EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2004)
John F. Hughes and Joaquim A. Jorge (Editors)

A Two-Stage Approach for Interpreting Line Drawings of Curved Objects

P.A.C.Varley Y.Takahashi J.Mitani H.Suzuki
Department of Precision Engineering, The University of Tokyo
Tokyo, Japan
{pvarley,takahashi,mitani,suzuki}@den.rcast.i-tokyo.ac.jp

ABSTRACT

We describe a two-stage approach for interpreting line drawings of curved objects. In the first stage, the user enters a natu-ral line drawing of a polyhedral template; this is automatically interpreted as the corresponding polyhedral object. In the second stage, the user enters freehand curves; by relating these to the template, a curved object can be constructed

Categories and Subject Descriptors (according to ACM CCS):): J.6: Computer Aided Engineering [Computer Aided Design]

1. Introduction

Studies such as Jenkins [Jen92] have shown that it is common practice for design engineers to sketch their ideas on paper before entering them into a CAD package. Clearly, valuable time and effort could be saved if a computer could interpret the engineer's initial concept drawings as solid models. Furthermore, if this conversion could be done within a second or two, it would give helpful feedback, further enhancing the designer's creativity [Gri97].

In an earlier paper [VSMM00], we described two programs: one, RIBALD, which interprets line drawings of simple solid objects, and another, 3D SKETCH, which interpreted freehand drawings in terms of a template. In this paper we describe how, by using the descendents of these two programs as components of a two-stage approach, it is possible to produce a system which interprets line drawings as curved objects. The individual components have previously been described elsewhere ([Var03],[Tak04]); the novel contribution of this paper is their combination to provide a flexible system for input of curved drawings.

RIBALD only interprets polyhedra. Its assumptions are reasonably flexible: the drawing represents a single manifold polyhedral object viewed from its most informative viewpoint and from a general viewpoint where no small change in the viewpoint would result in the topology of the drawing. It does not assume that the drawing is error-free.

Interpretation of drawings containing curved lines presents several unsolved problems even for perfect drawings. Many of the simplifying assumptions made in interpreting polyhedra (such as that if two edges meet the same two faces, the edges must be collinear) do not hold for curved objects.

Another unsolved problem is that in the domain of curved line drawings even simple facts such as the shape of a curve often have non-local consequences. This can become a serious obstacle to interpretation, as can be seen by considering Yonas's curves (Figure 1 [BT81]), in which turning the bottom line from a straight line to a curve changes the perception of the curved top lines.

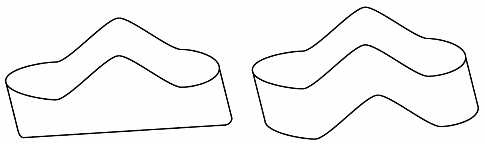
Figure 1: Yonas's Curves [BT81]

Another example given by Barrow and Tenenbaum [BT81] illustrates the problem of distinguishing similar drawings which depend on subtle mathematical points for their interpretation. In Figure 2, both the slice of cake (left) and the rocket nose-cone (right) are valid drawings; they are distinguished by the ``subtle mathematical difference'' that in the rocket nose-cone, the bottom curve is tangential to the vertical lines.

Where freehand drawing errors are allowed, as is necessary if line information is produced by processing a freehand sketch, such subtle differences can easily be missed (Barrow and Tenenbaum are concerned with drawings derived from processed greyscale information; such drawings may also contain small errors, and their point becomes of even more importance when applied to hand-drawn sketches).

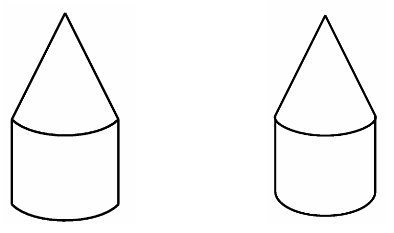

Figure 2: Slice of Cake, Rocket Nose Cone [BT81]

A further difficulty in interpreting line drawings of curved objects is that many of the established tools for interpreting polyhedra cannot be used. For example, line labelling (Section 3.2) is a well-established and useful technique for interpreting line drawings of polyhedra. It relies on the fact that in drawings of polyhedra, each line has the same label (convex, concave or occluding) throughout its length. This is not the case with drawings of curved objects, as can be seen in Figure 3 (where the curved line changes from convex to occluding).

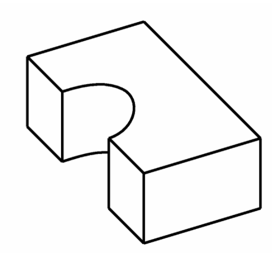

Figure 3: Invisible Junction

Similarly, many of the fundamental assumptions made by existing approaches to constructing hidden topology (Section 3.6), such as the supposition that an edge joins two vertices, do not remain valid in the domain of curved objects (consider a cylinder, with edges but no vertices).

Considerations such as these suggest that a single-stage process for interpreting curved drawings without user intervention may well be impossible and is certainly not likely to be seen in the near future.

The field of 2D drawing of 3D scenes has provided some work of interest. For example, Tolba et al [TDM99] use a limited form of 2½D reasoning to allow a scene, once drawn, to be viewed from different angles.

However, there remains the problem of deducing the hidden part of the object. Some investigators have chosen to allow unlimited user intervention. For example, *Teddy* [IMT99] (see Figure 4) uses an entirely different approach to creating solid models. The user may create, modify or delete parts of the model as viewed from the current viewpoint, and may also change the viewpoint. The system of Ferley et al [FCG00], although using 3D rather than 2D input devices and adding a few extra software tools, is essentially similar. Schuresko [Sch99] applies the techniques of *Teddy* to model human faces, in order to make the successes and limitations readily apparent.

