EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2006)
Thomas Stahovich and Mario Costa Sousa (Editors)

Producing Models From Drawings of Curved Surfaces

Matthew Kaplan[1] and Elaine Cohen[2]

[1]ARTIS, Inria Rhone-Alpes
[2]University of Utah

Abstract

We present a method for creating 2 $\frac{1}{2}$D models from line drawings of opaque solid objects. We allow the artist to draw naturally, differing from many previous approaches. Our system allows both perspective and orthographic projection to be used and makes no a priori *assumptions about the type of model to be produced (i.e. planar, curved, normalon) . The frontal geometry is reconstructed by placing constraints at the contours and solving a 2D variational system for the smoothest piecewise smooth surface. An analysis of line labelling allows us to determine what constraints are possible and/or required for each input line. However, because line labelling produces a combinatorial explosion of valid output geometries, we allow the user to guide the constraint selection and optimization with a simple user interface that abstracts the technical details away from the user. The system produces candidate reconstructions using different constraint values, from which the user selects the one that most closely approximates the model represented by the drawing. These choices allow the system to determine the constraints and reconstruct the model. The system runs at interactive speeds.*

Categories and Subject Descriptors (according to ACM CCS): ([I]: .3.3)Computer GraphicsShape Modeling

1 Introduction

Few sketch based modeling systems allow the artist to draw naturally. Typically, designers are forced to learn a set of drawing operators that are used as an interface to an underlying CAD system. Alternatively, previous methods that analyze existing drawings typically limit the type of drawings/models that can be reconstructed to a subset of models useful in CAD. The goal of this research is to allow the artist to draw as naturally as possible, placing minimal restrictions on the structure and process of the input drawing and the form of the output model. The reconstructed model should be a close approximation of the artists intent. We briefly review how our system differs from previous work:

We allow the user to draw interactively, without having to learn any special rules, although we do limit the contours in our system to being representative of surface geometry of an opaque solid object. Since contours can be either straight or curved lines, we make no assumption about the type of model to be produced. Most previous research limited the type of models that could be represented to either polyhedral or normalon (all object faces parallel to one of the three coordinate axes), or CSG-tree style construction of curved models. Freedom in the ordering of the input strokes also yields an implicit construction method. This means that we place no limitation on the process by which the drawing is created whereas the CSG-tree process requires an explicit construction sequence. We relax the simplifying orthographic projection assumption that most previous research imposes because it does not correspond with how artists actually draw. We reconstruct from a single view for the same reason.

To use our system, the designer draws into the screen buffer. The system automatically locates constraints along curved and straight contours through analysis of line labeling techniques. Because line labeling yields a combinatorial explosion of valid constraints, we employ the user's perception to find a correct constraint set but abstract the constraint selection mechanism from the user, in order to minimize extra knowledge required to use the system. The system iteratively produces candidate reconstructions with different constraint possibilities. From these choices, the user selects the reconstruction that best approximates the desired output model. Successive user choices help to define a gradient through the system's constraint search space. The output is a piecewise smooth surface, created by applying constraints to a $2\frac{1}{2}$D mesh embedded in the drawing plane.

2 Related Work

How to infer models from sketches has been extensively studied so we present only the most closely related work. To fully appreciate this technique, it helps to be familiar with the background material, especially line labeling, in [Mal87,LZS01,Var05].

2.1 Reconstruction Methods

Reconstruction methods create a model from an existing drawing. Most research in this area falls in the category of line labeling. See [CPM04] for a comprehensive overview.

The first successful attempts to catalogue types of line labels [Huf71,Clo71] were used to identify drawings that represented unrealizable scenes. The concept of gradient space was presented in [Mac73] to allow the labeling polyhedral scene drawings. The first method for producing labeling for curved line drawings was presented in [Tur74], while the first full theory of line labelings for piecewise smooth curved surfaces was developed in [Mal87]. Cases where three faces meet at a vertex were considered in [RH78], leading to a smaller junction catalogue. Another method for reconstructing drawings of curved objects was demonstrated in [VYJH04]. It required the user to create a line drawing of a polyhedral template corresponding to the curved drawing. Then the polyhedral template drawing would be inflated and used to guide reconstruction of the curved model. More recently, it was shown in [LB90] and [VM00,VSM04,VM02] that restricted classes of normalons and regular objects composed of planar faces can be interpreted. They argued strongly that the objects reconstructed under the assumption of regular angles are useful to engineers. A correlation based method, successful for polygonal objects, was presented in [LS02,LS96,Lip98], and extended with additional operators in [SC04].

2.2 Gestural Methods

Gestural methods use strokes to define input parameters to CAD operations. A CSG tree-like series of operations defines the model. Often, the user must select which operation each stroke should perform. This requires the user to learn the system conventions. Examples include Sketch [RCZ96] and Chateau [TI01].

Another class of algorithms assumes strokes are silhouettes and inflates the interiors of silhouette bounded regions. In [IMT99,dAJ03] the medial axis determines a polygonal height field for relative heights of shape interiors. In [KHR02] implicit surfaces are fitted to silhouette strokes, while in [TZF04,AGB04] implicit modes are formed by convolving implicit surfaces along stroke paths. They added complexity to the surfaces by composition and subtraction of implicit shapes. These systems require an explicit design order to model construction. Most steps extended, destroyed or altered previous detail. This removes one of the benefits of using drawings as input which is that there is no specific construction sequence.

