

A System for Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch — Geometric Finishing

P. A. C. Varley, R. R. Martin

Dept. of Computer Science
Cardiff University
Cardiff, Wales, UK
{Peter.Varley, Ralph.Martin}@cs.cf.ac.uk

Abstract

This paper discusses research results useful for producing a system which converts a two-dimensional computer sketch of a single homogeneous polyhedral object into a boundary representation solid model. An overview of our approach to such a system and its main algorithms is presented elsewhere.

This paper gives more detail of some of our methods for producing a finished, geometrically-correct model assuming that previous stages have produced a consistent topology with approximate geometry. It discusses various alternatives which were considered, and explains which were chosen and why. The topics covered in detail are the types of constraint which are generated, optimisation of face normals to meet these constraints, and the special case processing used for particular categories of object.

1. Introduction

This paper discusses research results useful for producing a system which converts a two-dimensional computer sketch of a single homogeneous object into a boundary representation solid model. The utility of such a system has been argued elsewhere [4, 16].

The problem is to convert the sketch into a boundary representation solid model of the most plausible 3D interpretation of the sketch, and to do so in a reasonable time (a second or less on a powerful personal computer is “reasonable”).

Although the theoretical impossibility of perfect conversion of 2D to 3D is both obvious and well-known, there exists an expanding set of objects for which conversion can be achieved in practice. Some assumptions concerning the sketch are required. We assume that the sketch is of a single manifold polyhedral object. Topologically, we assume that

all vertices in the object are trihedral, that no face contains more than one loop, and that the sketched object contains no through holes. Geometrically, we assume that the object is in general position, with no faces lying perpendicular to the view direction. Lastly, we assume that the object has been sketched from the “most informative viewpoint”, i.e. that there is nothing at the rear of the object which could not reasonably be inferred from the visible part of the object.

In terminology, we use *region*, *line* and *junction* to denote atoms of a two-dimensional sketch, and *face*, *edge* and *vertex* to denote atoms of the three-dimensional object which has been sketched. A junction where two lines meet is *bi-connected*; a junction of three lines is *triconnected*.

Several approaches to this problem are possible, and have been summarised elsewhere [16]. Of these, we have chosen Grimstead’s system [3] as our reference point, as this makes the same assumptions as our system and comes closest to achieving our aims. It produces valid 3D interpretations of many classes of valid sketches, it produces the most plausible interpretations for a subset of these, and it is quick enough to be considered interactive. Grimstead acknowledges that not all valid sketches produce correct topology — attempts to address this problem are described elsewhere [18].

We also make use of symmetries and regularities implied by the sketch in producing a more plausible topology. We use the terminology that rotation and reflection are *symmetry operations*. Rotation axes and mirror planes are *symmetry elements*. Other artefacts which give clues to the structure of an object, such as parallelism or extrusion, which do not meet the strict definition of symmetry, are *regularities*.

Elsewhere [16], we have described a method which improves on Grimstead’s system by interpreting a larger subset of valid sketches and by producing more plausible topological completion and geometric details of the resulting object. Briefly, this process can be subdivided into three main

steps: preliminary processing (including frontal geometry and sketch categorisation), topological reconstruction and geometrical finishing.

The aim of preliminary processing [17] is to deduce as much information as possible concerning the sketch, without making changes or additions: lines and junctions are labelled, candidate parallel line pairs are identified, a preliminary *frontal geometry* (the 3D geometry of the part of the object visible in the sketch) is produced, potential local symmetries are identified, and an attempt is made to categorise the object portrayed in the sketch as one or more of a number of cases which simplify reconstruction.

Symmetry artefacts and categorisations are allocated *figures of merit*. Figures of merit are justified and described in more detail in [17], which in particular defines how to calculate the figures of merit for two lines A and B being parallel ($F(A \parallel B)$) or perpendicular ($F(A \perp B)$); numerically, figures of merit range from 1 (certain) to 0 (impossible), and may be combined by *multiplication*, which reduces the merit, or *reinforcement*, which increases the merit.

The aim of topological reconstruction [18] is to produce a complete topology for the object. From the frontal geometry, the list of symmetries and regularities, and the categorisation, various hypotheses are made concerning the topology of the hidden parts using an iterative process. Each step chooses the best such hypothesis according to an estimate of merit, and adds the corresponding hidden part to the object.

By this stage, as well as a complete topology, the reconstruction has a preliminary geometry: the x and y coordinates of visible vertices come from the original sketch, and are reasonably trustworthy without being absolutely accurate; the z (depth) coordinates of visible vertices, and all coordinates of hidden vertices, are unreliable estimates.

Geometric finishing (described in this paper) attempts to produce the best geometrical interpretation of the recovered topology given the list of symmetries and regularities — “finishing” is to be understood as indicating both a final stage of processing and the means whereby an aesthetically appealing object is produced. Identification of a symmetry element or regularity in an object produces one or more *constraints*, which limit the possible positions of the vertices, edges and faces. The system attempts to satisfy as many of these constraints as possible, in descending order of merit.

The rest of this paper gives technical details of how certain key alternatives for finalising the geometry were chosen, and also includes progress since [16]. It is subdivided into (a) general-case finishing, a method which can process any sketch but which is sometimes unacceptably slow, and, while often producing results useful in practice, can not be justified theoretically and cannot therefore be claimed to be robust; and (b) special-case finishing for particular categories of object, methods which are quicker and more robust but limited in application. We conclude by outlining plans

for future work in geometric finishing.

2 General-Case Geometrical Finishing

This stage of processing takes a topologically-correct object with provisional vertex positions and a group of previously-identified symmetry and regularity artefacts, and aims to produce geometric information — specifically, face equations and vertex positions — which “achieves as much merit” as possible.

The stage can be subdivided into five substages: preliminary estimation of face normals; generation of constraints from symmetries and regularities; enforcing constraints on face normals; optimising face distances while enforcing constraints; and intersecting faces to obtain final edge geometry and vertex positions. The final step is straightforward (the x , y and z coordinates of each vertex are recalculated from the equations of the three faces on which it lies); we describe in detail the other stages.

2.1 Preliminary Estimates of Face Normals

Later, we generate various constraints, estimate their merit numerically, and attempt to optimise the geometry so as to maximise the overall merit. We require a preliminary geometry (a) as a basis for computing the numerical estimates of merit of our constraints, which depends upon how well they match the preliminary geometry, and (b) as a starting-point for the iterative optimisation process which determines the final geometry. A good initial estimate will both lead to a quicker solution and increase the likelihood of finding the correct global solution.

