosemantic constraints they affect the primitive they belong to.*

ner point and three orthonormal vectors; straight cylinders by an axis and one, two, or many radii; and bent cylinders by a 3D, planar ploy-line as its spine with a radius at each spine point. These are the degrees-of-freedom in our optimization procedure. Primitives' feature and silhouette curves are also defined in terms of these variables (e.g. Figure 4).

To perform snapping, an objective function ϕ is constructed for the primitive and minimized subject to the primitive's internal structure constraints. Note that as the sketch is 2D in nature, our fitting objective functions are ambiguous up to translation in depth. Primitives are initially centered on the $z=0$ plane. During the final snapping, geosemantic constraints determine the relative placement of primitives.

Spheres. The simplest objective function ϕ_{sphere} fits a sphere's feature curve, a circle with center $c=(c_x,c_y,c_z)$ and radius r, to a 2D sketch curve $S=\{s_i\}$:

$$\phi_{sphere}(p,r)=\sum_{i=1}^{N}[\|c_{xy}-s_i\|^2-r^2]^2$$

(Recall that all sketch curves are sampled uniformly in arc length during preprocessing.)

Cylinders and cones. Fitting cylindrical primitives is based on an objective function that fits their feature curves, 3D circles, to 2D sketch curves. The functional for fitting a 3D circle centered at c with normal direction n and radius r to a 2D sketch curve $S=\{s_i\}$ is

$$\phi_{circle}(c,n,r)=\sum_{i=1}^{N}\left[\|n_z(c_{xy}-s_i)\|^2+\left(n_{xy}\cdot(c_{xy}-s_i)\right)^2-(rn_z)^2\right]^2$$

This equation is based on the observation that a 2D point lies on the projection of a 3D circle if, for some z coordinate, it resides on the plane defined by the circle's center and normal, and the distance from the circle's center equals the radius.

When a cylinder's circular feature curve is not matched to a sketch curve, the following objective function is used, where p and q are the 2D dangling end-points of the sketch silhouette curves matched to the primitive:

$$\phi^0_{circle}(c,n,r)=\left(\|p-c_{xy}\|^2-r^2\right)^2+\left(\|q-c_{xy}\|^2-r^2\right)^2$$

For cylinders and cones, the fitting objective function is simply the average of the fitting objective functions of the two feature curves. That is, for a cone with axis a, bottom center c, radii r_{bottom} and r_{top} and length l, the fitting objective function is

$$\phi_{cone}=\frac{1}{2}\phi_{circle}(c,a,r_{bottom})+\frac{1}{2}\phi_{circle}(c+la,a,r_{top})$$

(A cylinder's fitting objective function is the same as a cone's, with $r_{bottom}=r_{top}$.)

Straight generalized cylinders. A straight generalized cylinder has a more complex objective function due to its varying radii. First, we pre-analyze the matched feature and silhouette curves in the sketch to obtain an estimate of the primitive's bottom p and top q (the 2D centers of ellipses fit to the two feature curves) and radii $\{r^0_i\}$ along the spine (the average distance from the matched sketch silhouette curves to evenly sampled points between p and q in the direction perpendicular to $\overline{pq}$). Then the objective for a straight generalized cylinder with axis a, bottom feature curve center c, length l and radii $\{r_i\}$ is

$$\phi_{sgc}(a,c,l,r)=\left[\begin{array}{l}\frac{1}{2}\phi_{circle}(c,a,r_1)+\frac{1}{2}\phi_{circle}(c+la,a,r_N)\\+\|p-c_{xy}\|^2+\|q-(c+la)_{xy}\|^2\\+\Delta r+\frac{1}{N}\sum_{i=1}^{N}(r_i-r^0_i)^2\end{array}\right]$$

(The Δr term penalizes non-smooth radii.)

The straight cylinder objective functions, ϕ_{cone} and ϕ_{sgc}, are minimized subject to the internal structure constraint $\|a\|^2=1$.