2.3 Other Related Areas

The field of shape from shading analyzes image color or intensity gradients to determine the geometric properties of a scene. In [JJAR97] it is suggested that most of the interpretive process results from the use of previous experience with an object in order to classify it. This may be impractical for dealing with arbitrary input since it would be necessary to classify every possible object that a user *might* draw, matching under such general conditions would still be a hard problem, and this would not allow the user to draw new or imaginary items. The reader is referred to [Wil90,Wil91,Kan98,HAA97,OCDD01] for related, but not directly applicable, research in 3D shape recovery.

The research presented in [LZS01] on reconstructing surfaces by optimizing constraints defined from a single view is closely related and is examined more closely in Section 6.

3 Definitions

We use a simplified model of line drawings based only on the projection of depth and orientation (normal) discontinuities of an individual object in 3D space with no surroundings. As in [Mal87], an *object* is defined as a connected, bounded and regular subset of R^3 whose boundary is a piecewise smooth surface, where *regular* means that it is the closure of the interior. Each point within the object domain is the projection of a visible point on the object onto the 2D image plane. At each planar position (x,y), a height $f(x,y)$ and a normal $n(x,y)$, are defined. These functions are continuous at all points within the image except at lines, which represent discontinuities. The line drawing, then, is defined as the locus of these discontinuities. The locations at which two or more lines meet is called a *junction*. A surface incident to an line, is considered *attached* to that line if its depth is at least C^0 continuous with the line, or *detached*, if it is not.

4 Line Constraints

Line labeling is typically means classifying each image curve as corresponding to either a depth or orientation discontinuity in the scene and further subclassifying each type of discontinuity. Furthermore, a junction catalogue is defined that represents the possible configurations of labelings of each incoming stroke at a junction. A labeling of the drawing that corresponds to a projection of a realizeable scene is known as a *legal* labeling.

In a simplified labeling scheme, a line may be indicative of either a normal or depth discontinuity. Normal discontinuities are denoted by '+' for a convex edge,i.e., adjacent surfaces enclosing a filled volume corresponding to a dihedral angle less than π, or by '-' for a concave edge , i.e., adjacent surfaces enclose a filled volume corresponding to a dihedral angle greater than π. Depth discontinuities (silhouettes) are denoted '$\longleftarrow$' for an occluding convex edge or '$\longleftarrow\longleftarrow$' for a silhouette, though these two labels are often combined. The labelings for several different objects are shown in Figure 1. Most line labeling solutions involve backtracking (an example of the NP-complete constraint satisfaction problem) and

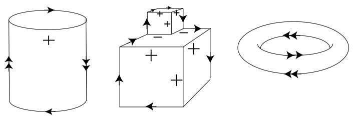

Figure 1: *Several models and their line labels.*

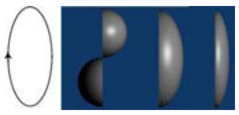

Figure 2: *A labeling for a drawing of an oval is shown at left. A reconstruction based on this labeling is not unique and can produce the three models whose profiles are shown at right.*

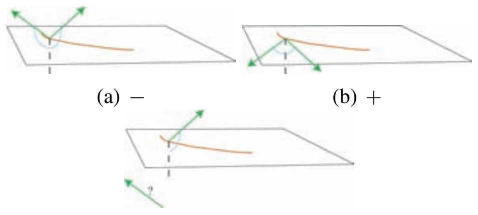

Figure 3: *The constraints formed by the line labels at an isolated point on the curve take the form of a) Concave b) Convex and c) Silhouette. The tangent to the incident surface is shown in green. The angle subtended by the local surfaces is shown in blue.*

produce numerous legal labelings. Once a legal labeling has been obtained, an optimization is performed to determine reasonable depth values, yielding a model. There is no way to know which legal labeling is the correct one, and even a correct labeling may yield a large number of output surfaces. An example of this is shown in Figure 2.

Line labeling produces constraints that are ambiguous. A contour is known to have normal and/or depth constraints, but the values that precisely define those constraints are unknown. We introduce a new method of classifying constraints for surfaces incident to a contour that allows identification of both the constraints and the parameters needed to fully specify a piecewise smooth output surface. Figure 3, which shows the type of constraints that may occur at each line.

- '+' A convex edge. The normal discontinuity that occurs along this edge represents the intersection of two smooth surfaces whose incident faces have surface normals with the angle in the span $[0,\pi)$.
- '-' A concave edge. The normal discontinuity that occurs along this edge represents the intersection of two smooth surfaces whose incident faces have surface normals with the angle in the span $[\pi,2\pi)$.
- '$\rightarrow$' A silhouette contour. The normal at the line satisfies the formula $N \cdot V = 0$, i.e., it is perpendicular to the view vector, or for convex occluders, $N \cdot V <= 0$. The attached surface has no constraint on its normal. The depth across the contour is discontinuous, and the attached surface must lie above the occluded surface.