Vertex coordinates generated by the topological completion stage are used to provide preliminary estimates of face normals. For all but triangular faces (which can be solved directly) we use a least-squares linear system [1] with weightings which give priority to visible vertices. Adopting the convention that face normals are directed to look towards the face from outside the object, normals of visible (including partially-occluded) faces have positive z -components, and normals of hidden faces usually have negative z -components. The latter is not always correct, as in complex objects it is possible to have hidden faces oriented towards the viewer but occluded by nearer faces. However, assuming this always to be true leads to simplicity, and introduces no processing errors — the difference between \hat{N} and $-\hat{N}$ is ignored except for its use in rotations, and rotation axes are identified at a processing stage [17] which only considers fully-visible faces).

It is possible to attempt to refine these normal direction estimates using skewed symmetry [5]. We have found no benefit in doing this. Even without using skewed symmetry, the estimates are accurate enough to be used as input to

the remaining stages of the process. The effect of improving them would be to reduce the time taken by the iterative optimisation. Since skewed symmetry generally improves the normal estimates for well-drawn sketches but can actually make them worse for poorly-drawn sketches, the expected effect of incorporating it would be to make the worst (and thus slowest) cases take longer, which is not helpful. In practice, the resulting time differences are negligible.

2.2 Constraints

Artefacts inferred from the sketch [17] such as symmetry elements and regularities imply various relationships between the faces of the final object. We generate constraints based on these relationships. Constraints are allocated a figure of merit [17] (identical constraints are merged, the merit being *reinforced*). The method attempts to satisfy constraints in descending order of merit, with constraints being accepted if there are enough degrees of freedom left in the object to accommodate them, or if they agree with the geometry produced by accepted constraints.

We first consider the types of constraint which we wish to enforce, and then the symmetry elements and regularities which produce them.

2.2.1 Constraint Types

Originally, five types of constraint were identified:

A parallelism constraint requires two faces M and N to be parallel (i.e. their normals \hat{M} and \hat{N} are parallel, $\hat{M} = \pm\hat{N}$).

A two-way perpendicularity constraint requires two faces M and N to be perpendicular.

A three-way perpendicularity constraint requires three faces M , N and O to be mutually perpendicular.

A rotation constraint requires three faces M , N and R to be related such that a rotation through an angle ρ about a perpendicular axis through the centre of R moves N to the position occupied by M . (Constraints for 180° rotations about edge centres and 120° rotations about vertices are a straightforward extension, but have not been implemented at the time of writing.)

A mirror constraint requires two faces M and N to be related via a mirror chain of lines C such that reflection through the mirror plane defined by C moves M to the position occupied by N and vice versa. Mirror chains, identified from the sketch [17], are contiguous sequences of lines of regional mirror symmetry; for any mirror chain C all of the faces C_1, C_2, \dots crossed by the lines must have their face normals in the mirror plane.

It was found in practice, when just using the above constraint types, that very implausible perpendicularity constraints were enforced between faces which were clearly not

perpendicular in the sketch or the preliminary frontal geometry, because after the “correct” constraints were satisfied, enough degrees of freedom remained for one more constraint to be accepted. Rather than introduce an arbitrary numerical merit threshold below which all constraints are rejected, we prefer to introduce a sixth constraint type, an angle constraint, which requires two faces M and N to have a fixed “common” angle ρ between them. We currently restrict ρ to be 30° , 45° or 60° , or other angles whose tangent is the ratio of integers each in the range 1–6. The latter cases arise commonly in the sketches in [21], and in semi-axis-aligned wedges, a common design feature according to [14]. (It can be noted that in semi-axis-aligned objects, if any face A is at 120° to another face B , it is likely to be at 60° to a face B' parallel to B , and this will constrain its orientation. Similarly, constraints to fix interfacial angles of 135° and 150° are not required either.)

For the purposes of constraint processing, mirror planes are treated as faces: they too have normals and distances.

2.2.2 Constraints from Mirror Symmetry

For each mirror chain (C_i) in the object, we identify the pairs of distinct faces which are reflected into one another by the mirror chain (N_{ij}, M_{ij}) and generate a mirror constraint linking $N = N_{ij}, M = M_{ij}, C = C_i$.

The base figure of merit for a mirror constraint is $(P_c^{2/n_c}) \times H_c^2$, where P_c is the figure of merit of the mirror chain estimated when the chain was identified [17], n_c is the number of faces in the mirror chain, and H_c is the proportion of faces in the object paired by the mirroring operation (locally-effective mirror chains can be useful, but are not as reliable or important as globally-effective mirror chains). This is multiplied by 0.8 for single-face mirror chains which terminate at a vertex (those terminating at edge mid-points are more reliable). If the mirror chain is the principal one for an object categorised as “a semi-axis-aligned sketch with mirror chain” [17], the figure of merit is reinforced by that for the categorisation.

The merit figure for each constraint is multiplied by a measure of how well it fits the preliminary estimates: the figure of merit for parallelism between normal \hat{M} and the vector obtained by reflecting normal \hat{N} through the mirror plane.

Additionally, for each mirror chain (C_i), any face which the mirror chain reflects to itself ($M_{ij} = N_{ij}$) (this includes all faces in the mirror chain (C_{ij}) and possibly others) must have a normal in the mirror plane, i.e. perpendicular to the mirror chain normal. A perpendicularity constraint is generated between each such face and the mirror plane normal.

2.2.3 Constraints from Rotational Symmetry

For each face (R_i) in the object containing an axis of rotation, we identify the face (M_{ij}) to whose position each face (N_{ij}) is rotated and (providing $M_{ij} \neq N_{ij}$) generate a rotation constraint linking $R = R_i$, $N = N_{ij}$, $M = M_{ij}$. Faces unchanged by the rotation ($M_{ij} = N_{ij}$) are perpendicular to and centred on the axis of rotation, so a parallelism constraint linking N_{ij} and R_i is generated instead.

The base figure of merit for a rotation constraint is $K_c \times P_c \times H_c^2$, where K_c is 0.8 for C_2 , 0.85 for C_3 , 0.9 for C_4 , 0.95 for C_5 and 1.0 for C_6 , giving some encouragement to higher-order symmetry, P_c is the figure of merit for the rotational symmetry axis estimated when the axis was identified [17], and H_c is the proportion of faces in the object paired by the rotation operation.