Bent-generalized cylinders. The objective function of bent generalized cylinders also includes a pre-analysis of the matched sketch curves. To generate target spine points, we smooth and resample the two silhouette curves, and then average successive points along them. After an additional smoothing and resampling step, we obtain the target 2D spine points $\{s^0_i\}$. Target radii $\{r^0_i\}$ are obtained by computing the minimum distance of $\{s^0_i\}$ to the silhouette curves. Recall that a bent generalized cylinder has a planar spine curve. Let u and v be orthonormal vectors of the spine curve plane, c be the bottom center point in $\mathbb{R}^3$, a and b be the bottom and top 3D circular feature curve normals in uv coordinates, $\{s_i\}$ be a sequence of points in uv coordinates representing the spine, and r_i be the respective radii. Finally, let $P(s,t)=c+su+tv$ be the function mapping a point in uv coordinates to $\mathbb{R}^3$, and let $V(s,t)=su+tv$ be the function mapping a vector in uv coordinates to $\mathbb{R}^3$. Then the objec-

tive function for the bent generalized cylinders is

$$\phi_{bgc} = \phi_{circle}(P(s_1), V(a), r_1) + \phi_{circle}(P(s_N), V(b), r_N) + \sum_{i=1}^{N} \|P(s_i)_{xy} - s_i^0\|^2 + \Delta r + \sum_{i=1}^{N} (r_i - r_i^0)^2$$

subject to the internal structure constraints $\|u\|^2 = 1$, $\|v\|^2 = 1$, and $u \cdot v = 0$ (basis orthonormality); $\|a\|^2 = 1$ and $\|b\|^2 = 1$ (unit-length feature curve normals); $s_i = 0$ (the "bottom" center is actually the bottom center); and $a \cdot (s_1 - s_2)^{\perp} = 0$ and $b \cdot (s_N - s_{N-1})^{\perp} = 0$ (feature curve normals are tangent to the spine).

Boxes. Recall that the matching function for the feature curves of a box primitive (Section 4) involves the detection of a cubic corner in the sketch. Following Perkins [Per68], we recover an orthonormal basis W_0, H_0, and D_0 and dimensions w_0, h_0, and d_0 from the cubic corner (let c^0 be its 2D position). Then for a box parameterized by orthonormal vectors W, H, and D, dimension scalars w, h, and d, and a central point c, the fitting objective is

$$\phi_{box}(c, W, H, D, w, h, d) = \alpha \left(h_0 d_0 \|W - W_0\|^2 + w_0 d_0 \|H - H_0\|^2 + w_0 h_0 \|D - D_0\|^2 \right) + (w - w_0)^2 + (h - h_0)^2 + (d - d_0)^2 + \|c_{xy} - c^0\|^2$$

subject to the internal structure constraints $||W||^2 = 1$, $||H||^2 = 1$, $||D||^2 = 1$, $W \cdot H = 0$, $W \cdot D = 0$, and $H \cdot D = 0$ to ensure that the resulting basis stays orthonormal. (In all of our examples, $\alpha = 0.1$.)

The optimization algorithm we use to minimize these objective functions is described in Section 7.

6. Geosemantic Relations

Following a line of previous works where semantic relations are inferred and imposed automatically [GSMCO09, LWC*11, XZZ*11, LWC*11], we detect several geometric relations among existing 3D primitives under some tolerance, and convert them to precise constraints. These constraints operate on primitives' feature curves. Geosemantic relations are detected and imposed as constraints only during the final snapping of a new primitive, once the drag is finished. For the purposes of geosemantic relations, the feature curves of boxes are grouped into six faces. We detect and impose the following geosemantic relations.

Parallelism. Two feature curves are parallel. This is applied if the angle between two feature curves' 3D normals is less than 20 degrees. The constraint is that the two normal vectors n_1, n_2 must satisfy $n_1 \times n_2 = 0$.

Orthogonality. Two feature curves are perpendicular. This is applied if the angle between two feature curves' 3D normals is between 70 and 110 degrees. The constraint is that the two normal vectors n_1, n_2 must satisfy $n_1 \cdot n_2 = 0$

Collinear centers. The centers of three or more feature curves lie on the same line. This is applied if the 2D triangle formed by three feature curve centers has an angle greater than 170 degrees. The constraint is that the feature curve centers $c_1, c_2, c_3, \ldots, c_N$ must satisfy $(c_1 - c_i) \times (c_1 - c_{i+1}) = 0$ for all $i > 1$.

Concentric. The centers of two or more feature curves are equal. This is applied if the 2D distance between two feature curve centers is within 5% of the screen width or height. It is also applied if two feature curves are snapped to the same curve on the sketch. The constraint is that the feature curve centers c_1, c_2 must satisfy $c_1 = c_2$ (in 3D).