Though three labels are shown, contours have only depth (*silhouette*) and normal (*crease*)discontinuities. More importantly, certain constraints on incident surfaces must exist for each type of discontinuity. Constraints on the surfaces incident to each contour either specify the normal of the incident surface at each point on the contour, or specify the difference in depth between the attached and detached surfaces across a depth discontinuity. A normal discontinuity has two attached incident surfaces, so a surface normal constraint exists for each incident surface and is defined by a direction vector. A depth discontinuity has a surface normal constraint for its attached surface and a depth discontinuity for its detached surface and is defined by a scalar distance between the attached and detached surfaces. Although the detached surface must lie below the attached surface, no assumption is made as to whether the attached surface is raised or the detached surface depressed.

In conclusion, a frontal reconstruction of a piecewise smooth model can be obtained if, given that each line is parameterized by $0 \leq t \leq 1$, at each point on every line in the scene it is necessary to have 1) a position $z = f(t)$ that is C^0 continuous on each line but not at junctions, 2) a surface normal, $N_0(t)$ and/or $N_1(t)$, for each incident attached surface, and 3) a depth relation for each detached surface incident from a line, $D_0(t)$ or $D_1(t)$. This is a more stringent set of conditions than that required in [Mal87] and may be unobtainable in an arbitrary sense since it requires a full dense labeling (see Section 5).

5 Reducing the Dense Labeling Problem

In drawings of curved objects a label may transition along the contour, as shown in Figure 4, at *critical points*. A single label applied to each line yields a *sparse* labeling, whereas applying a label to every point on every line yields a *dense* labeling. Dense labeling is neccesary to fully consider all curved objects that *may* be produced by a drawing. Fortunately, instances where scenes project to drawings that can be represented *only* by dense labelings are rare, so one solution is to split lines at critical points and then use a sparse labeling to find the solution.

If critical points are overlooked by the system, the reconstruction may be incorrect. If too many critical points are identified, then the reconstruction will be more difficult to produce. There has been extensive research on identifying how viewers locate important landmarks in line drawings. In [Mal87] splitting contours at zeros of curvature is suggested while in [HR85] it is postulated that part boundaries occur at extremas of negative curvature. Our system splits contours at sharp bends, zeroes of curvature, and local maxima and minima of curvature. Then it combines split locations that are close to avoid the creation of degenerate lines that occupy few pixels.

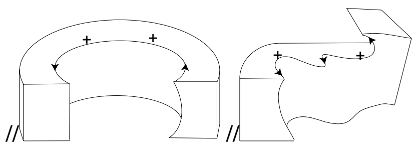

Figure 4: *Line labels may change at critical points (also known as* phantom junctions*) along contours.*

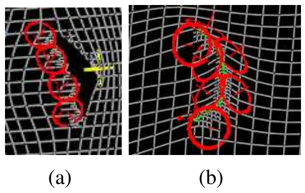

Figure 5: *a) Normal and depth constraints applied around a silhouette. b) Normal constraints applied around a crease.*

6 Solver

We require the system to allow constraints to be placed as presented in Section 4 in a mesh defined over the drawing plane, to be capable of producing both planar and curved output meshes, and to run at interactive rates. We use the method in [LZS01] and cast the model reconstruction problem as a constrained variational optimization problem. In [LZS01], they fit a piecewise continuous surface represented as an adaptive grid over the image plane and solve a large scale optimization with user defined constraints. This produces a smooth surface.

Our system places point constraints (normal and depth constraints) for incident surfaces and curve constraints (depth and normal discontinuity constraints) along contours. Constraints are automatically placed for all contours whose constraint types have been determined. Point constraints are placed every 5 pixels along a contour, several pixels away from the contour in the normal direction. If a contour has a detached surface, then a depth discontinuity constraint is placed. If a contour has two attached surfaces, then a normal discontinuity is placed. This specifies the location of the constraints in the image plane; Section 7 presents how to determine values for the normal vectors and depths.

While a human could place constraints manually and use [LZS01] to achieve a similar result, it would require a high degree of knowledge of their system to place proper constraints and would take far longer to do by hand. Indeed, even a simple scene with manual constraints placed mainly along very simple contours required 156 constraints in [LZS01], while for more complex scenes, they report requiring 264 or more constraints. Another other similar system [Koe98] required a constraint for every pixel. The reduction in human effort offered by our system is advantageous.

7 Finding Constraints with User Guidance

Since domain knowledge affects labeling and parameters, it may be impossible to choose them automatically. Therefore, we coopt the user's domain knowledge to determine correct constraints. By abstracting technical details from the artist and allowing him to apply just perception, we hope to minimize the amount of specific system knowledge needed.