The merit figure for each constraint is multiplied by a measure of how well it fits the preliminary estimates: the figure of merit for parallelism between normal \hat{M} and the vector obtained by rotating normal \hat{N} around the rotation axis.

2.2.4 Constraints from Expected Axis Alignment

In analysing the sketch [17], lines in the sketch are allocated to bundles, in each of which all lines are nearly parallel and expected (rightly or wrongly) to correspond to edges which are parallel in 3D. Three of the bundles are “special”, in that two of them (labelled B_0 and B_1) are believed to correspond to lines in a plane parallel with the base of the object, and the third (V) is believed to define a vertical axis perpendicular to the base plane. In axis-aligned sketches, we assume that the base bundles are mutually perpendicular, and in semi-axis-aligned sketches, we attempt to identify the two bundles which form the two non-vertical axes. Each face is classified as “vertical”, “horizontal” or other, according to whether there are edges in the face allocated to the vertical bundle, both of the base bundles, or otherwise.

Parallelism constraints are generated between each pair of faces classified as horizontal. Perpendicularity constraints are generated between each vertical and each horizontal face. The base merit figure for each constraint is 0.97 (the assumption that the object rises vertically from a flat base is generally a good one, but it is not a certainty); it is multiplied by a measure of how well it fits the preliminary estimate.

For each face, the bundles to which its edges have been allocated are tabulated. Any pair of faces with two or more bundles of edges in common are parallel if bundling is successful, so a parallelism constraint is generated. The base merit figure for each constraint is 0.99 (very occasionally, bundling is misleading, as an example in [17] shows); it is multiplied by a measure of how well it fits the preliminary estimate.

For each vertex which could be a “cubic corner” [12],

one three-way perpendicularity constraint and three two-way perpendicularity constraints are generated. The base merit figure for each constraint is that for the object being axis-aligned or semi-axis-aligned; it is multiplied by a measure of how well it fits the preliminary estimates, using the figures of merit described above.

In some cases, it can be deduced that a face must be perpendicular to a particular edge. The face normal can then be allocated to the bundle of the line corresponding to this edge. This is always the case with axis-aligned sketches, and often the case with many of the faces in semi-axis-aligned sketches. For each pair of faces with normals successfully allocated to the same bundle, a parallelism constraint is generated. The base figure of merit for this is 1; it is multiplied by a measure of how well it fits the preliminary estimates: the figure of merit for parallelism between the two face normals.

(The above method generates $n(n - 1)/2$ constraints where there are n faces with normals bundled together. An alternative, which would generate fewer constraints but which may be less robust, is to define a “desired direction” for each bundle, and generate n constraints requiring each face in turn to be parallel to the desired direction. This would be somewhat quicker if all constraints are accepted, but not greatly quicker, since in the case with $n(n - 1)/2$ constraints, the later constraints can be deduced to be true using logical reasoning (see section 2.3.1). More importantly, if some face normals cannot be made parallel to the desired direction, but could be made parallel to one another, our proposed method can enforce this, whereas the suggested alternative cannot.)

2.2.5 Constraints from Preliminary Estimates

The angle between each pair of faces in the preliminary estimate is considered. Either a parallelism or a perpendicularity constraint is generated, and in addition an angle constraint is generated for the nearest “common angle” to the estimated angle, as defined earlier. The base merit figure for each constraint is 1.0 for the parallelism and perpendicularity constraints, and 0.9 for the other angle constraints; it is multiplied by a measure of how well it fits the preliminary estimate.

2.3 Face Normal Constraint Enforcement

Constraints affecting normals are enforced in descending order of merit until there is no more freedom left to adjust further normals. We initially used the naive algorithm:

- the constraint with the highest figure of merit is always accepted

- for each other constraint, in descending order of merit:
 - an attempt is made to adjust the existing face normals to accommodate the new constraint as well as all previous accepted constraints; if this succeeds, the new face normals are stored and the constraint is accepted; otherwise, the previous face normals are restored, and the constraint is rejected;

As the numerical processing required is considerable, this is slow. The more sophisticated algorithm below is designed to bypass the numerical processing stage as often as possible, using logical reasoning. Significant work is only done for constraints requiring a numerical computation, which except in extreme cases is (much) smaller than the total number of constraints.

- the constraint with the highest figure of merit is enforced first
- for each other constraint, in descending order of merit:
 - if logical reasoning (see Section 2.3.1) can show that the constraint is necessarily valid, it is accepted and enforced;
 - if logical reasoning can show that the constraint is necessarily invalid, it is discarded;
 - if the constraint is already satisfied numerically by the object, it is accepted;
 - if enough angular degrees of freedom remain in the constrained faces, the constraint is accepted (see Section 2.3.2);
 - if the object has degrees of freedom elsewhere, an attempt is made to adjust the existing face normals to accommodate the new constraint as well as all previous accepted constraints (see Section 2.3.3); if this succeeds, the new face normals are stored and the constraint is accepted; otherwise, the previous face normals are recovered and the constraint is discarded;
 - otherwise, the constraint is discarded.

As an example, the semi-axis-aligned sketch in Figure 1 generates 302 constraints; of these, only 80 require numerical processing if the more sophisticated algorithm is used. For Figure 2 the corresponding figures are 32 out of 121 constraints, and for Figure 3 it is 29 out of 149 constraints. The resulting speed improvement is about a factor of four or five in these particular cases.

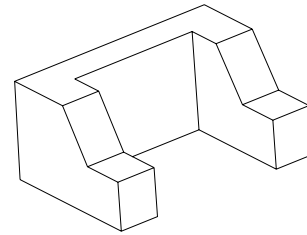


Figure 1. Grimstead's Bracket from [3]

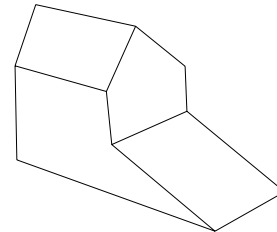


Figure 2. Sketch from [21]

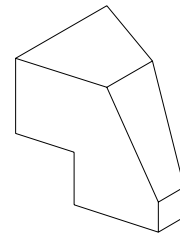


Figure 3. Another sketch from [21]

2.3.1 Logical Reasoning

As just noted, a considerable improvement in performance is obtained if we can deduce whenever two faces are of necessity either parallel or perpendicular to one another — the performance improvement is particularly significant for objects which are axis-aligned or semi-axis-aligned (one survey [14] shows that a large majority of engineering objects are). In many cases, parallelism and perpendicularity cases can be accepted or rejected without any numerical processing, and in some cases mirror constraints and rotation constraints (particularly C_2 and C_4) can also be accepted or rejected immediately.

