2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling (2005), pp. 1–10

Sketch-based Modeling of Parameterized Objects

Chen Yang Dana Sharon Michiel van de Panne

University of British Columbia†

Abstract

Sketch-based modeling holds the promise of making 3D modeling accessible to a significantly wider audience than current modeling tools. We present a modeling system that is capable of constructing 3D models of particular object classes from 2D sketches. The core of the system is a sketch recognition algorithm that seeks to match the points and curves of a set of given 2D templates to the sketch. The matching process employs an optimization metric that is based on curve feature vectors, and the search space of possible correspondences is restricted by encoding knowledge about relative part locations into the 2D template. Once a best-fit template is found, a 3D object is constructed using a series of measurements that are extracted from the labelled 2D sketch. We apply our sketch-recognition and modeling algorithms to sketches of cups and mugs, airplanes, and fish. The system allows non-experts to use drawings to quickly create 3D models of specific object classes.

Keywords: sketch-based modeling, sketch recognition, geometric modeling

1. Introduction

Consumers now have the ability to render 3D content on almost every computer and a growing number of portable devices. However, the capabality for non-experts to create 3D content has not kept pace. Even those with in-depth knowledge of computer graphics will often go to considerable lengths to find existing 3D models before resorting to building a 3D model of their own. This can be seen as an interface problem — if we were to imagine describing the shape of a mug to a colleague, we could imagine using hand gestures or a hand-drawn sketch to communicate the object shape, to a first approximation. However, developing 3D modeling systems to support this type of input is a daunting task. For example, inferring the 3D shape of objects from line drawings is a well-studied and largely unsolved computer vision problem.

The system we present builds 3D models from hand-drawn sketches. To make the problem tractable, we make a number of assumptions. First, we equip the system with a priori knowledge about particular classes of objects that we expect to be drawn using the system. This knowledge is encoded in terms of: (a) a flexible 2D template that can be matched to the sketch, and (b) a procedure for building the 3D model from the 2D template. Second, we require that the user draw in a fashion that is not overly sloppy, and require that the user have familiarity with the general structure of the underlying template. Our system does not solve the general problem of building 3D models from

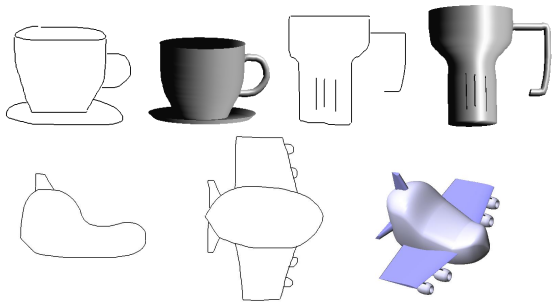

Figure 1: *Example input sketches and output 3D models. The airplane uses both the side view and the top view to construct the 3D model. Strokes which are not recognized as being part of an object can be treated as annotation scribbles that are mapped to the 3D surface, as shown in the example mug sketch on the top right.*

† email: cyang,dsharon,van@cs.ubc.ca

arbitrary hand-drawn sketches. Significantly, however, our system allows fast and flexible sketch-based modeling of particular classes of 3D objects with minimal training. Several examples of the input and output of our system are shown in Figure 1. A video of our results can be viewed at `http://www.cs.ubc.ca/~van/papers/sketch3D.mov`.

Our system assumes that it is worthwhile designing highly parameterized 3D objects for specific applications. This would allow for more transparent, user-driven content creation for games, such as a flight simulator where the user could sketch their own particular airplane design. Automotive design is another potential application that falls into this category. Our system opens the door to more general 'draw-it-as-you-see-it' modeling.

Our specific contributions are twofold: (1) a sketch-recognition algorithm that is tailored to the needs of sketch-based 3D model construction, and (2) a proof-of-concept system that demonstrates the potential of a new class of sketch-based modeling tools.

The remainder of this paper is structured as follows. Section 2 presents related work in 3D-modeling, sketch-recognition, and computer vision. An overview of our system is given in section 3. Section 4 describes the construction of the 2D templates and details the sketch recognition algorithm. Section 5 explains the 3D model construction. Section 6 presents results obtained with the system, followed by a discussion in section 7 of several common questions that arise with respect to our approach. Finally, section 8 presents conclusions and future work.

2. Related Work

There exists a large body of literature in the broad areas related to sketch-based modeling. A first research thread looks at the recognition of diagrammatic sketches such as hand-drawn circuit diagrams [AD04, GKS04], math notation [LZ04], drawings with engineering symbols [KS04], military planning drawings and other types of diagrams with symbols and connectors. A variety of algorithms are exploited within this domain, often including a mix of top-down and bottom-up procedures that are informed by some domain knowledge, or are statistically informed based on the observed stroke order [SD05] or the interpretation assigned to neighboring strokes [QSM05]. Forms of graph isomorphism have been exploited [MF02], as have dynamic bayesian networks [AD04]. However, the algorithms for this class of sketch recognition problem and their comparative evaluation [OAD04] still leave many unresolved issues in terms of building robust-yet-flexible systems. The sketch recognition component of our problem can also be thought of as being "diagrammatic". However, our 2D templates have features that are specifically tailored to our problem domain, such as the notion of part hierarchies, optional parts, and one-of-N part selection. We consider handwriting recognition as a distantly-related problem.