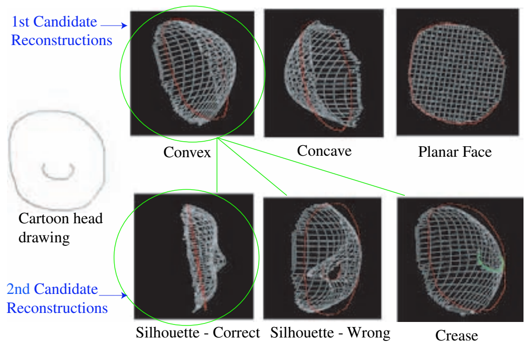

Figure 6: *A simple example: The PCG (Section 7.2) first produces constraints for the exterior silhouette. The convex case is most like the head, so the user chooses* Good *for it (its constraints descend to all future reconstructions), denoted by the green circle. Next, the PCG tests constraints for the interior contour as a crease and as a silhouette from either side. The crease case makes a small dent on the surface. One silhouette case actually places the nose behind the face, whereas the silhouette on the left places the nose correctly. The model is correspondingly as crude as the drawing.*

Our system iteratively presents the user with multiple candidate reconstructions, each constructed using different constraint values. Users choose, via a simple interaction mechanism, the reconstruction that best fits their concept of the model, gradually allowing the system to deduce correct constraints. The user never needs to know the details of the constraint selection mechanism. An overview of our system is shown in Figure 10. A simple demonstration of the process is shown in Figure 6.

Known capacities of viewer perception support the validity of this method. A series of papers on the topic of human perception of shape in 2D images and line drawings [Koe98,Koe84,KvDCL96,PTKK] that argue strongly that smooth surfaces generate a set of perceptually salient landmarks that are viewpoint invariant. They report that users are typically able to establish correspondence between surface normal and the projection of surface features with high accuracy,on average within a few degrees, over different orientations. Viewers are also able to establish surface depth but with slightly less accuracy. Techniques similar to ours have been previously demonstrated in other contexts [MAP*97].

7.1 User Interface

Each tentative reconstruction is viewed in its own window. We provide an interface where a user is able to indicate, at a very high level, the quality of each candidate surface reconstruction by applying their comparative perception to the surface meshes. The meaning of the five buttons provided as a selection mechanism is as follows:

- **Good** : This is good.
- **Bad** : Some portion is incorrect.

- **Refine** : This indicates that the the user desires more reconstructions like the current one.
- **Refine+** : A special case of the refine instance, this tells the system that the corrections needed are minor.
- **Refine-** : A special case of the refine instance, this tells the system that the corrections needed are large.

A history class records the type of constraint and parameters used for each line, and the user selected value in each candidate reconstruction. Selecting *Good* or *Refine* for any candidate reconstruction automatically closes all other currently shown reconstructions with a value of *Bad* in their history.

7.2 Finding the Right Constraint

We define a *Probable Constraint Generator* (PCG) that iteratively attempts to generate new, better sets of constraints based on a drawing and a history of attempted reconstructions. During any iteration, the PCG attempts to produce constraints for (in order of decreasing importance) : 1) a set of lines, 2) a set of faces, 3) an individual face or 4) an individual line. The PCG only attempts to generate constraints for one specific set of contours at a time, that is, though the PCG may produce several candidate reconstructions simultaneously, they will all be operating on the same contour(s). This restricts the search domain to simplify and speed the search.

The PCG first attempts to determine the appropriate type of each constraint and second, the value of the parameter for that constraint. In our system, a normal is defined by an angle in the range $[0, 2\pi]$. According to [Mal87] the surface normal at a contour should be perpendicular to that contour, so the angle simply specifies where the normal is within the unit circle lying in the normal plane at any given location on the contour. For a normal constraint representing a face, a hemisphere of directions must be considered. Depth constraints have an unbounded scalar domain, though in practice, this is bounded to near and far clipping planes.

In successive iterations, the PCG generates new candidate solutions either when no search has been initiated, or when a search is currently underway. In the first case, if there are contours with undefined constraints, a set of undefined contours are chosen and constraints are generated as discussed in Section 7.3. The first user selection of *Good* or *Refine* determines the correct constraint type for the contours being tested. Selecting *Bad* excludes the constraint type from the current search. The initial parameter value from case 1 then defines a start condition for a search of the parameter domain that is performed in case 2.

A simple search of the parameter domain, akin to a binary search, can be performed. The history set defines a gradient through the parameter domain, allowing the search to gradually approach a correct parameter value. For every parameter value being tested, a candidate reconstruction is created. If the user selects *Good* for a candidate, the constraint type and parameter are validated and the search is terminated. If *Refine* is selected, the PCG creates two new candidate reconstructions that bisect the remaining domain space surrounding the current parameter value. If *Refine+* is selected, the new parameter values move 75% in either direction in the surrounding domain, whereas a *Refine-* moves the parameter 25%.

Line drawing is a process, i.e., a given drawing may be a proper subset of the completed drawing. Therefore, successive lines may invalidate previously generated constraints. In these cases, some of the history set may need to be deleted.

Line labeling theory aids in making logical inferences about situations where certain constraints are required. This is done by analyzing the set of valid configurations of labels for incoming lines at a junctions. This defines a *junction catalogue*. We use the junction catalogue defined by Malik [Mal87], for curved surfaces. This is useful in determining many occlusion cases automatically, and can automatically determine many constraint types (especially for polyhedral models).

The PCG uses the junction catalogue to weight reconstruction attempts by keeping a record of how often each junction, constraint and parameter configuration occur. We give higher priority to reconstructions that abide by the catalogue rules and occur frequently. Because the catalogue does not consider surfaces that are not piecewise smooth, we do allow junction configurations that fall outside its rule set.