Coplanar. Two or more feature curves lie in the same plane. This is applied if:

- Two primitive feature curves are matched to the same sketch feature curve.
- Two feature curves are parallel (see above), and the distance d of each feature curve's center to the other feature curves' planes is within 5% of the sum of their sizes, where size is the radius of a circle or one-third the perimeter of a rectangle.
- The 2D projection of a feature curve from the new primitive is contained by the projection of an existing primitive's feature curve. (If this is true for both feature curves of the new primitive, then the bottom-most feature curve is chosen, corresponding to back-to-front modeling order.)

The constraint is that the (two or more) feature curve centers $c_1, c_2, \ldots, c_N$ and corresponding normals $n_1, n_2, \ldots, n_N$ must satisfy $(c_1 - c_i) \cdot n_1 = (c_1 - c_i) \cdot n_i = 0$ for all $i > 1$.

For each of these constraints, we define an objective function ψ for the primitives involved. Note that these are 3D constraints; as such, they determine the relative depth of primitives in 3D. (The entire model is still ambiguous up to translation in depth.) The optimization algorithm uses these constraints along with the 2D fitting constraints for the final optimization stage.

7. Optimization

While dragging a primitive, we optimize only the dragged primitive using its corresponding objective functions ϕ between sketch curves and the primitives' curves and its internal structure constraints. This is performed at interactive rates as only a single primitive is involved and no constraints ψ inferred from geosemantic relations are used. In the final optimization, we minimize the sum of all primitive objective functions subject to all inferred geosemantic relations (and all internal structure constraints). This second optimization is more computationally expensive, but, because it occurs only once the drag is finished, it need not run in real-time.

As our fitting problem is a non-linear, non-convex optimization problem subject to equality constraints, we use an augmented Lagrangian method to solve the optimization problem using a sequence of unconstrained problems [NW06]. Each unconstrained problem is solved using L-BFGS [LN89]. The primary advantage of the augmented Lagrangian method over the squared penalty method is that the solution is less subject to distortions of the objective function caused by large penalty weights. The augmented Lagrangian method never gives the squared penalty term a large weight, instead approximating the Lagrange multipliers that lead to constraint satisfaction.

We use automatic differentiation [GW08] to compute the gradients of our objective functions and constraints. We use reverse-mode automatic differentiation, which allows us to compute gradients extremely efficiently by building an expression tree for each objective function ϕ and constraint ψ. The expression trees are created automatically using our software package which we have made available to others[†]. Automatic, efficient gradient computation allows us to state our objective functions and constraints in a simple and straightforward manner; in fact, this makes it possible for advanced users to extend our system with additional primitives and constraints. (We hope to explore this in the future.)

8. Results

Modeling with our tool is extremely rapid (see Figure 5 and the video materials, which contains unedited modeling sessions). All models took at most several minutes to create. The real-time snapping optimization allows users to wantonly drag primitives and drop them as soon as the fit appears as desired. The user can work at high speed, dynamically adjusting the precision of her drag. The result is that humans spend the vast majority of their time performing high level operations (choosing primitives and dragging them in an approximate manner).

Many geosemantic constraints are correctly inferred. Thresholds for our inference procedures have been chosen conservatively, so the user must manually specify them in some cases. This occurred for only two of our examples, the phone handset and the trumpet. (Manual specification for the phone handset can be seen in the accompanying video at 3:55 m:s.) The annoyance of false positives associated with too-aggressive constraint inference could be mitigated with a suggestive interface; we hope to investigate this in the future.

Note that models sometimes do not adhere closely to the sketch. This is because sketches are typically inconsistent or "impossible", especially when geosemantic constraints such as collinearity, coplanarity, and concentricity are taken into account. (See [SKKS09] for a discussion of inconsistencies in sketches.) Despite the inconsistency of typical sketches, our approach is able to rectify the primitives to produce a plausible and consistent 3D model.

8.1. User Study

To evaluate the accessibility of our approach to novices, we performed a user study with 10 users (5 female/5 male), all of whom were fluent computer users. Users modeled from the following sketches: wine glass, goblet, handset, tap, street lamp, and candelabra (menorah) from Figure 5 top-right, second-row-left, third-row-right; Figure 1; Figure 5 bottom-left, and third-row-left, respectively. Users were given a 20 minute tutorial of the system, and then began modeling with no stated time limit. For each model, users first tagged the sketch (Section 3) and then dragged-and-dropped primitives to create the 3D models.

Eight of the ten users were able to successfully create all seven models; the remaining two users were able to create all but the handset model. The user-created models appear similar to those shown in the figures. User modeling sessions can be seen in the video materials, and timing results are shown in Table 1.