Interpreting line drawings of 3D polygonal objects is a problem that has attracted considerable interest in the past [LF92, BT93, EHBE97, KHD95], and continues to be the subject of ongoing work [LS02, SC04]. However, these techniques are limited to polygonal objects and they work best with objects having many 90-degree corners or sets of symmetric angles.

Sketch recognition can also be formulated as a 2D *shape matching* problem, where parts of a sketch can be identified and labelled by finding 2D templates that can match parts in the sketch while allowing for some class of deformations. A survey of shape matching can be found in [Lon98] while examples of recent shape matching work can be found in [BM02, CF02, TY04, FPZ04]. Much of this work assumes a single template that can undergo arbitrary scaling and rotation, as well as small non-affine deformations. We wish our templates to be more flexible in terms of handling large localized changes of scale, such as a small plane with large wings, and sub-parts that may or may not exist. The performance of shape contexts [BM02] in such situations is unclear. Shape contexts also require first point sampling all the strokes, which throws away useful information because a good set of segmented curves is readily obtained from a hand-drawn sketch. Thus, we choose to work with curves as a base primitive rather than the points required by shape contexts.

Many techniques for 2D shape matching use a silhouette matching approach. These do not apply to our problem because of important interior details in the sketches and our need to recognize parts for the 3D reconstruction. Work in model-based vision [Low91] is also related to our problem, although the models used are usually only parameterized in limited ways. More distantly related is the work of [RA99], which fits parameterized generative models to 3D range data.

The notion of 'pictorial structure' is used in a variety of vision algorithms to encode predictive spatial relationships between sub-parts or features of an object [PSZ98, CK01, FH05]. A similar idea is presented in the agent-based sketch interpretation system in [MA03]. We use 2D templates that encode similar predictive information regarding the location of various parts. Our template features are curve based and not image-based, and our goal is not only to identify a part but also to identify key points and curves on the part, as is necessary for the informed 3D construction of the object from the matched 2D template.

A variety of constructive, gestural 3D modeling techniques have been developed for pen-based interaction. These use pen strokes to incrementally develop the shape of a 3D object. SKETCH [ZHH96] and [IMT99] represent seminal work in this area. Our work has been inspired by these systems, and also by recent work on sketching clothing [TCH04]. The sketch-based system that we present assumes more knowledge about the objects to be drawn, but, in ex-

change, it supports drawings that can directly represent the final object shape. We see these approaches as being largely complementary in their application.

Also relevant to our work are ideas related to sketch-based retrieval of objects from a 3D database [FMK*03, FKS*04, CTSO03]. If there are a large number of sub-parts to an object, each of which can be parameterized in its own way, an approach that matches complete models requires a database to contain the cross-product of all variations, which does not scale effectively. A last interesting application of databases is the work of [TBSR04], where a database of airplane images is used to help simplify the design of 3D-curve networks, although the end product is not a 3D model and there is no general part recognition scheme.

3. System Overview

An overview of the system is given in Figure 2 and a specific example of the recognition process is given in Figure 3. The input pen strokes are first preprocessed in order to produce a graph structure. The preprocessing steps consist of (1) adding nodes at the start and end of each stroke; (2) adding nodes at corners and points of local curvature maxima; and (3) merging all nodes that are within a threshold distance of each other. The segmented strokes become the edges of the graph. The resulting sketch graph will not necessarily be fully connected because some parts such as windows and eyes can be drawn in isolation. We also choose not to segment strokes at T-junctions; these typically occur when subparts are attached to main object parts and our sketch recognition does not require graph connectivity for the recognition of such sub-parts. Figures 3(a) and (b) show the strokes of an input sketch and the resulting sketch graph.

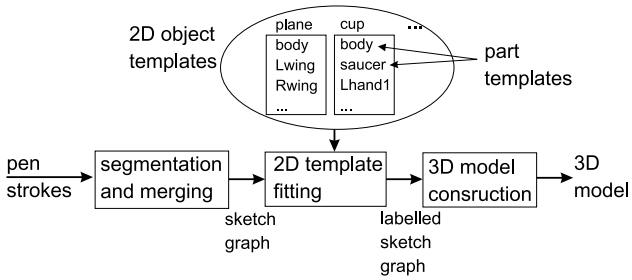

Figure 2: *System Overview.*

Given a sketch graph, the next step is that of fitting a series of 2D templates to the sketch graph. This happens at two levels: at the higher level, multiple object templates are fitted to the sketch graph and scored for their fit. At a lower level, each object template consists of multiple part templates. The object template supports a flexible instantiation of the part templates. Parts may be deemed as mandatory or optional.