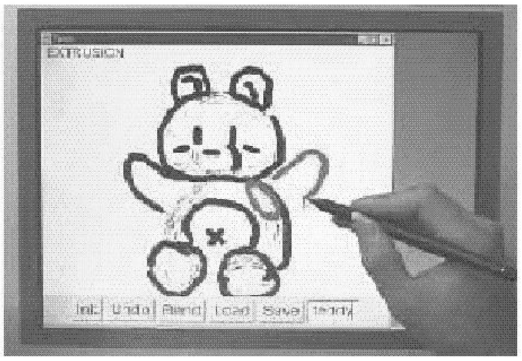

Figure 4: Teddy Bear Design [IMT99]

We prefer, instead, to limit the need for user intervention. Ideally, we should wish to interpret line drawings of curved objects without any user intervention at all. However, in view of the problems noted above, we propose instead a two-stage approach for designing and creating solid models of curved objects: firstly, draw a polyhedral template for the desired object, and secondly sketch the corresponding curved object in the same orientation as the template. We describe the two components of such an approach separately.

To be useful for interpreting curved drawings, the template must follow certain conventions. The conventions we assume are that (a) the template must have the same vertex/edge graph as the curved drawing and (b) it must be possible, by matching the extreme (top, bottom, left, right) vertices of the template and drawing, to establish the translation between the two.

Section 2 outlines our approach. Section 3 describes how we can create templates. Section 4 describes how a freehand sketch including curved lines can be interpreted by means of a template. Section 5 shows examples of curved objects modelled using this two-stage process. Section 6 discusses limitations of our proposed method. Section 7 summarises our conclusions.

2. Outline of Approach

The outline of our proposed approach is as follows (see Figure 5 for illustrations):

a) Create a line drawing of a polyhedral template by means of which drawings of curved objects are to be interpreted. This drawing should have the same vertex-edge graph as the curved drawings, and should be from a similar viewpoint.

b) Use the methods of Section 3 to interpret this line drawing as a polyhedron.

c) Create a line drawing of the curved object.

d) Use the methods of Section 4 and the template from (b) to interpret this drawing as a curved object, and create the corresponding triangulated mesh model.

e) Repeat from (c) as often as is desired.

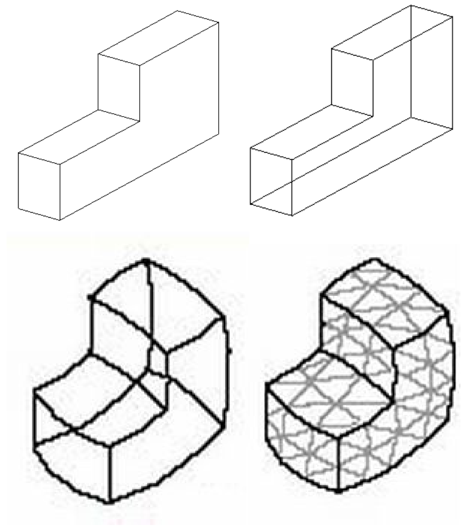
Figure 5: Illustration of Outline Method

3. Template Creation

3.1 Overview.

The first of our components is one which takes a line drawing of a polyhedral object and produces from it a solid model of the object which the drawing represents.

Although the current state of the art, as exemplified by RIBALD [VSMM00, Var03], is not capable of reliably interpreting complex drawings, it can reliably take line drawings of typical object templates and produce 3D models from them.

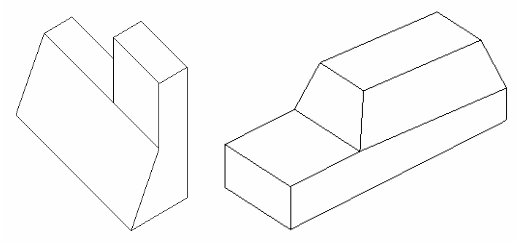
Figure 6: Two Engineering Object Templates

The problem is to produce a model of the 3D object the engineer would regard as the most reasonable interpretation of the 2D drawing, and to do so quickly. Obviously, there are infinitely many objects which *could*, if viewed if viewed from a particular viewpoint, result in drawings such as those in Figure 6. In practice, an engineer would usually be in little doubt as to which was the intended interpretation.

For this reason, the problem is as much heuristic as geometric: it is not merely to find a geometrically-realisable solid which corresponds to the drawing, but to find the one which corresponds to the engineer's expectations.

Briefly, the approach taken by RIBALD is as follows (for more details, including a discussion of alternative approaches, see [Var03,VMS04b]):

a) Label the lines in the drawing, to determine which are convex, which are concave, and which are occluding. This is described in Section 3.2.

b) Determine which pairs of lines in the drawing are intended to be parallel in 3D. This is described in Section 3.3.

c) Inflate the drawing to 2½D by determining *z*-coordinates for each vertex. This is described in Section 3.4.

d) Determine any symmetry elements (mostly planes of reflection) in the object. This is described in Section 3.5.

e) Classify the drawing (e.g. extrusion, normalon, general case). This is also described in Section 3.5.

f) Complete the object topology by determining the topology of the hidden part of the object. This is described in 3.6.

g) RIBALD finishes by tidying the geometry of the completed object. This is not relevant to our current purpose, since intermediate templates need not be tidy, and it is not described in this paper. See the literature on ``beautification of solid models'' problem, which has applications outside the field of line drawing interpretation, such as reverse engineering [VM02].