To carry out this logical process, two faces are considered to be in one of three logically-related states: they are either parallel, perpendicular, or at some other angle. A set of the three possible relationships is stored for each pair of faces. Initially, all three relationships are possible, except that faces sharing an edge cannot be parallel, and if the object is axis-aligned no pair of faces can be at some other angle. Accepting a constraint of a given type (e.g. M and N parallel) narrows down the relationship (e.g. to parallel). As processing continues, the remaining states may enable

us to deduce whether a given relationship is necessary (the only remaining state) or invalid (not in the set of remaining states).

Extra rules can be used to restrict the set of states further, for example:

- two faces are necessarily parallel if they might be parallel and a third face is known to be parallel to both of them
- two faces are necessarily parallel if they might be parallel and both are perpendicular to two other mutually-non-parallel faces
- two faces are necessarily perpendicular if they might be perpendicular and a third face is known to be parallel to one and perpendicular to the other.
- if a constraint is accepted requiring two faces M and N to be parallel, then any third face R which is known to reflect M into N and vice versa must be either parallel or perpendicular to M and N

2.3.2 Angular Degrees of Freedom

The upper limit for the number of angular degrees of freedom (DoF) of a face can be determined from the relationship sets used for logical reasoning:

- face 0, chosen arbitrarily, has no degrees of freedom — it is used as a reference datum;
- face 1, chosen arbitrarily from those faces sharing an edge with face 0, has at most one degree of freedom — this prevents the object spinning around the normal to face 0;
- other faces have at most two degrees of freedom;
- any face which is parallel to a lower-numbered face has no degrees of freedom
- any face which is perpendicular to two lower-numbered faces which are not parallel to one another has no degrees of freedom
- any other face which is perpendicular to a lower-numbered face has at most one degree of freedom

To obtain the actual number of degrees of freedom remaining we must now subtract those used by accepted mirror, rotation and angular constraints. To this end, we store each accepted mirror, rotation or angular constraint. In estimating the number of degrees of freedom remaining at a face, we attempt to subtract the degrees of freedom used up by the stored constraints.

We can define the number of degrees of freedom required for these constraints, but we know of no satisfactory method for allocating semi-local constraints (e.g. a rotation constraint reduces the total number of degrees of freedom of three faces by two) to particular faces. This is more difficult to ascertain because of the degeneracy problem — different combinations of constraints may reduce to the same information, and existing constraints may turn out to be equivalent if a later constraint is accepted. Methods of calculating the number of degrees of freedom after satisfying a number of geometric constraints have been the subject of several studies. Sugihara [15] lists a number of these. More recently, Kramer [6] has shown the difficulty of allowing for geometrical coincidences in two dimensions. Latham [7] has methods for resolving this issue which appear to improve on existing algorithms, but these are based on symbolic reasoning (using formal languages which allow potentially unbounded sets of possibilities) and are therefore unsuitable for current interactive systems. Owen's [11] algorithm and its implementation are fast enough, but are restricted to 2D and cannot guarantee to find a solution for underconstrained systems. Whitely's [20] extension to 3D of a method which is successful in 2D is unsatisfactory in that it is intolerant of sketching inaccuracies and that it is limited to triangular and quadrilateral faces.

Rather than attempt to solve the degree of freedom problem exactly, we recognise that since perfect conversion of sketches to objects is formally impossible, an algorithm which gives reasonable results in practice is acceptable. In the spirit of using "heuristic search" methods (such as genetic algorithms and simulated annealing) to *hard* problems, we try several times to guess an allocation to specific faces of the degrees of freedom required by each constraint, and if any of the guesses leaves the face in question with enough degrees of freedom to accommodate the constraint under consideration, the constraint is accepted. Although practically effective, this algorithm cannot be justified theoretically:

- repeat a number of times (we currently use 6, an arbitrary small integer)
 - initialise all DoF values to the ones given by the logical data
 - for each constraint already considered and accepted
 - * allocate the DoF required for this accepted constraint, choosing at random from those faces affected by the constraint with remaining DoF
 - if this gives more DoF in the faces we are currently trying to constrain than any other try so far, note this as the candidate best solution

- return the number of DoF in the best solution

The method we use is known to be flawed (in addition to its lack of theoretical rigour) — it is possible that, when a new constraint produces a re-interpretation of existing constraints (e.g., accepting a perpendicularity constraint may allow a previously-accepted rotation constraint to be expressed in terms of parallelism and perpendicularity), the number of degrees of freedom around a face may increase, and a lower-merit constraint could thus be accepted after a high-merit constraint was rejected. However, this is rare, and no other ideas we tried came close to producing an acceptable allocation of angular degrees of freedom.

The test used to decide whether the object as a whole has degrees of freedom left is equally fallible, for the reasons noted above:

- the object has remaining degrees of freedom if, using the above method, any face has remaining degrees of freedom

2.3.3 Adjusting Normals

In the absence of enough angular degrees of freedom, we test whether the object can be adjusted to enforce a new constraint as well as all previously-accepted ones by trying find a numerical solution which accommodates them all. We use an iterative process which terminates either when the objective function (see Section 2.3.4) goes below a threshold value (success) or when a specified maximum number of iterations has been exceeded (failure). Before starting the process, we determine which face normals can move:

A face orientation is fixed (not free to move) if its face normal can be calculated from known information:

- it is face 0 (an arbitrarily-chosen fixed face)
- it is face 1 and the angle between it and face 0 is known
- it is parallel to another fixed face
- it is perpendicular to two other non-parallel fixed faces
- a rotation constraint has been accepted tying it to two other fixed faces
- a mirror constraint has been accepted tying it to two other fixed faces
- it is perpendicular to one fixed face and an angular constraint ties it to another fixed face

On each iteration $i + 1$, we adjust the face normal $\hat{\mathbf{N}}$ of each face N which can move to try to meet the accepted constraints and the constraint under consideration, and evaluate the objective function. We calculate estimates of $\hat{\mathbf{N}}_{i+1}$

based on each constraint and the values of other normals calculated in iteration i , and use a weighted average for the overall estimate of $\hat{\mathbf{N}}_{i+1}$, the weights being the figures of merit for each constraint divided by the number of faces affected by that constraint which can still move.