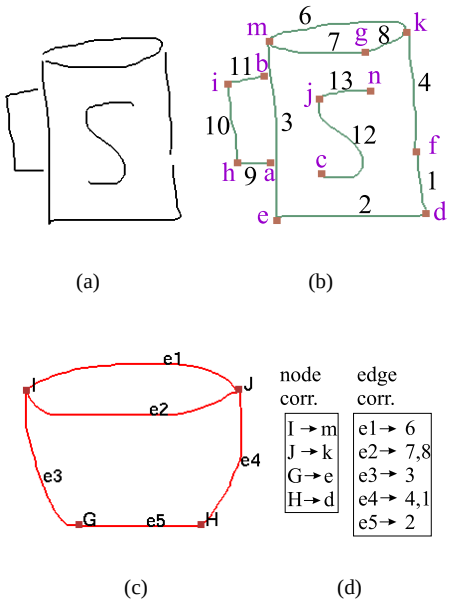

Figure 3: *The recognition process. (a) original sketch, consisting of 5 strokes. (b) graph computed from sketch strokes. (c) cup-body template graph. (d) computed best-fit correspondences.*

A choose-one-of-N option may also be specified. For example, a cup template can specify that it may have an optional right handle and that the system should choose between a rounded handle part template and a square handle template, depending on which one fits best. The ability to support this type of information distinguishes our approach from most other shape recognition approaches that support only a flat hierarchy, i.e., given N fixed templates, find the best-fit template.

Both object and part templates are represented as graphs with pre-specified node locations. Just as in a sketch graph, template graph edges are sketched curves. The actual process of matching part templates is thus accomplished using a search over node correspondences, which are then scored using a curve-matching metric on the best-fit curve correspondences that the node correspondences induce. Figure 3(c) shows a cup-body template and Figure 3(d) lists the best-fit correspondences that were computed for it. These correspondences thus define the labelling of the sketch graph.

Given a best-fit object template and all the instanced part templates that participated in the fit, the last step is that of constructing a 3D model from the labelled 2D sketch. This is done by extracting measurements from the 2D sketch, such as wing length or cup height, as well as by fitting spline curves to particular stroke segments. For example, we use a multi-segment spline curve to smoothly approximate the

sides of the airplane fuselage and the shape of the rounded cup handle. Given a priori knowledge of the object class and the extracted measurements that describe this particular desired instance, a 3D model can be constructed. Adding a new class of object to the system requires designing a 2D object template, consisting of multiple part templates, as well as developing a procedural means of constructing the 3D shape from the labelled sketch.

4. Sketch Recognition

A sketch recognition algorithm is at the core of our sketch-based modeling system. In this section, we first describe the 2D templates and their construction. We then describe how template graphs are matched to the sketch graph using a search over possible correspondences. We explain the curve feature vector which is used to evaluate the correspondences. Lastly, we give further details about the hierarchical structure of the templates.

4.1. Template Construction

Like the sketch graph, an object template is a graph, consisting of nodes which represent key points on the object, and edges, which represent curves of particular shapes that are expected to be found in a sketch. An example of an object template for a cup is shown in Figure 4. An object template is itself constructed using a sketch, where each stroke becomes an edge in the resulting template graph.† Nodes are added at the start and end of each stroke and then merged with nearby neighbors. Figure 4(a) shows the initial sketch for the template. The resulting template graph is shown in Figure 4(b). The coloring of this figure also illustrates the next step, which consists of creating multiple part templates from the global template.

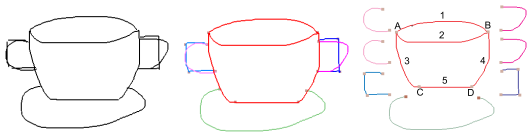

Figure 4: *Construction of the cup template. (a) sketched strokes (b) strokes grouped by part (c) the cup-body template and its 4 key-points and 5 curves, drawn with its child parts separated for clarity. The left-side handles, right-side handles, and saucer are parented by the cup-body curves 3, 4, and 5, respectively.*

Individual part templates are subgraphs of the global template graph. A simple graphical user interface supports their construction. Using a mouse, the user selects a set of edges that constitute a desired part template. The nodes and edges of this subgraph are then saved as the part template definition. Lastly, a hierarchy is established among the parts by designating an edge of an existing part template to serve as the *parent edge* of the new part template. This serves the dual purpose of introducing a hierarchical order for search and instantiating the model parts, as well as a means to encode knowledge about the relative location of parts.

Creating a 2D template for a new class of objects requires careful thought, although its actual construction can be done in about 15-20 minutes using the GUI. The template should represent a stereotypical example of the desired class of object. The template parts also need to provide the information necessary for the appropriate 3D reconstruction of the various parts.

Also associated with the parent edge is a *bounding polygon* (BP), which serves to represent the area in which to search for a part template during the sketch recognition. We use a convex quadrilateral to represent the BP. The BP is specified during the design of a part template by dragging its vertices to the desired locations in the template sketch window. The polygon lives in a coordinate system that is defined by the parent edge. One of the two nodes associated with the edge becomes the origin of this coordinate system, and the other node locates the point $(1,0)$. As shown in Figure 5, this allows the BP to be transformed to the sketch coordinate system once the parent edge has been labelled in the sketch. If the parent edge is not labelled in the sketch, i.e., the parent part was not instantiated, then there will be no attempts to fit the part templates which use that parent edge. The BP provides the sketch-recognition algorithm with necessary information about the expected relative location of parts and thus serves to greatly constrain the search space of possible correspondences, as will be discussed further in section 4.2.