3.2 Which Lines are Convex/Concave

Line labelling is the process of determining whether each line in the drawing represents a convex edge, a concave edge or an occluding edge. Unfortunately, there is no single best solution to this problem.

For drawings of genus zero trihedral objects, the line labelling problem was essentially solved by Huffman [Huf71] and Clowes [Clo70], who elaborated the catalogue of valid trihedral junction labels. This translates line labelling into a discrete constraint satisfaction problem where the constraints are the 1-node constraint that each junction must have a label in the catalogue and the 2-node constraint that each line must have the same label throughout its length.

The Clowes-Huffman catalogue for *L*-, *W*- and *Y*-Junctions is shown in Figure 7. + indicates a convex edge; - indicates a concave edge; an arrow indicates an occluding edge with the occluding face on the right-hand side of the arrow.

In trihedral objects, *T*-junctions (see Figure 8) are always occluding. (Although, as in Table 1, these are always listed as part of the trihedral catalogue, they are general to all catalogues as occluding *T*-junctions do not correspond directly to vertices.)

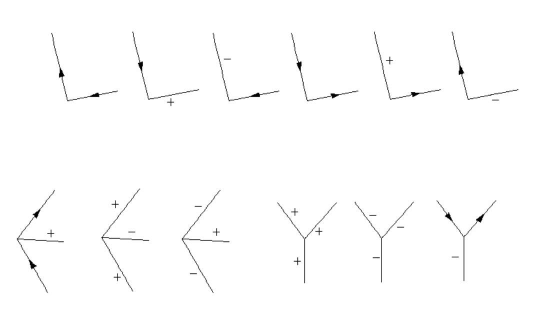

Figure 7: Clowes-Huffman Catalogue

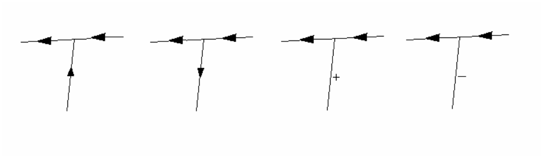

Figure 8: Occluding *T*-junctions

For trihedral objects, algorithms for Clowes-Huffman line labelling such as those of Waltz [Wal72] or Kanatani [Kan90], although **O**(e^n) in theory, are usually **O**(n) in practice [PT94]. It is believed that the time taken is more a function of the number of legal labellings than of the algorithm [Var03], and for genus zero trihedral objects there is usually only a single legal labelling.

For this reason, extension to the 4-hedral general case presents problems. The catalogue of 4-hedral junction labels is much larger [Var03]: see Table 1. For this reason, there may be many valid labellings (perhaps millions) for a drawing. Non-trihedral line labelling using the simple combinatorial algorithms used for solving discrete CSPs is unreliable and can be very slow (see [Var03,VMS04b]).

Junction	3-hedral	Ext 3-hedral	4-hedral	Total
L	6	0	2	8/16
T	(4)	4	12	20/64
W	3	0	25	28/64
Y	5	0	27	32/64

Table 1: Number of Entries in Junction Catalogues

The current state of the art for labelling non-trihedral drawings is that described in [VMS04a]. This requires a preliminary inflation. From the resulting preliminary frontal geometry, predictions are made as to whether each line is convex, concave or occluding. Since this preliminary inflation cannot make use of line labels or parallel line information, these predictions are not in themselves entirely reliable, so they are used as seeds in a probabilistic labelling algorithm. The probabilities so generated are generated are propagated around the lines and junctions of the drawing, and the combined junction catalogues of Table 1 are used to reject illegal combinations. This latter method can be used to label the two examples of Figure 6 (amongst others) correctly.

Choice between the two methods, Clowes-Huffman and preliminary inflation plus propagation, is generally straightforward: if a drawing can be labelled using Clowes-Huffman, it should be. RIBALD defaults to choosing its labelling method automatically using that rule. There are rare occasions when, although a drawing can be labelled as trihedral, it should be labelled as non-trihedral; requiring the user to specify this in advance on such rare occasions, although theoretically unsatisfactory, is hardly onerous.

3.3 Which Lines are Parallel?

Which lines in a drawing are intended to correspond to edges parallel in 3D? If user inaccuracies are allowed, this is not an easy question to answer.

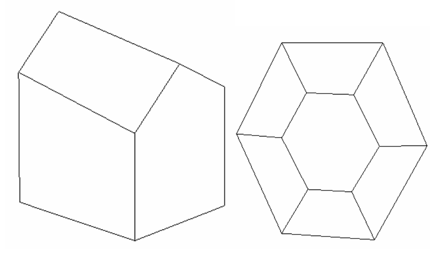

Figure 9: Which Edges are Parallel in 3D?

It is, for example, obvious to a human which lines in the two drawings in Figure 9 are intended to be parallel in 3D and which are not, and it proves to be easy enough to discover rules by means of which these two drawings can be interpreted correctly (see [Var03]). However, there remain outstanding problems. See [VMS04b] for a discussion of the current state of the art. Despite such occasional failures, existing methods are good on the whole.

3.4 Inflation to 2½D

Inflation is the process of converting a flat 2D drawing into 2½D by assigning *z*-coordinates (depth coordinates) to each vertex, producing a *frontal geometry*.

There are many ways of inflating drawings to 2½D. Here, we illustrate the simplest: use *compliance functions* [Lip96] to generate equations linear in vertex depth coordinates, and solve the resulting linear least squares problem, as used in [Var03].