- A parallelism constraint between faces M and N predicts a new value $\hat{\mathbf{M}}_{i+1} = \pm\hat{\mathbf{N}}_i$ and vice versa (the estimate nearer $\hat{\mathbf{M}}_i$ is chosen).
- A perpendicularity constraint between faces M and N predicts a new value $\hat{\mathbf{M}}_{i+1} = \pm\text{normalise}((\hat{\mathbf{N}}_i \times \hat{\mathbf{M}}_i) \times \hat{\mathbf{N}}_i)$ and vice versa.
- A mirror constraint between faces M and N and mirror chain C predicts as a new value for $\hat{\mathbf{M}}$ the vector obtained by reflecting $\hat{\mathbf{N}}$ through the mirror plane: $\hat{\mathbf{M}}_{i+1} = \pm(\hat{\mathbf{N}}_i - 2(\hat{\mathbf{N}}_i \cdot \hat{\mathbf{C}}_i)\hat{\mathbf{C}}_i)$, where $\hat{\mathbf{C}}$ is the mirror plane normal.

The predicted new value of the normal for any face A in the mirror chain is found as $\hat{\mathbf{A}}_{i+1} = \pm\text{normalise}((\hat{\mathbf{C}}_i \times \hat{\mathbf{A}}_i) \times \hat{\mathbf{C}}_i)$ as described above.

- For a rotation constraint, rotating face N an angle ρ about a normal $\hat{\mathbf{R}}$ through the centre of face R to obtain face M , if $\hat{\mathbf{R}}$ is known, $\hat{\mathbf{M}}$ and $\hat{\mathbf{N}}$ can be estimated using standard geometry: $\hat{\mathbf{M}}_{i+1} = \pm\mathcal{R}(\rho, \hat{\mathbf{R}})\hat{\mathbf{N}}_i$ where \mathcal{R} is the rotation matrix for rotating through an angle ρ about $\hat{\mathbf{R}}$.

The method for estimating $\hat{\mathbf{R}}$ from $\hat{\mathbf{M}}$ and $\hat{\mathbf{N}}$ is less well-known. We derive this by considering R , M and N as points on the Gaussian sphere, and adding point D , a point on the sphere mid-way between M and N , and G , a point 90° from D around a great circle including R and D . If the angle σ between \vec{OD} and \vec{OR} can be found, then clearly $\hat{\mathbf{R}} = \hat{\mathbf{G}} \sin \sigma \pm \hat{\mathbf{D}} \cos \sigma$ (rotation about either axis will move N to M).

To find σ , we take the cosine rule for spherical triangles [8, 19]: given points A , B and C on the surface of a sphere centred on the origin, and angles \hat{A} , \hat{B} and \hat{C} being the angles between the planes meeting at those points, and vectors $\hat{\mathbf{l}}$, $\hat{\mathbf{m}}$ and $\hat{\mathbf{n}}$ their respective position vectors, we can define angles α , β and γ so that $\cos(\alpha) = \hat{\mathbf{m}} \cdot \hat{\mathbf{n}}$, $\cos(\beta) = \hat{\mathbf{n}} \cdot \hat{\mathbf{l}}$, $\cos(\gamma) = \hat{\mathbf{l}} \cdot \hat{\mathbf{m}}$ and obtain the expression $\cos(\alpha) = \cos(\beta) \cos(\gamma) + \sin(\beta) \sin(\gamma) \cos(\hat{A})$.

By construction, the planes ORD and OMN (which includes D) are perpendicular, so by taking the spherical triangle RDM we obtain

$$\cos(\theta) = \cos(\sigma) \cos(\delta), \text{ where } \cos \theta = \hat{\mathbf{R}} \cdot \hat{\mathbf{M}}, \cos \delta = \hat{\mathbf{M}} \cdot \hat{\mathbf{D}}, \text{ and } \cos \sigma = \hat{\mathbf{R}} \cdot \hat{\mathbf{D}}.$$

It can be shown by a second application of the spherical triangle rule that $\cos \phi = 1 + \sin^2 \theta (\cos \rho - 1)$ where: ϕ

is the angle between face normals $\hat{\mathbf{M}}$ and $\hat{\mathbf{N}}$, so $\cos \phi = \hat{\mathbf{M}} \cdot \hat{\mathbf{N}}$, and θ is the angle between face normals $\hat{\mathbf{R}}$ and $\hat{\mathbf{M}}$, so $\cos \theta = \hat{\mathbf{R}} \cdot \hat{\mathbf{M}} = \hat{\mathbf{R}} \cdot \hat{\mathbf{N}}$.

Rearranging this result, we obtain: $\sin \theta = \sqrt{(\cos \phi - 1)/(\cos \rho - 1)}$.

By construction, $\delta = \phi/2$.

Combining results, we obtain

$\cos^2 \sigma = 2(\cos \rho - \cos \phi)/((\cos \phi - 1)(\cos \rho - 1))$
from which $\cos \sigma$ and $\sin \sigma$ are easily obtained.

We therefore estimate $\hat{\mathbf{R}}_{i+1}$ as follows:

- Set $\hat{\mathbf{D}} = \text{normalise}(\hat{\mathbf{M}}_i + \hat{\mathbf{N}}_i)$;
- Set $\hat{\mathbf{G}} = \text{normalise}(\hat{\mathbf{M}}_i \times \hat{\mathbf{D}})$;
- Set $\cos \phi = \hat{\mathbf{M}}_i \cdot \hat{\mathbf{N}}_i$;
- Set $\cos^2 \sigma = 2(\cos \rho - \cos \phi)/((\cos \phi - 1)(\cos \rho - 1))$;
- Estimate the two values $\hat{\mathbf{R}}_{i+1} = \hat{\mathbf{G}} \sin \sigma \pm \hat{\mathbf{D}} \cos \sigma$;
- Choose the nearer estimate of $\hat{\mathbf{R}}_{i+1}$ to $\hat{\mathbf{R}}_i$

Special-case code is needed where ρ is 180° since $(\hat{\mathbf{M}}_i + \hat{\mathbf{N}}_i)$ may be zero, in which event the algorithm above breaks down (and our estimate of $\hat{\mathbf{R}}_{i+1}$ is the nearest vector to $\hat{\mathbf{R}}_i$ perpendicular to $\hat{\mathbf{N}}_i$). We also use special-case code to speed up the calculation where ρ is 90° or 180° .