7.3 Generating Initial Constraints

A input contour can generate constraints in its immediate neighborhood or over a region occupied by a set of lines. Here, we define the situations in which both can occur and present methods to generate initial sets of constraints.

Single Contour. Initially, the junction catalogue is consulted to see if any cases can be automatically determined for a line. If not, the system produces three initial guesses for each contour added to the system (as specified in Section 4): a crease case and two silhouette cases (occluding on either side). A crease has normal constraints created for each incident surface. A silhouette has normal constraints created for the incident surface and depth constraints are created that specify that the attached surface lay above the detached surface.

Multiple Contours. A line can affect a region beyond its immediate neighborhood if it extends the surface area of the model, creates a face that defines either a hole or a bounded planar face or modifies an existing face. These situations cause new constraints to be generated simultaneously for multiple contours.

Surface extension occurs any time a loop of contours is created in which part of the loop falls outside the current model domain. This occurs automatically when the closure of the interior, or, the object's silhouette, is first defined. Silhouette constraints are automatically created for all contours that bound the object. Contours identified as silhouettes, but no longer on the boundary of the object domain after surface extension, have their constraints reset. A strong assumption [Var05] about silhouettes is that surfaces all locally fit

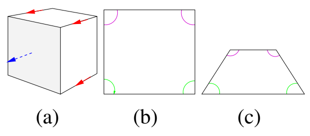

Figure 7: *a)The directions (red) of lines incident to the shaded face help approximate a normal for that face (blue). b) A regular polygon is shown perpendicular to the viewer. c) The interior angles converge to a limit of 0° (green) or 180°(purple) under projection. Measuring how close the interior angles are to their limit allows determining how perpendicular a polygon is to the viewer.*

within an osculating curve (corresponding to surface inflation). This can be emulated by using convex normal constraints for all silhouettes. A weaker, opposite assumption may be made that all silhouette constraints are concave.

A *face* is a bounded closed loop of lines. In the case that faces are added, removed or divided, constraints may be generated for the set of lines that compose each new or altered face. In our system, faces may represent either holes or planar faces. In the case that a face represents a hole, all the lines that compose the face are marked as silhouettes and the interior of the face is removed from the model.

For faces composed of straight lines, the probability that the interior of the face is planar is so strong that most surface reconstruction methods consider *only* this case. A normal constraint is created, at the center of the face, that is used for each contour that bounds the face on the side attached to the face interior. The constraints on the other side are not defined by this method and must be determined later.

The junction catalogue defines certain restrictions on the interpretation of line drawings. Some suppositions may be made automatically, such as those relating to T-Junctions and depth discontinuities [Mal87]. We define a best guess method that attempts to determine if any lines are restricted to a single interpretation based on the junction catalogue. Many useful relationships can be deduced automatically this way. We created a probability density function that records how often each junction configuration occurs in practice, allowing us to make a best guess even for configurations that are not automatically determined by the catalogue.

Initial Normal Estimation. In order to make an initial guess about the normal direction and magnitude, we assume that the model is composed of all orthogonal, regular faces. We do not require our final reconstruction conform to this assumption. In practice, the method presented here yielded reasonable results even for models that do not have these properties. Since mutual orthogonality is assumed, both the normal and adjacent faces must be orthogonal to the current face, so the lines composing adjacent faces can be used to estimate the normal direction. Though perspective projection will distort the relationships of adjacent faces in the drawing, averaging all of the lines that connect to the current face accounts for this (excluding all lines whose directions are similar to lines composing the current face isolates the orthogonal direction). This yields a perspective correct estimate of the direction vector, assuming all lines converge at a vanishing point.

Assuming that all faces are regular allows us to estimate the magnitude of the normal. In a drawing, all internal angles of a regular polygon that is perpendicular to the view vector are equal. As faces are tilted away from the view vector, their internal angles in the image plane converge to either 0° or 180°. By calculating how far away from equal each internal angle is, we can estimate the perpendicularity of each face. We then scale the normal based on this measure.

8 Results

We tested our implementation on a variety of line drawings, shown in Figures 8- 12, of both curved and polyhedral models, some containing features known to be difficult. Shown are both the input drawings and the output models. Reference images used as textures are shown, where applicable.

Figure 8a-c shows a line drawing of a polygonal surface and its corresponding reconstruction. The reconstructed surface approximates the correct planar normals to within about $5-15^\circ$, which we consider acceptable. The lines do not follow the correct projection of straight lines, due to the fact that they are hand drawn, creating errors in the planarity near normal discontinuities. The more accurate the input drawing, the less apparent this error is. However, the polygonal reconstructions are correct in a coarse sense: the planar faces and connections between the polygons are all correct; only errors present in the input lines induce errors in the reconstruction.

Figure 9 shows several reconstructions from drawings of polyhedral objects. In these cases, the face normals are within a few degrees of the correct normals. For simple polyhedral models such as these, the junction catalogue is useful in determining the correct constraint types for each lines. If the initial parameter estimates are good, no user intervention is required to produce the output model. The constraints for all models shown in Figure 9 were determined automatically by the system. Our initial normal estimates were even reasonable for models without orthogonal faces such as Figure 9j. Figure 9p shows that traditional line labels can be deduced using our system.