Many compliance functions can all be translated into linear equations in *z*-coordinates. [Lip96] provides a list of several. Of these, the most useful are:

Perkins's **Cubic Corners** [Per68], sometimes called **corner orthogonality**, assumes that a *W*-junction or *Y*-junction corresponds to a vertex at which three orthogonal faces meet. See Figure 10: the equation relating depth coordinates z_A and z_a to angles C and D is linear in the z-coordinates.

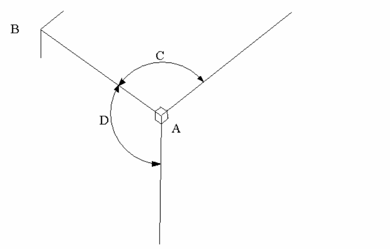

Figure 10: Cubic Corner

Line Parallelism assumes that two edges have been identified as being parallel in 3D. The equation relating the four z-coordinates of the vertices at either end of the two edges is obviously linear in the z-coordinates. Line parallelism is not, by itself, inflationary: there is a trivial solution ($z=0$ for all vertices) which meets line parallelism constraints.

Vertex Coplanarity assumes that four vertices have been identified as being coplanar. Again, the equation relating the z-coordinates of the four vertices is obviously linear, and the coefficients are easily obtained from 2D geometry. Vertex coplanarity is also not, by itself, inflationary: the trivial solution ($z=0$ for all vertices) also meets vertex coplanarity constraints. General use of vertex coplanarity is not recommended (Lipson [Lip98] points out that where three consecutive vertices on a face are collinear, successive use of four-vertex coplanarity does not guarantee a planar face). However, it is invaluable for cases such as those shown in Figure 11, where without it the linear system of depth equations would be disjoint, with infinitely many solutions.

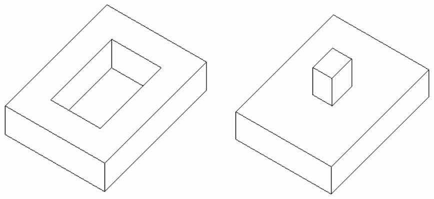

Figure 11: Drawings with Disjoint Subgraphs

Junction-Label Pairs [Var03] assume that pairs of junctions with particular labels have the same depth implications they would have in the simplest possible drawing containing this pair. It generates an equation relating the vertex depths of the ends of the line based on the junction labels of those vertices.

Given correct inputs, inflation is the most reliable of the stages of processing described in this paper. Although it occasionally fails to determine correctly which end of a line should be nearer the viewer, such failures are in cases where a human would also have difficulty. The only systematic case where the approach of a linear system of compliance functions fails is that of the Platonic and Archimedean solids, and a known special-case method (Marill's MSDA [Mar91]) is successful for these.

Figures 12 and 13 illustrate the results of inflating two of our test drawings, as viewed from different viewpoints.

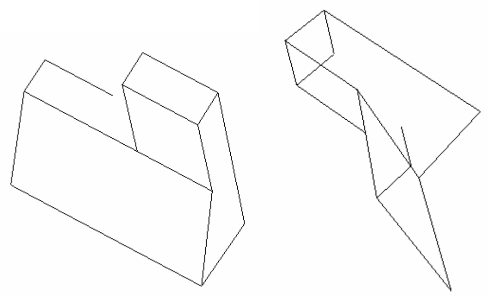

Figure 12: Inflation to 2½D

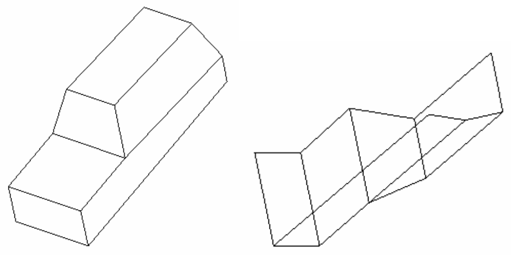

Figure 13: Inflation to 2½D

3.5 Symmetry and Classification

Once a drawing has been inflated, it is straightforward both to identify potential symmetry elements and to assess their merit.

Identification of potential symmetry elements is essentially a topological task. Each face centre is considered as the possible location of an axis of rotational symmetry and of planes of mirror symmetry. Candidate symmetry elements which do not match the topology in the region of the face under consideration are rejected.

Assessment of the merit of symmetry elements is twofold: topological and geometric. Topological assessment considers questions such as whether the postulated symmetry operation propagates as expected across the visible part of the object. Geometric assessment considers whether the 3D location of visible vertices matches those which would be predicted by the assumption that the postulated symmetry operation is genuine.

Determining the major plane of symmetry of the object is required in Section 4: Takahashi's method of solving the problem of curve ambiguity requires this information. Determination of other symmetry elements is not strictly necessary (the other methods of Section 3 should be sufficient without it) but can help both in avoiding mistakes

mistakes when constructing of hidden topology and in tidying the geometry of this hidden topology.

RIBALD's current algorithm for detecting symmetry is known to be suboptimal. It is slow ($O(n^4)$, whereas Sugihara has shown that an $O(n^2)$ algorithm must exist) and in some cases unreliable. In particular, although it uses topological reasoning when propagating matches across the visible part of the object, the existing implementation only applies this topological reasoning at trihedral vertices. As will be seen in Section 6.1, it can also fail to identify mirror planes which run along consecutive edges.

Classification attempts to classify the depicted template (e..g extrusion, normalon, trihedral). Again, this is not strictly necessary, there are several short-cuts which can be taken when constructing hidden topology if the template can be classified in this way, making the process both faster and more reliable.