- An angular constraint predicts as the estimate for $\hat{\mathbf{M}}_{i+1}$ the vector in the plane of $\hat{\mathbf{M}}$ and $\hat{\mathbf{N}}$ which is at an angle ρ from $\hat{\mathbf{N}}$, i.e.
 $\hat{\mathbf{M}}_{i+1} = \hat{\mathbf{N}}_i \cos \rho + \text{normalise}((\hat{\mathbf{N}}_i \times \hat{\mathbf{M}}_i) \times \hat{\mathbf{N}}_i) \sin \rho$.

2.3.4 Objective Function

The objective function is the measure of how well constraints are met by a set of face normals, and is the function being minimised by the optimisation process. A value of zero indicates that all constraints are met perfectly. It is computed as the numerical sum of an appropriate term for each constraint under consideration, as listed here. Figures of merit used in the calculation are defined in [17].

- the term for a parallel constraint is $1 - F(M \parallel N)$
- the term for a perpendicularity constraint is $1 - F(M \perp N)$
- the term for a mirror constraint is $1 - F(M \parallel M'(N, C))$, where $M'(N, C)$ is face M repositioned using the partial estimated normal based on N and C as derived above

- the term for a rotation constraint is $1 - F(M \parallel M'(R, N, \rho))$, where $M'(R, N, \rho)$ is face M repositioned using the partial estimated normal based on N , R and ρ as derived above
- the term for an angular constraint is $1 - F(M \parallel M'(N, \rho))$, where $M'(N, \rho)$ is face M repositioned using the partial estimated normal based on N and ρ as derived above

2.3.5 Problems and Alternatives

Choosing the maximum number of iterations to apply when trying to adjust the geometry to satisfy a constraint is not simple — if it is too low, valid constraints can be rejected, and if too high, speed is affected as each unsatisfiable constraint takes this number of iterations to discard. We use 1000 iterations as the maximum; this is not quite free of either problem but is a reasonable compromise.

The threshold used for success in the objective function is arbitrary too. Too low a value might result in valid constraints being rejected through accumulation of numerical errors, and too high a value might allow unsatisfiable constraints to be accepted. Also, a lower value produces a more accurate geometry at the cost of increasing processing time. We use $1/1000$.

In principle, either of these constants could be tuned to meet user preferences.

Apart from speed, the other problem with this method is that there is no guarantee that the iteration is working towards, rather than away from, the optimal solution. The objective function is used solely to identify a successful terminating condition of the optimisation, and does not influence the way normals are adjusted by guiding the process “downhill”. The process of adjusting normals is separate, and could be taking the object away from the optimum geometry (in principle, it could even be oscillatory, although we have not observed this in practice). The decoupling of the adjustment mechanism and the objective function also has speed implications, in that although we can accept a constraint immediately if the objective function is below a threshold, if the objective function remains above a threshold there is no terminating condition other than exceeding the maximum number of iterations — we have no way of telling (for example) whether the adjustment mechanism is leaving a local minimum to find a better minimum elsewhere, or travelling “uphill” through trying to enforce incompatible constraints.

As an alternative, we have tried using an established non-specific optimisation algorithm, the downhill simplex method *amoeba* [10, 13]. This takes as its parameters a set of n independent real variables and an objective function of those variables, and terminates when a minimum is reached or a maximum number of iterations is exceeded (no threshold need be specified). We use the objective function de-

scribed above, but the algorithm itself decides how the normals should be adjusted. There are problems with this idea, in that it is not easy to express face normals naturally as n independent variables. Also, although *amoeba* always moves downhill, it can become trapped in a local minimum. We found that in practice, some constraints were rejected because of the problem of local minima, and in some but not all cases this affected the resulting geometry (sometimes a later constraint, expressing the same geometrical relationship in a different way, would be accepted). We found *amoeba* to be even slower than the geometry-specific optimisation described above, and although it remains a possibility to which we might return, it is not our current recommendation.

In both our current system and the alternative using *amoeba*, we have tried using skewed symmetry [5] to produce a better initial estimate of face normals, in the hope that this would help the optimisation process to converge more quickly. From Kanade [5], skewed symmetry is a method for calculating the normal of a face given two line (axes) on the corresponding region which are believed to be perpendicular on the face in the 3D world. Given α and β , the angles of the two axes, chosen such that the angle between them is obtuse, it can be shown that the face normal $[P, Q, R]$ is given by $P/R = \rho \cos \lambda$ and $Q/R = \rho \sin \lambda$ where $\lambda = (\alpha + \beta)/2$ and $\rho = \sqrt{-\cos(\alpha - \beta) / \cos(\alpha + \beta)}$. The angles α and β are already available for any face believed to have mirror symmetry, having been calculated as part of the process of identifying the symmetry [17]. However, the performance improvement, if any, was not noticeable and skewed symmetry is not used in our current system.

As a different extension, we used a version of *amoeba* incorporating simulated annealing [13]. It was not significantly quicker and was observed to reject valid constraints as early, high-entropy stages of the annealing process took the geometry away from its best fit. This combines the worst features of the two methods described above, and this approach was also rejected.

We also tested an alternative approach to the problem of estimating the axis of rotation given the start and end positions of a face normal and the rotation angle, using a method based on quaternions. We define quaternions \mathbf{m} and \mathbf{n} to represent the face normals $\hat{\mathbf{M}}$ and $\hat{\mathbf{N}}$, and \mathbf{r} to represent a rotation ρ about the rotation axis $\hat{\mathbf{R}}$. Since $\mathbf{m} = \mathbf{r} \cdot \mathbf{n} \cdot \mathbf{r}^{-1}$, we can estimate \mathbf{r}_{i+1} as $\mathbf{m}_i \cdot \mathbf{r}_i \cdot \mathbf{n}_i^{-1}$. The estimated quaternion \mathbf{r} represents an angle (which is discarded) and a vector, our new estimate of $\hat{\mathbf{R}}$.