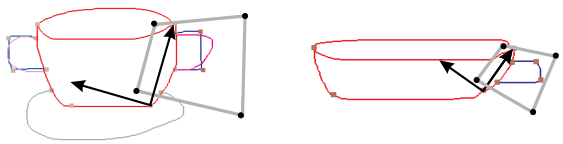

Figure 5: *Transformation of the bounding polygon for the right-cup handle. The template is shown on the left, and a sketch on the right. The right-hand edge of the cup-body template serves as the parent edge for this part.*

The part templates that are at the root of the part hierarchy, such as the body of the plane or the body of the cup, do not have a parent edge. For these parts, the part template explicitly stores an expected location for the graph nodes that comprise the part. By using a normalized coordinate system that is defined by the axis-aligned bounding box of the template sketch, normalized coordinates are computed for each

† Stroke segmentation is turned off when drawing the template graph.

node. The bounding box of a drawn sketch serves to instantiate this normalized coordinate system for the sketch and thereby provides a set of expected locations for the nodes. As shown in Figure 6, the sketch recognition algorithm will only search for correspondences for a template-graph node within a given radius of the expected location. This radius is defined in normalized coordinates (we use $r = 0.3$) and thus typically maps to an elliptical region in the sketch.

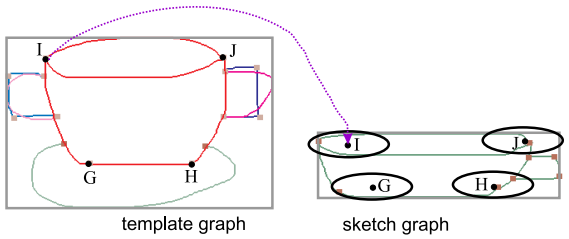

Figure 6: *Expected locations in the sketch graph are computed for the template graph nodes of the cup body using a normalized coordinate system based on the bounding box.*

4.2. Template Matching

Given a sketch graph and a series of object templates, the sketch recognition system needs to determine which object template best fits the given sketch graph. Each object template consists of multiple part templates and thus at the core of the sketch recognition algorithm is a scheme for matching a given part template to the sketch. The matching process consists of searching the sketch graph for best-fit correspondences of the template graph nodes and edges. A best-fit solution consists of (1) a corresponding sketch graph node for each template graph node; (2) a correponding *path* of one-or-more sketch edges (curves) for each template edge; and (3) a score that denotes the quality of the fit.

The matching process is summarized in Algorithm 1 and consists of an iterative process that begins by assigning a particular set of corresponding sketch-graph nodes to the template-graph nodes (lines 4–6). From this assumed node correspondence, a best-fit correspondence is computed for each template edge (lines 8–12). To understand this process, we note that a given template edge, such as the left-hand side of the cup body, may correspond to a *path* in the sketch graph that traverses multiple sketch-graph edges. This is because the sketch strokes may be over-segmented when constructing the sketch graph, or because a particular feature may have been drawn using multiple strokes. Both of these cases can be observed in the example shown in Figure 3.

There may furthermore be multiple paths between a given pair of nodes in the sketch graph, and it needs to be determined which of these paths best corresponds to the given template edge. The *pathset*(n_a, n_b) function determines a set of possible paths, $\mathbb{P}$, between nodes n_a and n_b in the sketch graph using a depth-first graph traversal. In order to limit the number of possible paths, which can become large for a highly connected sketch graph, we bound the search to paths whose ratio of arclength to straight-line distance is less than twice this same ratio as computed for the template edge that we are seeking to match.

Algorithm 1 Part Matching

```
1:  input 𝕊(N_s, E_s) { Sketch Graph}
2:  input 𝕋(N_t, E_t) { Template Graph}
3:  for i = 1 to N_iter do
4:     for all n_t ∈ N_t do
5:        C[n_t] ← select( N_s )
6:     end for
7:     score ← 0
8:     for all e_t ∈ E_t do
9:        ℙ ← pathset( C[n_1(e_t)], C[n_2(e_t)] )
10:       p ← bestMatchPath( e_t, ℙ )
11:       score ← score + 𝕄(e_t, p)
12:    end for
13:    if score > bestScore then
14:       update bestScore and best correspondences
15:    end if
16: end for
```

Each path $p \in \mathbb{P}$ is tested for similarity with the given template edge that we seek to match (line 10). This is accomplished by computing a *curve feature vector*, $\mathbb{F}$, for the template curve e_t and for each path p, and computing a match score $\mathbb{M}(\mathbb{F}(e_t), \mathbb{F}(p))$ based upon these feature vectors. The details of how $\mathbb{F}$ and $\mathbb{M}(\mathbb{F}_1, \mathbb{F}_2)$ are computed are given in the following section.

Several methods could be used to arrive at appropriate choices of node correspondences to be evaluated (lines 4–6). One choice would be to systematically evaluate all permutations of assignments of the N_t template nodes to the N_s sketch nodes. While this approach is guaranteed to find globally-optimal correspondences, it suffers from a combinatorial explosion of the number of combinations to be considered. Our system currently employs a stochastic local search method, beginning a search by using a uniform random assignment of sketch nodes to template nodes. Randomized local changes to the current set of correspondences are then evaluated and accepted or rejected in a greedy fashion. If a local maxima is reached, the current best solution is updated if necessary, and the search is restarted at another randomized point in the space of all possible correspondences. We limit the total number of iterations to $N_{iter} = 2000$.