3.6 Determine Hidden Topology

Once the frontal geometry has been determined, the next stage of processing is to add the remaining topology. The method is essentially that presented in [VSMM00]: firstly, add extra edges to complete the wireframe, and then add faces to the wireframe to compete the object, as follows:

- While the wireframe is incomplete:
    - Project hypothesised edges at each incomplete vertex along the appropriate axes
    - Eliminate any edges which would be visible at their points of origin
    - Find the locations where the remaining edges intersect, assigning merit figures according to how certain it is that the edges intersects at this location (e.g. an edge which intersects only one other edge gives a higher merit figure than an edge which intersects two or more other edges)
    - Reduce the merit for any locations which would be visible (if drawing errors were not allowed, such locations could be eliminated)
    - Find the location at which the crossing merit is greatest
    - Add a vertex at this location, and the hypothesised edges which crossed at this location, to the known object topology

The process of completing the wireframe topology varies in difficulty depending on the type of object portrayed in the drawing. This paper illustrates two special-case types of object, extrusions and normalons, and the general case. In some cases (e.g. where the object is symmetrical or includes a recognized feature), more than one vertex can be added in one iteration; these are described in [Var03]. Such cases help both to speed up the process of determining the complete wireframe and to make it more reliable.

Completing the topology of extrusions from a known front end cap is straightforward (and useful, since many curved objects can be approximated by extrusions). Figure 14 shows a drawing and the corresponding completed extrusion wireframe.

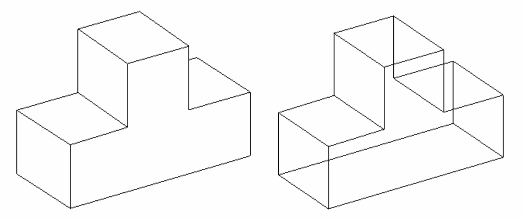

Figure 14: Extrusion

The knowledge that the object is a normalon makes it easier to reconstruct the wireframe correctly, since when hypothesised edges are projected along axes, there will usually be only one edge projected from any particular incomplete vertex. Figure 15 shows a drawing of a normalon and the corresponding completed wireframe.

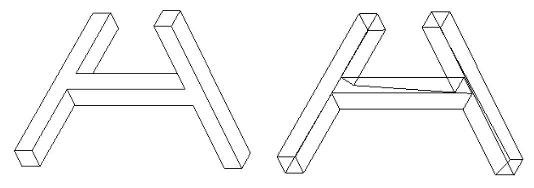

Figure 15: Normalon

Similarly, the knowledge that the object is trihedral helps reduce the number of possible edges projected at each vertex (again, there will be at most one from each incomplete vertex) and thus helps in reconstructing the correct wireframe. Figure 16 shows a drawing of a trihedral object and the corresponding completed wireframe. Note that although the resulting wireframe is topologically correct, geometric inaccuracies in the frontal geometry are magnified when this information is used (i) to define the mirror plane and then (ii) to reflect known vertices through this mirror plane [Var03].

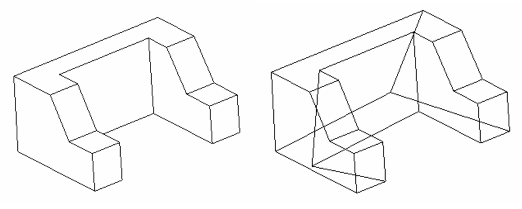

Figure 16: Trihedral Object [Gri97]

However, in the general case, where the object is neither a normalon nor trihedral, there is the significant difference that hypothesised edges may be projected in any direction parallel to an existing edge. Even after eliminating edges which would be visible, there may be several possibilities at any particular incomplete vertex. In practice, the large number of possible options rapidly becomes too confusing and it is easy to choose an incorrect crossing-point at an early stage. Although such errors can sometimes be rectified by backtracking, the more common result is a valid but unwanted wireframe. Only very simple drawings can be processed reliably. Figure 17 shows a general-case object

object and the corresponding completed wireframe; this represents the limit of the current state of the art.

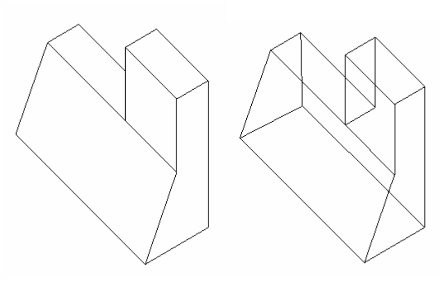

Figure 17: General Case Object

Finally, we note that adding additional faces to a wireframe topology for which the frontal geometry is already known is straightforward. We use repeated applications of Dijkstrå's Algorithm [Dij59] to find the best loop of unallocated half-edges for each added face, where the merit for a loop of half-edges is based both on the number of half-edges required (the fewer, the better) and their geometry (the closer to coplanar, the better). Colour Slide I shows the completed topology created for two of our test objects, as viewed from various viewpoints.

3.7 Tailoring RIBALD

The version of RIBALD described previously [Var03] is a general-purpose program. When used in this specific application, it is possible and beneficial to tailor it to match the requirements of Section 4. In particular, there are two rules which can be added.

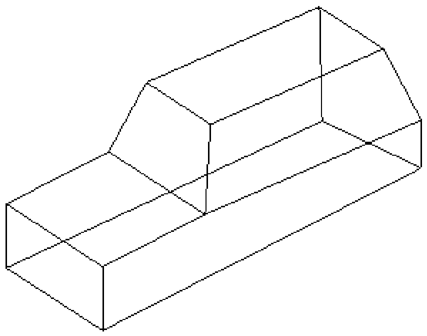

Figure 18: Wrong Interpretation

Firstly, since Takahashi's approach (Section 4) requires that the template be mirror-symmetric, RIBALD must use the best plane of mirror symmetry as its starting point when constructing hidden topology. In other applications and with other drawings, the merits of preferring symmetry to other clues can be argued either way, but for this application Figure 18 is unequivocally the wrong interpretation, whereas Figure 19 is an acceptable interpretation.