This approach could be more robust, in that it does not require choosing the nearer of two vectors to the starting value of face normal $\hat{\mathbf{R}}$, and it could also be marginally quicker for the same reason (though this would depend on the respective implementations). The deciding factor is the accuracy of the predicted value of $\hat{\mathbf{R}}_{i+1}$. We have tested cases

where $\hat{\mathbf{N}}$ and $\hat{\mathbf{M}}$ are accurate and both perpendicular to the true $\hat{\mathbf{R}}$, and $\hat{\mathbf{R}}_i$ is inaccurate by a predetermined angle in the range 5° to 30° , and measured the resulting inaccuracy in $\hat{\mathbf{R}}_{i+1}$ predicted using the two methods. The geometric approach invariably gave correct estimates (to 6 significant figures), while the quaternion approach gave inaccurate estimates, with the output error sometimes being as large as the input error. On the basis of these results, we prefer the geometric approach.

As noted elsewhere [16, 17], using a single symmetry element (such as a mirror chain) to derive geometry directly is particularly sensitive to sketching inaccuracies. It is also inappropriate to many sketches, either because they contain no symmetry element or because they contain several. This approach was also rejected.

2.4 Distances

Once the face normals have been fixed, we can produce a unique geometry by allocating distances (from the origin) for each face. This can be viewed as an n -dimensional iterative optimisation problem where the n unknowns are the face distances and the objective function is a measure of how well the coordinates of points at which faces intersect match the coordinates of vertices drawn in the original sketch.

Our objective function is chosen to spread changes from the sketch evenly throughout the solid object, an approach recommended in [3].

- For each vertex V_j , intersect the three faces it lies on (each defined by a face normal $\hat{\mathbf{N}}$ and an iteratively-changing distance D_i , i being the iteration counter) to obtain vertex coordinates x_{ji}, y_{ji}, z_{ji} ;
- Calculate the sum $\sum_{j=1}^n (x_{ji} - x_{js})^2 + (y_{ji} - y_{js})^2$ (where the original sketch places vertex V_j at (x_{js}, y_{js})) and return this as the objective function value;

It can be noted that if one part of the object is particularly badly drawn (such as the misplaced vertex A in Figure 4), this objective function will effectively hide the error by spreading it evenly through the object rather than correct the error locally. This is believed [3] to be less common in practice than sketches where all junctions are close to, but not precisely at, their proper positions. (Also, the consequences of choosing to move the wrong vertex to “fix” the error are undesirable.)

The number of variables n can sometimes be reduced after consideration of symmetry. We base this on a subset of the constraints identified above:

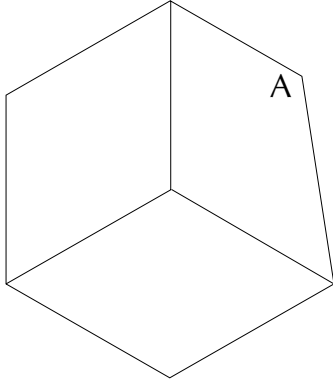


Figure 4. Misplaced Vertex

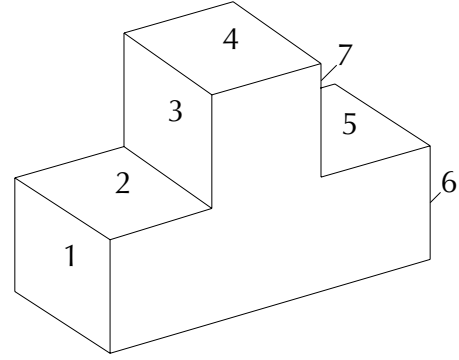


Figure 5. T Block

- parallelism, perpendicularity and angularity constraints have no effect on distances and are not used in distance optimisation;
- any constraint rejected during the process of adjusting face normals is discarded;
- any mirror or rotation constraint which when propagated through the object pairs a convex edge with a concave edge is geometrically incorrect and is discarded;
- of the remaining constraints, if the mirror plane or rotation axis constrained is perpendicular to the two faces M and N , and M and N are parallel, then they must also be coplanar: the distance for N is then dropped from the optimisation (the number of variables is decremented) and set equal to that for M on each iteration (see faces 2 and 5 in Figure 5)
- of the remaining constraints, if faces M and N are parallel, and two other paired faces M' and N' are also parallel to one another and to M , then the distances obey the equation $D_M - D_{M'} = D_{N'} - D_N$. The distance for N' is then dropped from the optimisation (the number of variables is again decremented) and $D_{N'}$ is set equal to $D_M + D_N - D_{M'}$ on each iteration (see faces 1 and 3, and 6 and 7, in Figure 5)

We have found that the downhill simplex method *amoeba* [10, 13] produces an output object with acceptable appearance (although this is subjective). The time taken is also acceptable provided that

- the process is seeded with plausible initial values — we use the mean predicted for a face from applying the equation $d = -(px + qy + rz)$ to each vertex in turn ((p, q, r) is the face normal, (x, y, z) the vertex coordinate as output from topological completion).

- the topology is correct (it will be, providing previous stages have been successful).

Note that this method does not constrain faces containing no visible vertices. In our current system, the distances of these faces are not changed from the preliminary estimates calculated in Section 2.1. This is not entirely satisfactory — it would be preferable to adjust these distances after taking account of symmetry elements — but it may be observed that the only sketches we have tested so far which produce objects with faces containing no visible vertices are the Platonic and Archimedean solids, which are handled as a special case as described in Section 3.6.

Another weakness is that there is as yet nothing constraining edges which are “almost” the same length to be exactly the same length in the finalised geometry. This is a function of the set of constraints identified, not of the choice of algorithm — it is, in principle, straightforward to introduce an equi-length constraint requiring two edges to be the same length (this would be ignored by the face-normal adjustment process and affect only the face-distance process). Practical implementation details, such as the merit threshold above which such a constraint should be enforced, are still a matter for investigation, and other methods of achieving the same objective may prove preferable.

3 Special-Case Geometrical Finishing

During the development of our system, we experimented with an alternative approach to geometrical finishing: we attempt to interpret the completed topology wherever possible firstly as an axis-aligned object, then a semi-axis-aligned object, then a mirror-symmetric object, and so on through a predefined list of categories of decreasing symmetry to the worst-possible case, an asymmetric object in which no two edges are parallel (the selection could be based either on the sketch categorisation [17] or on the actual topology generated [18]). This approach would have the advantage that it

could be developed further to provide the user of the system with a choice of possible objects, with the most regular or symmetrical being given first. However, it is not possible to determine a universal order of precedence between, for example, axis-aligned objects and frusta — some sketches are more likely to represent one, some the other. Many are potentially both, and the choice should then depend on the geometry of the original sketch. Ordering our assumptions in a hierarchy based on degrees of symmetry and frequency of occurrence, rather than on a merit figure describing how well the assumption fits the sketch, seems over-restrictive.