4.3. Curve Matching

In order to determine if a given path through the sketch graph is similar in shape to a desired edge of the template

that we seek to locate in the sketch, we first define a curve feature vector $\mathbb{F}$ over paths or edges. This is defined as $\mathbb{F} = [f_1 \quad f_2 \quad f_3]^T$, where $f_1 = \theta$, $f_2 = a/d$, $f_3 = A/d^2$, θ is the angle of the straight line between the curve endpoints with respect to the horizontal, a is the arclength of the curve, d is the straight-line distance between the endpoints, and A is the signed area between the curve and the straight-line. The key parameters are illustrated in Figure 7. Feature f_1 encodes preferences for desired angles, as our sketch recognition scheme is not rotation-invariant. Features f_2 and f_3 encode information about the type of curve that connects the endpoints and provide useful distinctions between a curve that passes to above or below the straight line (f_1), as well as how much the curve meanders (f_2).

We note that a limitation of the curve feature vector is that two curves that are symmetric with respect to the perpendicular bisector of the endpoints will have the same curve feature vector. Also, the curve-feature vector is potentially problematic for closed curves. For this reason, closed curves are automatically segmented into open curves. An axis-aligned box is computed for the stroke. If $w > h$, the curve is segmented at the points where x_{min} and x_{max} occur. Otherwise, the curve is segmented where y_{min} and y_{max} occur.

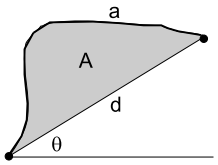

Figure 7: *Key parameters used to compute the curve feature vector: arclength a, straight-line distance d, angle θ, and area A.*

The matching function that compares two curve feature vectors is defined as $\mathbb{M}(F_a, F_b) = k\sum_i w_i g(\sigma_i, f_{i_a} - f_{i_b})$, where $g(\sigma, x) = e^{-0.5(x/\sigma)^2}$ and $k = 1/\sum_i w_i$. The w_i values provide a relative weighting of the feature vector components.[‡] The matching function provides a maximum score of 1 for curves that are highly similar, and a score of zero for those that are very dissimilar. σ_i provides a means of scaling the feature vector elements (or rather their differences) to provide a meaningful level of sensitivity.[§] As described in Algorithm 1, the match scores of all curves that make up a part template are summed in order to yield a total match score for the part template.

4.4. Template Hierarchy

Object templates are specified in a simple script file and consist of an ordered list of part templates T_j to be matched to the sketch graph, $\mathbb{S}$, as well as an instancing threshold, α_j. The thresholds provide a means to allow parts that have a low best-match score to not be instanced as part of the model. This provides control to the template designer as to whether or not a scribble drawn to the right of the cup should be interpreted as a cup-handle anyhow (low threshold), or as a scribble that is to be ignored (high threshold).

Part templates also provide support for one-of-N matching. For example, it may be possible to interpret a cup-handle as either a rounded handle or a square handle, each of which has its own part template and its own associated 3D-model-construction method. A comma-seperated list of part-template names in the object template has a one-of-N semantics, meaning that a fit should be attempted of all the templates in the list, but only the best-fitting template should be instanced.

The scores of part templates can be combined to compute an overall object template score. Currently, we compute a score based on the mean scores of the instantiated part templates. The object template scores allow a sketch to be fit with all the possible object templates, with only the best-fit object template being used to instantiate the 3D model.

5. 3D Model Construction

Once the sketch recognition is complete, a 3D model is constructed in a procedural fashion. This procedural construction is hard-coded for each object class. We have found that this is generally the most time-consuming aspect of adding a new object class. We now discuss the construction rules used for each object class.

The cup construction proceeds as follows. A centerline is estimated from the axis-aligned bounding box for the cup-body strokes. A spline is fitted to the sketched curves segments that make up the right side. A surface of revolution is then constructed from the centerline and the fitted spline. The saucer is similarly constructed as a surface of revolution from a spline that is fitted to the appropriate sketched curves. The 3D handle is constructed by sweeping a predefined 3D cross-section along a spline that is fitted to the sketched handle. Adjustments are made to the positions of the saucer and the handle(s) so that these parts are contacting the body even if there are gaps that exists in the sketch. Unrecognized strokes that are drawn over the region of the cup body are projected onto the cup body as annotations, as shown in Figure 1.

Our airplane model assumes the existence of two recognized sketches of traditional orthographic views: a side view and a top view. These each have their own separate template. The information from both labelled sketches is then combined in the model building process. The system can also accept a top-view alone, in which case default assumptions are made about the fuselage and the tail.

The airplane fuselage has a default cross-sectional shape

[‡] We use $w_1 = 2, w_2 = 1, w_3 = 1$.

[§] We use $\sigma_1 = 22^\circ, \sigma_2 = 0.25, \sigma_3 = 0.1$.

plane	*n*	*e*	cup	*n*	*e*	fish	*n*	*e*
body	2	2	body	4	5	body	3	3
Lwing	4	3	saucer	2	1	topfin	2	1
Rwing	4	3	RHround1	2	1	midfin	2	1
Ltail	4	3	RHround2	2	1	eye	2	2
Rtail	4	3	LHround1	2	1	botfin1	2	1
Lengine	2	1	LHround2	2	1	botfin2	2	1
Rengine	2	1	RHsquare	4	3			
Lengine2	2	1	LHsquare	4	3			
Rengine2	2	1						

Table 1: *List of template parts used to build the cup, plane, and fish templates. n and e denote the number of nodes and edges in the part template graphs, respectively.*