Figure 8d-g shows a reconstruction of a two point perspective cube. All constraints were determined automatically. The planar normals for the three faces are mutually orthographic to within a few degrees.

Figure 8h-j shows an example of a line drawing with a well known problem corner containing two incoming lines that could be either occluding silhouettes or convex. Our system distinguishes between the two cases with user guidance producing silhouettes that create tears in the surface rather than a solid corner, as shown in Figure 8g. This reconstruction also handles contour splitting at critical points, switching from a crease to a silhouette in the contour interior. Critical points are outlined by blue rectangles in Figure 8e. Our contour splitting algorithm, while effective, was a bit overzealous in practice. Often, it made regular cases more difficult to process since some lines that could have been handled with a single label were split. It is possible that some further user guidance in this area would be of benefit for determining problem areas.

Figures 11- 12 show models reconstructed from drawings of curved surfaces. Each drawing is based at on a source image, which is subsequently used to texture the output model. For a simple drawing, such as Figure 11b, the junction catalogue can be used to determine the constraint type of every line automatically. Using our systems user interface, depth relationships and exact normals can be defined more precisely, but in this instance, our initial parameter estimates for normal and depth values are reasonable. It is impossible to make a claim that our values are correct since they are subject to the artists perception.

Figure 11j shows the use of a hole within the model domain (on the bears mouth) demonstrating that our system can handle topologically complex frontal geometry.

While the reconstruction of the head in Figure 12 is far from perfect, note that the input drawing is simplified and does not contain many of the discontinuities present on an actual human head. For commonplace objects, many discontinuities are *assumed* as opposed to expressed. The artist assumes that viewers know such discontinuities exist and does not include them. A drawing with some contours assumed in Figure 12b is shown in Figure 12l. In this case, the quality of the reconstruction is limited by the lack of relevant input data. There may be no practical solution to this since the system can not guess what data might be missing. In the worst case, users can be prompted for more detail if they are unhappy with the quality of the reconstruction.

Furthermore, many real world objects, such as the head, are not C^2 continuous within bounded surface elements and cannot be fully represented with piecewise smooth models. Strokes related to curvature are rarely addressed in previous research and we view the extension of this methodology to curvature discontinuities as future work. Yet despite these limitations, the features of the head model roughly correspond to the features of an actual human head. Therefore, we view this example as a success since the reconstruction method creates a good model despite its limitations.

The time complexity of our system scaled with the complexity of the input drawing and was limited only by the time taken by the solver to converge. Drawing operations occurred in real time and all non-solver related operations took neglible calculation time. The solver typically required 2-5 seconds to converge for a reconstruction with five or fewer contours. Average convergence time for a single reconstruction was about 5 seconds, though the gross differences between the reconstructions being compared was usually visible after a few seconds. Highly complex models required up to 10-20 seconds to converge but we gained a dramatic speedup by initializing new candidate reconstructions with the last known mesh selected as *Good*. This took computation time down to around 5-10 seconds for complex models. Typically, 2-3 candidate solutions were created simultaneously. All timings are for a 1.8 Ghz Pentium 4.

Simple models such as those shown in Figure 8 required under a minute to create. The models shown in Figure 9 had constraints that were automatically deduced by the system and required only the time for the solver to converge on the solution which was typically 10-15 seconds for the full model. The bear model took about 4 minutes to create. The head model shown in Figure 12 took about 6 minutes to create. This drawing was saved and running the algorithm on the completed drawing required about 3 minutes to recreate.

Users required very little time discerning which candidate reconstructions were appropriate for the scene at a high level. They did require convergence of the solver and several seconds for comparison when attempting to pinpoint exact normal and depth parameters, since candidates were similar at that scale. We found that users selection was more efficient when starting from a completed drawing. This may be because the PCG attempted to change large regions first on completed drawings, whereas interactively created drawings updated constraints one detail at a time.

9 Future Work

We have presented a method of generating constraints that asks for extensive user input for analysis and verification. It would be preferable to make those decisions without user input, where possible. We feel this area may be improved significantly in the future as the understanding of human vision and line drawing interpretation improves.

The simplified model of drawing we consider is not adequate to fully represent all drawings that occur in practice so extending our system to encompass other types of lines such as those generated by curvature, color, lighting, and texture is desirable. Many surface reconstruction methods use a beautification step to clean up the mesh after reconstruction, so we would like to explore this option to lessen the influence of hand-drawn errors in the output surface or as a pre-process in the input drawings.

Our ultimate target is to produce a system that automatically reconstructs models from drawings created by an artist that imposes no interference on the artistic process and requires no extra input or knowledge from the artist whatsoever. The presented system is a first step towards that goal.

References

[AGB04] ALEXE I. A., GAILDRAT V., BARTHE L.: Interactive Modelling from Sketches using Spherical Implicit Functions. In *AFRIGRAPH* (03-05 novembre 2004), ACM.

[Clo71] CLOWES M. B.: On seeing things. *Artificial Intelligence 2* (1971), 79–116.

[CPM04] COMPANY P., PIQUER A., M.CONTERO: On the evolution of geometrical reconstruction as a core technology to sketch-based modeling. *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2004).