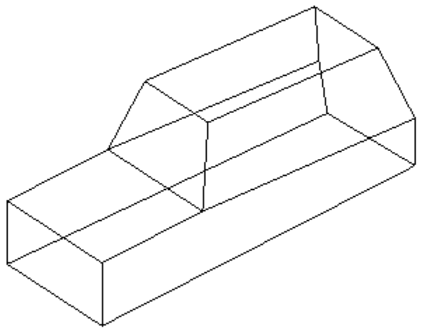

Figure 19: Better Interpretation

However, Figure 20 is a better interpretation still. Although it is not clear at this point whether the added line is necessary or not, it *might* be useful, and since the final output from the second stage will be a triangulated mesh model, adding the line will do no harm.

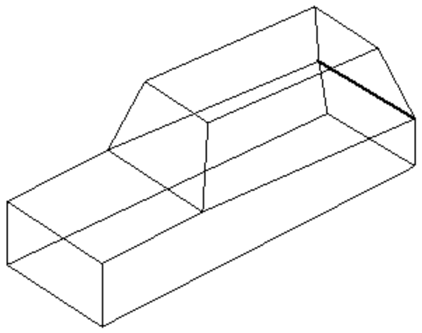

Figure 20: Best Interpretation

In general, for this application, it is always best to add such cross-mirror lines, provided only that they are not in positions where they ought to be visible in the original line drawing. (Figure 26 shows another example where adding such lines would clearly be correct.)

3.8 Alternative Approaches

We have described in detail an approach for creating object templates from natural line drawings, since this is our preferred approach. However, creating such templates from wireframe drawings is a fully viable alternative [Lip98, CCG99].

4. Curved Objects

Our second stage is to interpret a curved drawing by means of a polyhedral template. Note that a single template can be used to interpret several curved drawings, as will be seen in the examples in Section 4.

Mitani's original 3D SKETCH [Mit99] applied mirror symmetry to enable reconstruction, in an approach based on Furushima's method [Fur93]. This illustrated the power of the concept but was limited to a single template, that in Figure 21.

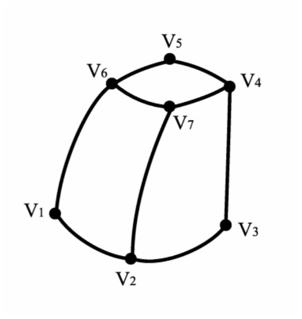

Figure 21: Single Template [Mit99]

Furthermore, the approach also made several assumptions about the template and the geometry of any object created from it. While some assumptions are certainly necessary, those made in [Mit99] were specific to the one template and not readily extended to a variety of templates.

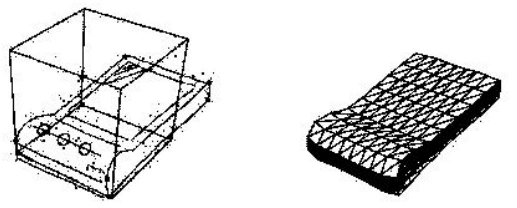

Figure 22: Design from Fixed Template [Mit99]

Takahashi [Tak04] has taken this work further, generalising it so that it allows for alternative templates, and improving it to remove the requirement for considerable specific knowledge about how to interpret a particular template. Where some template-specific knowledge is still required (such as the predominant plane of mirror symmetry), this is provided in the template, not built into the algorithm.

However, the outline concept remains the same:

a) The vertex/edge graph of the curved sketch is matched to that of the template, using standard graph-matching techniques. The extreme (uppermost, bottommost, leftmost and rightmost) vertices of each as a starting-point.

b) The edges of the template are bent so as to match the curves in the sketch. This is discussed below.

c) The resulting curved surfaces are smoothed and subdivided to produce a triangulated mesh model, using standard mesh subdivision techniques (initial triangulation of the template faces followed by Loop subdivision [Loo87]).

When bending edges to match the sketched curves, the principle is that each point on the curve is in a 3D location which would, in 2D projection, appear as sketched. Although theoretically an insoluble problem – there are an infinite number of such 3D locations – there are two candidate locations which are clearly preferable to the infinite number of alternatives. See Figure 23 for an illustrative example.

Curve ambiguity, the choice between these two locations, remains an outstanding problem. Does the curved line in the central drawing of Figure 23 represent a convexity in the “windscreen” face (left-hand drawing) or a concavity in the “side” face (right-hand drawing)?

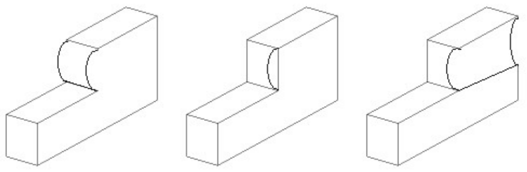

Figure 23: Curve Ambiguity

Takahashi’s solution to this problem is to assume that curves drawn in this manner always represent symmetrical distortions across the object’s major plane of mirror symmetry. If the alternative interpretation is desired, this is indicated by adding an extra line. See Figure 24.