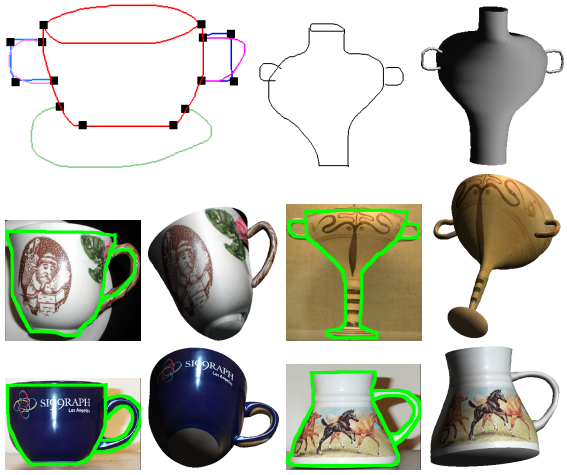

Figure 8: *The cup template graph, and sketches of cups and wine glasses shown together with the synthesized 3D models.*

that is uniformly scaled as it is swept along the spline curves defined by the fuselage side, top, and bottom curves. The wings and horizontal stabilizers are constructed using a swept-airfoil cross-section. The tail fin has a similar swept construction. Engines are instanced as copies of a single 3D engine model, scaled and translated to match the top-view sketch. Left-right symmetry is enforced in the final model. The model building process must also resolve any conflicts between the top view and the side view, such as the length of the fuselage. This is handled by scaling one of the views so that the fuselage lengths matches the other view.

The 3D fish model consists of a body, dorsal fin, anal fin, a pair of pectoral fins, and a pair of pelvic fins. Top and bottom splines are fitted to the top and bottom of the body, starting at the front tip of the fish and ending at the top and bottom of the caudaul fin, respectively. A default elliptical cross-sectional shape is then swept from front-to-back along these splines, undergoing uniform scaling as necessary. This swept surface is tapered to end with zero-width at the beginning of the caudal fin, which is identified as the narrowest part of the right-half of the fish. All the fins, including the caudal fin, are modeled as polygons. The pectoral and pelvic fins are tilted at predefined angles away from the body.

6. Results

The current recognition process requires on the order of one second to match an object template hierarchy against a sketch, as measured on an 800 Mhz TabletPC. The current sketching interface consists of a sketch area, an area for displaying the resulting 3D model, and a button panel. The user draws their desired sketch in the sketch area and hits a 'Recognize' button. The interface supports a progressive workflow if this is desired. The recognition and model building can thus be invoked at any time for whatever parts that have currently been drawn. One caveat is that parts such as the handles of the cup cannot be recognized in isolation. The parent part must first be successfully instantiated.

We have tested the system using 3 classes of objects: cups, planes, and fish. Table 1 gives the list of part templates used for each object template. Symmetry in the drawn sketch is not a requirement of the system, as can be seen in the side view of the airplane.

The most effective mode of use for our system is one in which the user can tell the system which class of object is being drawn, and thus the system knows in advance which object template hierarchy to match against the drawn sketch. Another mode that we support is to have the system do a linear search through each of the available object templates in turn and then instantiate the 3D object corresponding to the best-fit object template.

Images can be loaded into the background to use as a reference for tracing a particular airplane, cup, or fish. At the same time, the model building can extract a texture map from the background image if this is desired. This thus supports quick-and-dirty construction of models from photographs.

The cup template represents the prototypical view that is commonly used to draw or photograph a cup. A variety of cups and mugs modeled using our system are shown in Figures 1 and 8. The template supports left and right rounded handles and square handles. A second left and right rounded handle can also be recognized, which can serve to either define inside and outside edges for the handle, or as a second rounded handle. In our implementation, it is the model building process that distinguishes between these two cases. For single-curve handles, a default handle width and cross-section is assumed. For double-curve handles, a rounded-rectangular cross-section is assumed that has fixed depth but varies in width according to the drawn curves.

Figures 1 and 9 show a number of sketches and the re-

sulting 3D airplane models. The current implementation assumes that the side-view sketch is drawn and recognized first. The 3D model is then instanced when the top-view sketch is drawn and recognized.