[dAJ03] DE ARAUJO B., JORGE J.: Blobmaker: Free form modelling with variational implicit surfaces. *Proceed-*

ings of 12th Encontro Português de Computação Gráfica (2003).

[HAA97] HORRY Y., ANJYO K.-I., ARAI K.: Tour into the picture: using a spidery mesh interface to make animation from a single image. In *SIGGRAPH '97* (1997), pp. 225–232.

[HR85] HOFFMAN D. D., RICHARDS W. A.: Parts of recognition. In *Visual Cognition*, Pinker S., (Ed.). MIT Press, London, 1985, pp. 65–96.

[Huf71] HUFFMAN D.: Impossible objects as nonsense sentences. *Machine Intelligence 6* (1971), 295–323.

[IMT99] IGARISHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3d freeform design. In *SIGGRAPH* (1999).

[JJAR97] JOSEPH J. ATICK P. A. G., REDLICH N.: Statistical approach to shape from shading: Reconstructin of 3d face surfaces from single 2d images.

[Kan98] KANG S.: Depth painting for image-based rendering applications, 1998.

[KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-form sketching with variational implicit surfaces. *Computer Graphics Forum 21*, 3 (Sept. 2002), 585–594.

[Koe84] KOENDERINK J. J.: What does the occluding contour tell us about solid shape?, 1984.

[Koe98] KOENDERINK J. J.: Pictorial relief, 1998.

[KvDCL96] KOENDERINK J. J., VAN DOORN A. J., CHRISTOU C. G., LAPPIN J. S.: Shape Constancy in Pictorial Relief. *Perception 25*, 2 (Feb. 1996), 155–164.

[LB90] LAMB D., BANDOPADHAY A.: Interpreting a 3d object from a rough 2d line drawing. *Proceedings of Visualization90* (1990), 59–66.

[Lip98] LIPSON H.: Computer aided 3d sketching for conceptual design, phd thesis, 1998.

[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.

[LS02] LIPSON H., SHPITALNI M.: Correlation-based reconstruction of a 3d object from a single freehand sketch, 2002.

[LZS01] LI ZHANG GUILLAUME DUGAS-PHOCION J.-S. S., SEITZ S. M.: Single view modeling of free-form scenes. In *Proc. Computer Vision and Pattern Recognition* (2001).

[Mac73] MACKWORTH A. K.: Interpreting pictures of polyhedral scenes. *Artif. Intell. 4*, 2 (1973), 121–137.

[Mal87] MALIK J.: Interpreting line drawings of curved objects. *International Journal of Computer Vision* (1987), 73–103.

[MAP*97] MARKS J., ANDALMAN B., P.A.BEARDSLEY, W.FREEMAN, S.GIBSON, J.HODGINS, T.KANG, B.MIRTICH, H.PFISTER, W.RUML, K.RYALL, J.SEIMS, S.SHIEBER: Design galleries:a general approach to setting parameters for computer graphics and animation. In *SIGGRAPH* (1997), pp. 389–400.

[OCDD01] OH B. M., CHEN M., DORSEY J., DURAND F.: Image-based modeling and photo editing. In *SIGGRAPH 2001* (2001), pp. 433–442.

[PTKK] PHILLIPS F., TODD J. T., KOENDERINK J. J., KAPPERS A. M. L.: Perceptual representation of visible surfaces.

[RCZ96] ROBERT C. ZELEZNIK KENNETH P. HERNDON J. F. H.: Sketch: An interface for sketching 3d scenes.

[RH78] R.SHAPIRA, H.FREEMAN: Computer description of bodies bounded by quadric surfaces from a set of imperfect projection, 1978.

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

[TI01] T. IGARISHI J. F. H.: A suggestive interface for 3d drawing. In *Symposium on User Interface Software and Technology* (2001).

[Tur74] TURNER K.: Computer perception of curved objects using a television camera, 1974.

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

[Var05] VARLEY P. A. C.: The state of the art in line drawing interpretation, 2005. http://uk.geocities.com/pacvarley/StateOfTheArt.html.

[VM00] VARLEY P. A. C., MARTIN R. R.: A system for constructing boundary representation solid models from a two-dimensional sketch. In *GMP* (2000), pp. 13–32.

[VM02] VARLEY P. A. C., MARTIN R.: Estimating depth from line drawings. In *Proc. 7th ACM Symposium on Solid Modeling and Applications* (2002), pp. 180–191.

[VSM04] VARLEY P. A. C., SUZUKI H., MARTIN R. R.: Making the most of using depth reasoning to label line drawings of engineering objects, 2004.

[VYJH04] VARLEY P. A. C., Y.TAKAHASHI, J.MITANI, H.SUZUKI: A two-stage approach for interpreting line drawings of curved objects. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2004).

[Wil90] WILLIAMS L.: 3d paint. In *SI3D '90: Symposium on Interactive 3D graphics* (1990), pp. 225–233.

[Wil91] WILLIAMS L.: Shading in two dimensions. In *Graphics Interface '91* (1991), ACM Press, pp. 143–151.