On average, users completed the semi-automatic tagging step in under a minute and none reported the process to be tedious or difficult, so we conclude that the step can be easily learned and is not overly burdensome. We observed that some users under-used the semi-automation: they tagged half of the sketch in a single batch operation, leaving only the remaining half of the sketch to be automatically tagged.

The drag-and-drop modeling took less a minute, average and median, for the simple models (wine glass and goblet) and approximately 5–7 minutes for the more complex models (handset, tap, street lamp, and candelabra). We believe that this validates our approach, as even first-time users were able to create models extremely rapidly. We report, however, that users pointed out annoyances resulting from the "research" quality of our code.

Global optimization, which occurs whenever a primitive is "dropped", takes anywhere from less than a second to a few seconds per step depending on the complexity of the model (see the video materials, which have not been edited to remove wait times). To quantify the portion of time spent waiting for the system versus modeling, we asked an expert user to model the candelabra (Figure 5 third-row-left), which is among the most complex of our models, five times. On average, 28% of the total modeling time was spent waiting (35.4 out of 127.6 seconds, with standard deviation of 3.4 and 7.9 seconds, respectively). For novices, we speculate that the wait time constitutes an even smaller fraction, because their total modeling time is longer than an expert's. None of our novice users reported discomfort due to optimization time.

[†] http://autodiff.codeplex.com/

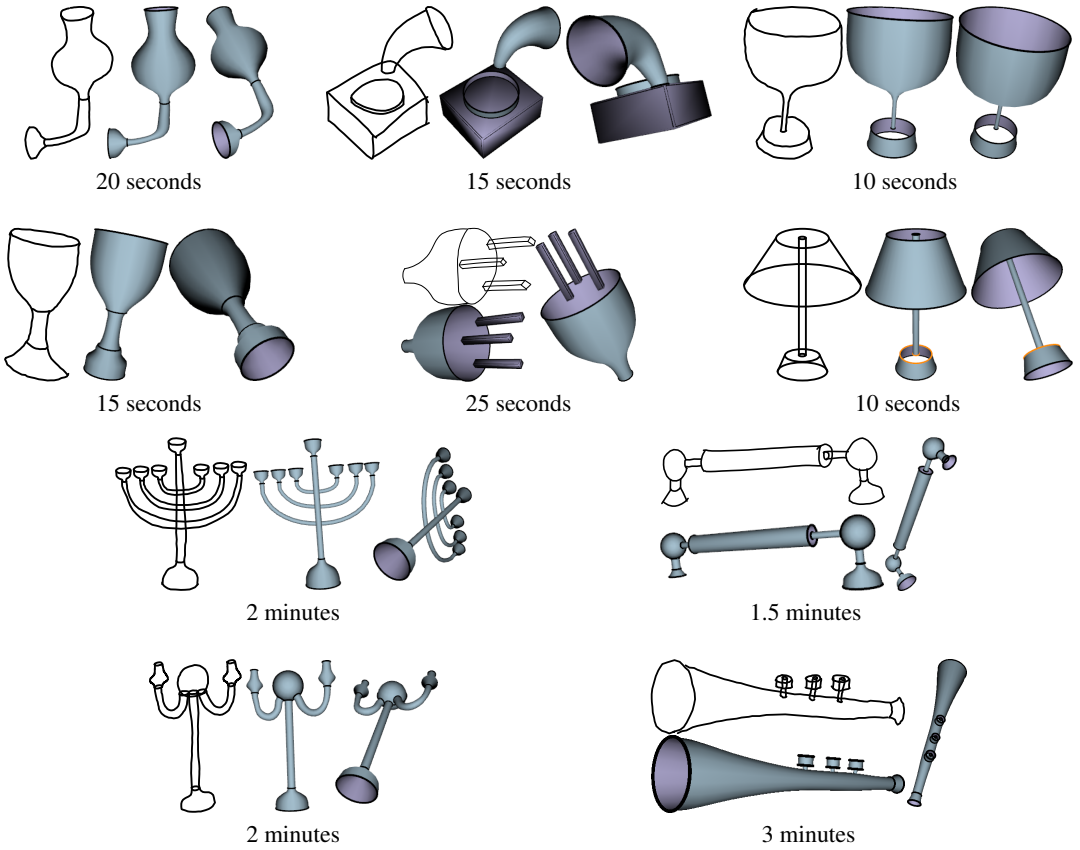

Figure 5: *Modeling results by expert users. Modeling sessions can be seen in the video materials.*

	tagging			modeling		
sketch	average	std. dev.	median	average	std. dev.	median
wine glass	0:23	0:12	0:21	0:40	0:25	0:33
goblet	0:21	0:09	0:20	0:44	0:20	0:44
handset	0:48	0:14	0:48	6:49	1:28	6:22
tap	1:03	0:32	1:00	6:05	1:42	5:37
street lamp	0:53	0:27	0:50	6:05	2:27	5:05
candelabra	1:04	0:37	0:49	6:38	2:55	5:04