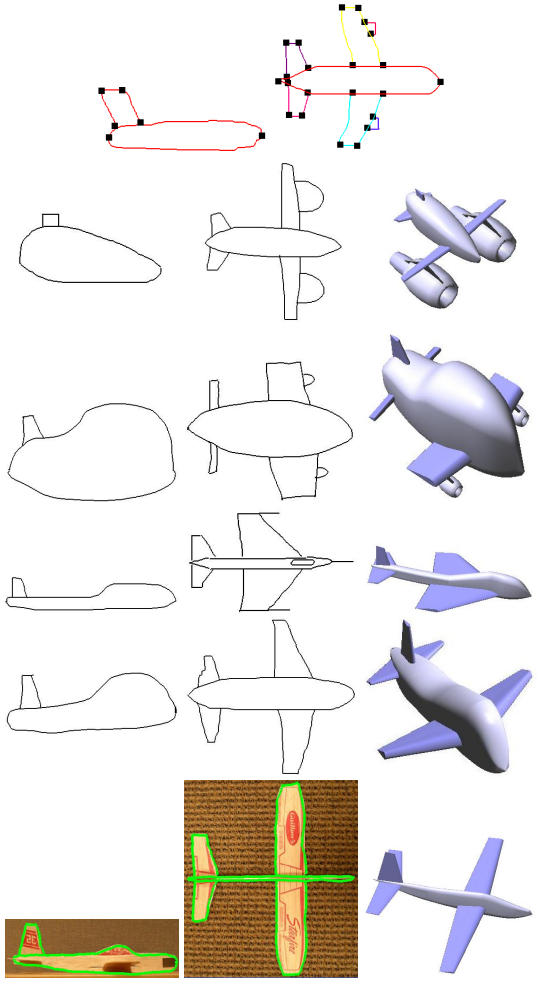

Figure 9: *The airplane side-view and top-view templates, and five examples of input-sketches shown together with the resulting 3D models.*

Figure 10 shows two examples of fish that have been constructed by tracing over photographs of fish. The texture is then lifted from the images and applied to the 3D models.

7. Discussion

An obvious question that arises from our template-driven system is: "Is it not simpler to have users directly edit the template curves to get the object they want?". Our answer to this is twofold. First, drawing is often easier than template manipulation. Consider the case of the cup template, which has 4 handles and a saucer, as shown in Figure 4. A great

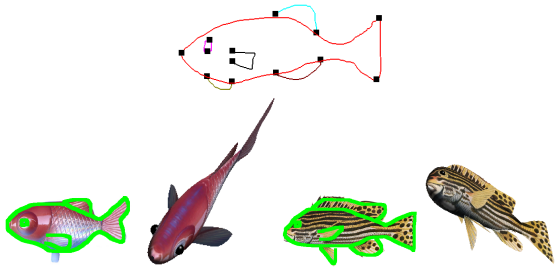

Figure 10: *The fish template and two fish models that were created by tracing over photographs.*

many control points would need to be repositioned to create the tall mug shown in Figure 1, whereas it can be drawn in a matter of seconds. The user also has to understand what control points are, why there are two left handles and two right handles, has to delete 3 of these, and then move the control points on the remaining handle in order to achieve the squarish right handle that is finally desired. A similar case can be made for drawing airplanes instead of editing an airplane template. All the optional template parts introduce considerable visual clutter and need to be explained to the user. A sketched curve that represents a particular feature, such as the side of a cup can be fit with a dynamically-chosen number of spline segments, whereas a template would typically be constructed with a fixed number of spline segments. Second, and perhaps most importantly, a drawing interface allows for the kind of transparency in use that we strive for in this work. The user should be required to know as little as possible about the underlying representation and assumptions that will be used to construct the 3D model.

A second question that one can ask of our system is: "Will it scale to large numbers of object classes?". Our solution is scalable with the addition of more templates. To be efficient with a large number of templates would require a separate object classification step in order to identify a small set of candidate object classes to which the full template match could then be applied. While our system still has robustness issues, it can support large within-class variations. For example, our cup template can model a large variety of cup or mug types and shapes (Figures 1 and 8). Similarly, our airplane template supports a significant variety of shapes (Figures 1 and 9). With respect to the use of templates, is is clear that some form of prior knowledge is essential for sketch or line-drawing recognition, and we choose to embody this prior knowledge in our 2D templates.

How familiar do users need to be with the templates in order to use the system? This project began with us collecting sketch data on a TabletPC, giving purposely-vague instructions by asking users (with no knowledge of templates) to “draw a cup,” or “draw a plane,” in a wizard-of-oz experiment. Many of the cup sketches can be recognized by our

system. The airplanes can generally not be recognized without some errors. Thus, some familiarity with the template is generally required, as mentioned in the introduction.

8. Conclusions and Future Work

We have presented a system that supports quick-and-dirty creation of 3D models based upon a sketching interface. Many recent sketch-based systems apply a gestural or context-sensitive interpretation to drawn strokes. Instead, we propose a template-based recognition algorithm which treats strokes as a sparsely-populated image and then drives the instantiation of a flexibly-paramaterized 3D model.

Our sketching system has a number of known limitations. The system is not nearly as robust as we would like. The most common reason for sketches not being recognized is because of some sloppiness in sketches that results in disconnected strokes. Increasing the threshold distance within which sketch-graph nodes are merged allows for more of such gaps to be bridges, but this comes at the expense of losing further detail in the drawn parts because strokes which were intended to be distinct may be merged together. The key-points in the curves that become the nodes of the sketch graph are not always extracted as desired. The curve feature vector currently has no notion of the smoothness of the curve, and thus smooth template curves may be matched to jagged paths through the sketch graph without penalty, assuming that the remaining curve features match well.

If there are many templates loaded, the recognition can become slow and it also becomes more prone to recognition errors, such as potentially interpreting a flat cup as a plane body. In the future, statistically-based object-class recognition algorithms borrowed from the computer vision literature could be used to help identify likely object classes and to therefore avoid a linear search through all object templates.

There are numerous interesting directions for future work. Our sketch recognition method allows for recognition with only one example per object class, namely the object template, but a statistically-based approach based on a set of hand-labelled sketches would potentially provide a more informed approach. This could be applied to modeling the space of expected part shapes as well as the sketching tolerances that should be accomodated. We foresee the possibility of using current-generation image segmentation algorithms to further automate the knowledge-based construction of 3D models from images.