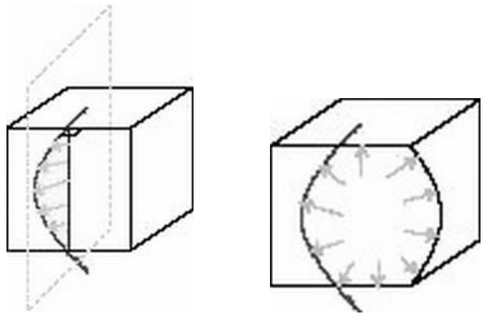

Figure 24: Resolving Curve Ambiguity [Tak04]

While functional, this solution is not ideal, and further work in this area is required in order to present a more natural, intuitive interface which allows for resolution of curve ambiguity.

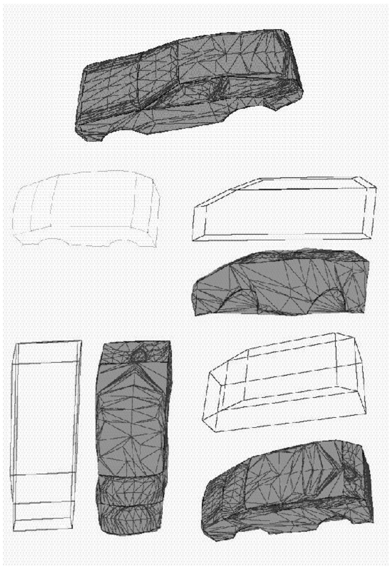

Figure 25: Car Body Design [Tak04]

5. Examples

Examples of completed templates have already been given in Section 3.

Figure 25 and Colour Slide II [Tak04] show the output from the second stage, and illustrate one possible application of this approach: car body design.

6. Limitations

The limitations of our approach come from two sources: the limitations of RIBALD, and the limitations of Takahashi's program.

6.1 Limitations of RIBALD

Although RIBALD's ability to label drawings correctly, and to determine the hidden topology of objects, is limited, these limitations do not appear important in practice. In general, the drawings used for templates are those which RIBALD handles most successfully.

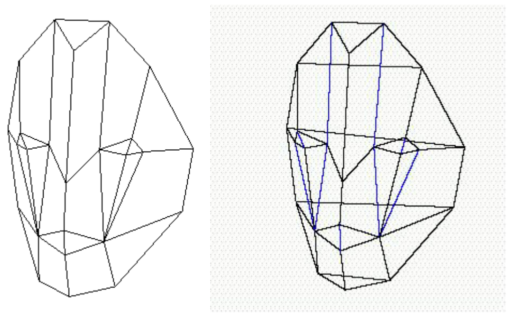
Figure 26: Drawing and Template [Tak04]

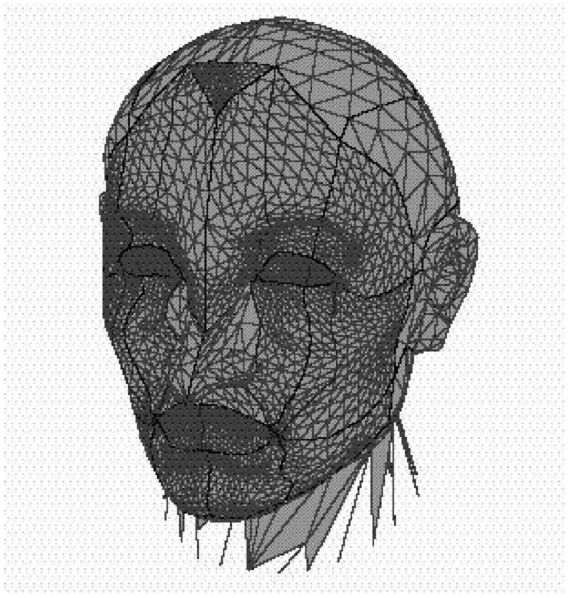
Figure 27: Results [Tak04]

However, there is a major limitation imposed by RIBALD's suboptimal algorithm for detecting mirror symmetry in natural line drawings. Consider the line drawing in Figure 26. Geometric considerations notwithstanding, the mirror symmetry of the drawing is evident to a human, but RIBALD fails to detect it. As a result, although Takahashi's program can produce interesting results from the corresponding template (see Figure 27), this template had to be produced by hand.

6.2 Limitations of Takahashi's Program

Although Takahashi's implementation only processes genus zero templates, there appears no reason in principle why it could not be extended to objects with through holes.

Another limitation is that templates are limited to those produced from line drawings which do not contain T-junctions. The problems here are those of determining how far along the partially-occluded line occlusion takes place and the curvature of the occluded part of the line. These problems, although not trivial, should be soluble, but Takahashi's current implementation does not include solutions.

7. Conclusions

We have suggested a method for interpreting 2D line drawings as 3D curved objects by reference to a template, also 3D but input as a 2D line drawing. We have illustrated this concept with reference to a particular industrial application in which curved surfaces are important, that of car body design.

Extending the idea to other application areas should be straightforward. We also anticipate no difficulties in processing non-genus-zero templates. Drawings with T-junctions may present problems, but these should be soluble.

In conclusion, the idea presented here is straightforward to use, flexible, and although not fully automatic requires less manual intervention than previous approaches.

8. Acknowledgements

Part of this work was funded by financial support from JSPS (Japan Society for the Promotion of Science) Fellowship number P03717.

References

[BT81] H. G. Barrow and J. M. Tenenbaum, *Interpreting Line Drawings as Three-Dimensional Surfaces*, Artificial Intelligence **17**, 75-116, 1981.

[Clo70] M. B. Clowes, *On Seeing Things*, Artificial Intelligence **2**, 79-116, 1970.