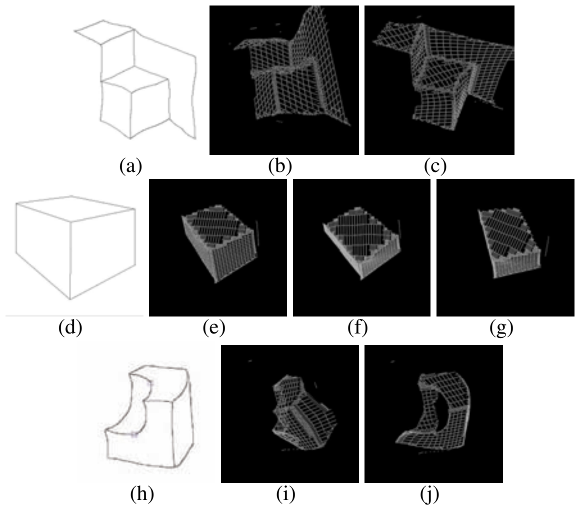

Figure 8: *Various results. Critical points in e) are outlined in blue. a,e,h) are the input drawings.b-d,f,g,i,j) are the output models.*

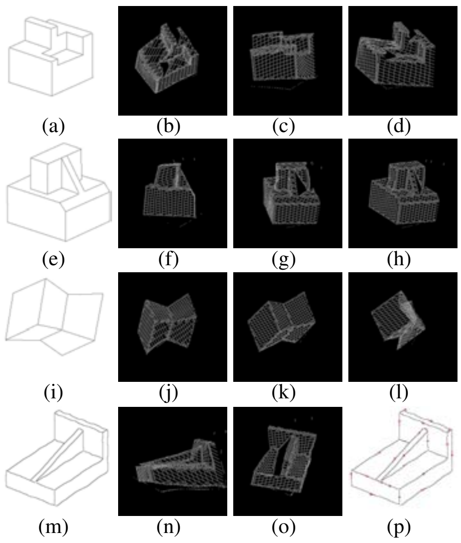

Figure 9: *Input drawings of polyhedral surfaces are shown at left. Output models are shown in the three right columns. p) Line labels can be calculated based on the constraints produced by our system.*

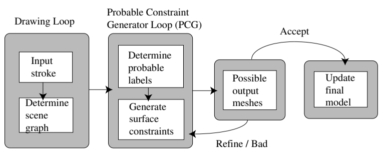

Figure 10: *This diagram illustrates the process of our system and the relationship between the principle components.*

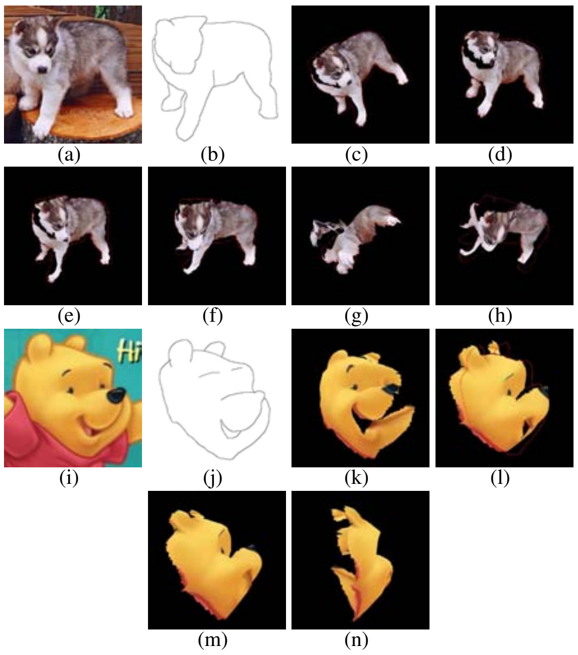

Figure 11: *a,i) source images. b,j) drawings based on the source images. c-h,k-n) output models based on the input drawings, texture mapped with the source images.*

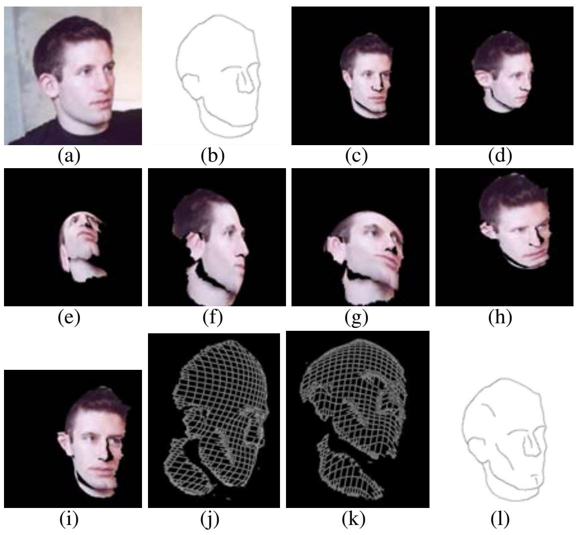

Figure 12: *a) source image. b) input drawing. c-i) output models, texture mapped with the source image. j-k) output meshes. Such models are often difficult to produce since many discontinuities present in a) are not represented in b). This occurs because some lines, such as those shown in l), are assumed by both viewer and artist when the object depicted is well understood by both.*