Table 1: *Timing results from the user study. Times are expressed in minutes:seconds.*

9. Conclusions

We have made a highly non-linear and non-convex optimization problem tractable by defining a flexible collection of parameterized primitives, designing a simple user interface around providing a good starting point for an augmented Lagrangian optimization. The key is snapping geometric primitives guided by the sketch, together with an inference mechanism that detects and imposes beautifying geosemantic relationships among a collection of primitives. By merely requiring the user to drag-and-drop primitives over the sketch, we have separated that which is easy for humans and challenging for machines, from that which is easy for machines and challenging for humans. The machine does not have to identify the primitives or categories of shapes that belong in the sketch; it has only to infer and enforce geometric relationships and precisely align shapes to match the sketch.

Limitations and Future Work. Our system is currently limited to and geared towards mechanical parts composed of various kinds of generalized cylinders, spheres, and boxes. This includes most shapes whose parts can be created on a lathe or out of boxes, as well as shapes with "bent spines." This does not however, include polyhedra other than cubes, or tubular shapes with non-circular cross sections. While this could include characters, or any models that merit a skeletonization, our geosemantic constraints are not designed for naturally-posed characters. In the future, we plan to support additional, possibly more complex, primitives and geosemantic constraints in order to support a wider range of models. We would also like to support a wider range of sketches. At present, sketched occlusions present a challenge to our system, and may be impossible (Figure 6) or require the user to manually annotate geosemantic relations on the occluded feature curves. We would also like to be able to operate on raster sketches and without requiring the user to tag any strokes as silhouettes or feature curves. Finally, we wish

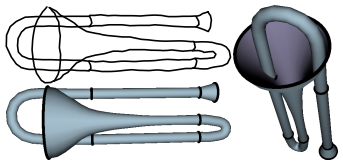

Figure 6: *This trombone sketch cannot be properly modeled in our system due to the sketched occlusion.*

to explore techniques to speed the global optimization procedure involving all primitives, which currently can become lengthy with a large number of primitives and geosemantic constraints.

Acknowledgments. This research was supported in part by the Israel Science Foundation (grant no. 324/11), Sloan Foundation, NSF (CAREER Award CCF-06-43268 and grants IIS-09-16129, IIS-10-48948, IIS-11-17257, CMMI-11-29917, IIS-09-16845), and generous gifts from Adobe, Autodesk, Intel, mental images, NVIDIA, Side Effects Software, and the Walt Disney Company.

References

[BBS08] BAE S.-H., BALAKRISHNAN R., SINGH K.: ILoveSketch: As-natural-as-possible sketching system for creating 3D curve models. In *Proceedings of ACM UIST* (2008), pp. 151–160. 3

[CGL*08] COLE F., GOLOVINSKIY A., LIMPAECHER A., BARROS H. S., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S.: Where do people draw lines? *ACM Transactions on Graphics (Proc. SIGGRAPH) 27*, 3 (Aug. 2008). 4

[CKX*08] CHEN X., KANG S. B., XU Y.-Q., DORSEY J., SHUM H.-Y.: Sketching reality: Realistic interpretation of architectural designs. *ACM Trans. Graph. 27*, 2 (2008), 11. 3

[Coo08] COOPER M. C.: *Line Drawing Interpretation*. Springer, 2008. 3, 4

[DTM96] DEBEVEC P. E., TAYLOR C. J., MALIK J.: Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of ACM SIGGRAPH* (1996), pp. 11–20. 3

[EHBE97] EGGLI L., HSU C.-Y., BRUDERLIN B. D., ELBER G.: Inferring 3D models from freehand sketches and constraints. *Computer-Aided Design 29*, 2 (February 1997), 101–112. 3

[GIZ09] GINGOLD Y., IGARASHI T., ZORIN D.: Structured annotations for 2D-to-3D modeling. *ACM Trans. Graph. 28*, 5 (2009), 148. 2

[GSMCO09] GAL R., SORKINE O., MITRA N., COHEN-OR D.: iWIRES: An analyze-and-edit approach to shape manipulation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 28*, 3 (2009), 33:1–33:10. 6