User testing under controlled conditions would provide useful data with respect to robustness and determining the extent to which familiarity with the template structure is helpful. Given a 3d object, how do people sketch it in the absence of any instructions ? What is the variability in such drawings?

References

[AD04] ALVARADO C., DAVIS R.: Sketchread: a multi-domain sketch recognition engine. In *UIST '04 ACM symposium on User interface software and technology* (2004), pp. 23–32.

[BM02] BELONGIE S., MALIK J.: Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 24 (April 2002), 509–522.

[BT93] BARROW H. G., TENENBAUM J. M.: Retrospective on interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence 59* (1993), 71–80.

[CF02] COUGHLAN J. M., FERREIRA S. J.: Finding deformable shapes using loopy belief propagation. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III* (2002), Springer-Verlag, pp. 453–468.

[CK01] CYR C. M., KIMIA B. K.: 3d object recognition using shape similarity-based aspect graph. In *Proceedings of ICCV* (2001).

[CTSO03] CHEN D.-Y., TIAN X.-P., SHEN Y.-T., OUHYOUNG M.: On visual similarity based 3d model retrieval. *Computer Graphics Forum (Eurographics 2003) 22*, 3 (2003).

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

[FH05] FELZENSZWALB P. F., HUTTENLOCHER D. P.: Pictorial structures for object recognition. *Intl. Journal of Computer Vision 61*, 1 (January 2005), 55–79.

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Trans. Graph. 23*, 3 (2004), 652–663.

[FMK*03] FUNKHOUSER T., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D.: A search engine for 3d models. *ACM Trans. Graph. 22*, 1 (2003), 83–105.

[FPZ04] FERGUS R., PERONA P., ZISSERMAN A.: A visual category filter for Google images. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic* (May 2004).

[GKS04] GENNARI L., KARA L. B., STAHOVICH T. F.: Combining geometry and domain knowledge to interpret hand-drawn diagrams. In *AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Intelligent and Natural* (2004).

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '99* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 409–416.

[KHD95] KANUNGO T., HARALICK R. M., DORI D.: Understanding engineering drawings: A survey. In *Proceedings of First IARP Workshop on Graphics Recognition* (1995), pp. 217–228.

[KS04] KARA L. B., STAHOVICH T. F.: An image-based trainable symbol recognizer for sketch-based interfaces. In *AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Intelligent and Natural* (2004).

[LF92] LECLERC Y. G., FISCHLER M. A.: An optimization-based approach to the interpretation of single line drawings as 3d wire frames. *Int. J. Comput. Vision 9*, 2 (1992), 113–136.

[Lon98] LONCARIC S.: A survey of shape analysis techniques. *Pattern Recognition 31*, 8 (1998), 983–1001.

[Low91] LOWE D. G.: Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 5 (May 1991), 441–450.

[LS02] LIPSON H., SHPITALNI M.: Correlation-based reconstruction of a 3d object from a single freehand sketch. In *AAAI Spring Symposium on Sketch Understanding* (2002), pp. 99–104.

[LZ04] LAVIOLA J. J., ZELEZNIK R. C.: Mathpad2: a system for the creation and exploration of mathematical sketches. *ACM Trans. Graph. 23*, 3 (2004), 432–440.

[MA03] MACKENZIE G., ALECHINA N.: Classifying sketches of animals using an agent-based system. In *Proceedings of the 10th International Conference CAIP, Springer Lecture Notes in Computer Science 2756* (2003), pp. 521 – 529.

[MF02] MAHONEY J. V., FROMHERZ M. P. J.: Three main concerns in sketch recognition and an approach to addressing them. In *AAAI Spring Symposium on Sketch Understanding* (March 2002).

[OAD04] OLTMANS M., ALVARADO C., DAVIS R.: Etcha sketches: Lessons learned from collecting sketch data. In *Making Pen-Based Interaction Intelligent and Natural* (2004), AAAI Fall Symposium.

[PSZ98] PELILLO M., SIDDIQI K., ZUCKER S. W.: Matching hierarchical structures using association graphs. *ECCV'98 1407* (1998).

[QSM05] QI Y., SZUMMER M., MINKA T. P.: Diagram structure recognition by bayesian conditional random fields. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)* (2005).

[RA99] RAMAMOORTHI R., ARVO J.: Creating generative models from range images. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), pp. 195–204.

[SC04] SHESH A., CHEN B.: Smartpaper: An interactive and user friendly sketching system. *Computer Graphics Forum (Eurographics Proceedings) 23*, 3 (2004).

[SD05] SEZGIN T. M., DAVIS R.: Hmm-based efficient sketch recognition. In *Proceedings of the International Conferences on Intelligent User Interfaces (IUI'05)* (New York, New York, January 9-12 2005), ACM Press, p. (To appear).

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

[TCH04] TURQUIN E., CANI M.-P., HUGHES J.: Sketching garments for virtual characters. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2004), Hughes J. F., Jorge J. A., (Eds.).

[TY04] TU Z., YUILLE A. L.: Shape matching and recognition using generative models and informative features. In *Proceedings of ECCV'04* (2004).

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: Sketch: An interface for sketching 3d scenes. In *Computer Graphics (SIGGRAPH '96 Proceedings)* (1996), pp. 163–170.