[CHZ00] J. M. Cohen, J. F. Hughes and R. C. Zeleznik, *Harold: A World Made of Drawings*, in Proceedings of the First International Symposium on Non Photorealistic Animation

Photorealistic Animation and Rendering (NPAR) for Art and Entertainment, 2000.

[CCG99] J. Conesa Pastor, P. Company Calleja and J. M. Gomis Marti, *Initial Modeling Strategies for Geometrical Reconstruction - Optimization-Based Approaches,* in Proceedings of 11th International Conference on Design Tools and Methods in Industrial Engineering, 161-171, 1999.

[Dij59] E.W. Dijkstrå, *A Note on Two Problems in Connexion with Graphs*, Numerische Mathematik **I**, 269-271, 1959.

[FCG00] E. Ferley, M. P. Cani and J. D. Gascuel, *Practical volumetric sculpting*, The Visual Computer **16**(8), 469-480, 2000.

[Fur93] S. Furushima, S. Kanai and H. Takahashi, *Generation of 3 Dimensional Free-Form Curve Models by Automatic Recognition of Rough Sketches*, J. Japan Society for Precision Engineers, **59**(6), 105-110, 1993. (In Japanese)

[Gri97] I. J. Grimstead, *Interactive Sketch Input of Boundary Representation Solid Models*, PhD Thesis, Cardiff University, 1997.

[Huf71] D. A. Huffman, *Impossible Objects as Nonsense Sentences*, Machine Intelligence **6**, 295-323, New York American Elsevier, 1971.

[IMT99] T. Igarashi, S. Matsuoka, H. Tanaka, *Teddy: A Sketching Interface for 3D Freeform Design*, ACM SIGGRAPH'99 pp.409-416, Los Angeles, 1999.

[Jen92] D. L. Jenkins, *The Automatic Interpretation of Two-Dimensional Freehand Sketches*, PhD Thesis, University of Wales College of Cardiff, 1992.

[Kan90] K. Kanatani, *Group-Theoretical Methods in Image Understanding*, Number 20 in Springer Series in Information Sciences, Springer-Verlag, 1990.

[LS96] H. Lipson and M. Shpitalni, *Optimization-based Reconstruction of a 3D Object from a Single Freehand Line Drawing*, Computer-Aided Design **28** (8), 651-663, 1996.

[Lip98] H. Lipson, *Computer Aided 3D Sketching for Conceptual Design*, PhD Thesis, Technion-Israel Institute for Technology, Haifa, 1998.

[Loo87] C. Loop, *Smooth Subdivision Surfaces Based on Triangles,* MSc Thesis, University of Utah, 1987.

[Mal87] J. Malik, *Interpreting Line Drawings of Curved Objects*, International Journal of Computer Vision **1**, 73-103, 1987.

[Mar91] T. Marill, *Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects*, International Journal of Computer Vision **6**(2), 147-161, 1991.

[Mit99] J. Mitani, *A Study of 3D Sketching used for*

[PT94] P. Parodi and V. Torre, *On the Complexity of Labeling Perspective Projections of Polyhedral Scenes*}, Artificial Intelligence **70**, 239-276, 1994.

[PLVT98] P. Parodi, R. Lancewicki, A. Vijh and J. K. Tsotsos, *Empirically-Derived Estimates of the Complexity of Labeling Line Drawings of Polyhedral Scenes*, Artificial Intelligence **105**, 47-75, 1998.

[Per68] D. N. Perkins, *Cubic Corners*, Quarterly Progress Report 89, 207-214, MIT Research Laboratory of Electronics, 1968.

[Sch99] M. D. Schuresko, *Image-Based Modeling of Curved Objects* http://www-2.cs.cmu.edu/~ph/869/results/schuresko/final_project/report.html

[Sug84] K. Sugihara, *An n log n Algorithm for Determining the Congruity of Polyhedra*, Journal of Computer and Systems Science **29**(1), 36-47, 1984.

[Sug86] K. Sugihara, *Machine Interpretation of Line Drawings*, MIT Press, 1986.

[Tak04] Y. Takahashi, *Constructing 3D from Sketch Portaits according to Prepared Templates*, MSc Thesis, The University of Tokyo, 2004. In Japanese.

[TDM99] O. Tolba, J. Dorsey and L. McMillan, *Sketching with Projective 2D Strokes,* Proceedings of UIST 99. ACM SIGCHI, 1999.

[VM02] T. Varady and R.R. Martin, *Reverse Engineering* in: The Handbook of Computer Aided Design, eds. G. Farin, J. Hoschek, M.-S. Kim, 651-681, Elsevier, 2002.

[VSMM00] P. A. C. Varley, H. Suzuki, J. Mitani and R. R. Martin, *Interpretation of Single Sketch Input for Mesh and Solid Models*, International Journal of Shape Modelling **6**(2), 207-241, 2000.

[Var03] P. A. C. Varley, A*utomatic Creation of Boundary-Representation Models from Single Line Drawings*, PhD Thesis, Cardiff University, 2003.

[VMS04a] P. A. C. Varley, R. R. Martin and H. Suzuki, *Making the Most of Using Depth Reasoning to Label Line Drawings of Engineering Objects*, Proc. 9th ACM Symposium on Solid Modeling and Applications SM'04, 191-202, 2004.

[VMS04b] P. A. C. Varley, R. R. Martin and H. Suzuki, *Can Machines Interpret Line Drawings?* Eurographics Workshop on Sketch Input SBM'04, 2004.

[Wal72] D. M. Waltz, *Generating Semantic Descriptions from Drawings of Scenes with Shadows*, Tech Rept AI-TR-271, M.I.T., Cambridge USA, 1972.