[GW08] GRIEWANK A., WALTHER A.: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial Mathematics, 2008. 7

[IKTM98] IGARASHI T., KAWACHIYA S., TANAKA H., MATSUOKA S.: Pegasus: A drawing system for rapid geometric design. In *Proceedings of ACM SIGCHI* (1998), pp. 24–25. 3

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3D freeform design. In *Proceedings of ACM SIGGRAPH* (1999), pp. 409–416. 3

[Kuh55] KUHN H. W.: The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2* (1955), 83–97. 4

[LN89] LIU D. C., NOCEDAL J.: On the limited memory BFGS method for large scale optimization. *Mathematical Programming 45* (1989), 503–528. 10.1007/BF01589116. 7

[LS96] LIPSON H., SHPITALNI M.: Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design 28*, 8 (1996), 651–663. 3

[LSMI10] LAU M., SAUL G., MITANI J., IGARASHI T.: Modeling-in-Context: User design of complementary objects with a single photo. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM)* (2010), pp. 17–24. 2

[LWC*11] LI Y., WU X., CHRYSATHOU Y., SHARF A., COHEN-OR D., MITRA N. J.: GlobFit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph. 30* (2011), 52:1–52:12. 3, 6

[LZ04] LAVIOLA JR. J. J., ZELEZNIK R. C.: MathPad^2: a system for the creation and exploration of mathematical sketches. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 432–440. 3

[NHS*12] NORIS G., HORNUNG A., SIMMONS M., SUMNER R., GROSS M.: Topology-driven vectorization of clean line drawings. *ACM Trans. Graph. 31*, accepted (2012). 4

[NW06] NOCEDAL J., WRIGHT S. J.: *Numerical Optimization*. Springer, 2006. 7

[OD11] OUYANG T. Y., DAVIS R.: ChemInk: a natural real-time recognition system for chemical drawings. In *Intelligent User Interfaces (IUI)* (2011), pp. 267–276. 3

[OSSJ09] OLSEN L., SAMAVATI F., SOUSA M., JORGE J.: Sketch-based modeling: A survey. *Computers & Graphics 33* (2009), 85–103. 3

[Per68] PERKINS D. N.: *Cubic Corners*. Tech. Rep. 89, MIT Research Laboratory of Electronics, April 15 1968. 6

[PJBF03] PEREIRA J. A. P., JORGE J. A., BRANCO V. A., FERREIRA F. N.: Calligraphic interfaces: Mixed metaphors for design. In *Interactive Systems: Design, Specification, and Verification*. 2003, pp. 154–170. 3

[Pug91] PUGH D.: *Interactive Sketch Interpretation Using Arc-labeling and Geometric Constraint Satisfaction*. No. v. 91-181 in Technical Report. School of Computer Science, Carnegie Mellon University, 1991. 3

[SC04] SHESH A., CHEN B.: SMARTPAPER: An interactive and user friendly sketching system. *Computer Graphics Forum 23*, 3 (2004), 301–310. 3

[SKKS09] SCHMIDT R., KHAN A., KURTENBACH G., SINGH K.: On expert performance in 3D curve-drawing tasks. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM)* (2009), pp. 133–140. 7

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

[SSS*08] SINHA S. N., STEEDLY D., SZELISKI R., AGRAWALA M., POLLEFEYS M.: Interactive 3D architectural modeling from unordered photo collections. *ACM Trans. Graph. 27*, 5 (2008), 159. 3

[Sut63] SUTHERLAND I. E.: *Sketchpad: A man-machine graphical communication system*. PhD thesis, Massachusetts Institute of Technology, 1963. 3

[TBSR04] TSANG S., BALAKRISHNAN R., SINGH K., RANJAN A.: A suggestive interface for image guided 3D sketching. In *Proceedings of ACM SIGCHI* (2004), pp. 591–598. 2

[vdHDT*07] VAN DEN HENGEL A., DICK A., THORMÄHLEN T., WARD B., TORR P. H. S.: VideoTrace: Rapid interactive scene modelling from video. *ACM Trans. Graph. 26*, 3 (2007), 86. 3

[XZZ*11] XU K., ZHENG H., ZHANG H., COHEN-OR D., LIU L., XIONG Y.: Photo-inspired model-driven 3D object modeling. *ACM Trans. Graph. 30* (Aug. 2011), 80:1–80:10. 3, 6

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: SKETCH: An interface for sketching 3D scenes. In *Proceedings of ACM SIGGRAPH* (1996), pp. 163–170. 3