Furthermore, many combinations of categories are possible, and adding more would result in a combinatorial explosion of cases to consider — separate handling of each combination would be impractical. Ideally, we should therefore prefer a single approach catering for all objects.

However, special-case methods will inevitably be faster, and it is likely that they may be demonstrably more robust (particularly since the general-case method is demonstrably not robust). These advantages may outweigh the disadvantages of special-case methods for commonly-occurring categories.

Since no entirely satisfactory general method has been found for finishing the geometry of an object, we recommend that special methods be used for objects which fall into the easily-processed categories listed below.

3.1 Right Extrusions

The end-caps of a right extrusion are known to be parallel to one another, and perpendicular to the sides. In order to complete the geometry, it is still necessary to find the orientations of the sides with respect to one another, and to determine the aspect ratio of sides to end caps.

If the object is believed to be an axis-aligned extrusion, we can use the axis-aligned category to generate side face normals. Otherwise, we use the general-case method for determining face normals, but with a significantly reduced number of constraints. The end caps are parallel to one another and perpendicular to the sides, and these facts are preset in the logical relationship database rather than used to generate constraints. The only constraints generated are angular constraints between adjacent sides and constraints from any mirror and rotational symmetry of the end caps. The mirror and rotational symmetry of the sides has no effect on their relative orientation, so no constraints are generated for these symmetries.

In terms of processing time, it would be somewhat quicker to have special-case code which deskews the front end cap (perhaps using skewed symmetry [5] if the front end cap has an axis of mirror symmetry or a “cubic corner”) to obtain rough estimates of the side face normals, and then imposes plausible symmetries and regularities to generate the

final normals. We do not consider that minor speed increases justify the effort involved in implementing this since we already have a system for imposing face parallelism and perpendicularity and rotational and mirror symmetries in 3D, and adding a new system to do much the same seems unnecessary. Unnecessary special case code is to be avoided.

In our system, the aspect ratio is determined by the general-case distance optimisation method described above (as are all other face distances). As an alternative, it would be possible to use the assumption of isometry of the projection to determine an aspect ratio — this might produce more plausible results, and would complement the approach of obtaining the front end cap geometry by deskewing.

3.2 Right Frusta

The method for right frusta is similar to that for extrusions, except that here, the end caps are known to be parallel to one another and known to be neither parallel nor perpendicular to the sides. Again, this knowledge can be preset as logical relationships rather than used to generate constraints.

3.3 Axis-Aligned Objects

For axis-aligned objects, we do not need individual face normals — if we have three orthogonal axes, we can align each face normal with the appropriate one. The three axes are estimated by grouping the face normals and taking the mean value of each. These are then made mutually orthogonal using the algorithm:

- Input three non-coplanar vectors $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}$
- reorder $(\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}})$ if necessary to form a right-handed coordinate system
- iterate
 - set $\hat{\mathbf{A}}_{i+1} = \text{normalise}(\hat{\mathbf{B}}_i \times \hat{\mathbf{C}}_i)$
 - set $\hat{\mathbf{B}}_{i+1} = \text{normalise}(\hat{\mathbf{C}}_i \times \hat{\mathbf{A}}_i)$
 - set $\hat{\mathbf{C}}_{i+1} = \text{normalise}(\hat{\mathbf{A}}_i \times \hat{\mathbf{B}}_i)$

We use four iterations, which is sufficient to produce axes perpendicular to within 3.6×10^{-8} degrees from every set of non-coplanar vectors we have tested.

Since face orientation has already been determined, no parallelism or perpendicularity constraints are generated and the face normal estimation process is bypassed (all face normals are aligned with one of the three axes). Mirror and rotation constraints are still generated for use in the distance optimisation process.

3.4 Semi-Axis-Aligned Objects

This categorisation does not provide enough information to bypass the general case entirely, but additional information is available.

As with axis-aligned objects, the face normals are grouped, and the groups which correspond to axis-alignment identified. The mean values for these are calculated — in the event that an axis has no face normal to it, the estimate is made by using the cross-product of the other two. The estimates are refined into three orthogonal axes as for axis-aligned objects, and the grouped face normals fixed to these axes.

All constraint types listed for the general case are still generated, including parallelism and perpendicularity (to allow for the possibility that the sketch represents an axis-aligned sketch but was not identified as such because of sketching inaccuracies). Normals for the faces not identified as axis-aligned are generated using the general-case normal adjustment method (the logical datasets are preset with parallelism / perpendicularity information for the axis-aligned faces to fix them in place), and all face distances are computed using the general-case face distance method.

3.5 Semi-Axis-Aligned Objects with Mirror Symmetry

Semi-axis-aligned objects with mirror symmetry follow the same route as those without mirror symmetry, but extra information can be deduced prior to the general-case face normal adjustment.

In enumerating the bundles to which edges of a particular face belong, any face which maps to itself across the mirror plane can be treated as including an edge using the mirror bundle whether or not any such edge actually exists; if its edges include one other axis-aligned edge, the axis-alignment of the face can be determined.

Where a hidden or partial face M can be paired across the mirror plane with an axis-aligned face N , the axis-alignment of M can be deduced given knowledge of the alignment of the mirror plane (earlier processing [17] identifies this as one of four possibilities, listed in Table 1 using the notation of Section 2.2.4). The logical datasets are preset with this information.

i	$B_0 \mapsto B_0$	$B_1 \mapsto B_1$	$V \mapsto V$
ii	$B_0 \mapsto B_1$	$B_1 \mapsto B_0$	$V \mapsto V$
iii	$B_0 \mapsto V$	$B_1 \mapsto B_1$	$V \mapsto B_0$
iv	$B_0 \mapsto B_0$	$B_1 \mapsto V$	$V \mapsto B_1$

Table 1. Semi-Axis-Aligned Mirror Planes

Any face in the mirror chain must be perpendicular to any face with a face normal grouped with the mirror bundle.

3.6 Platonic and Archimedean Objects

Although the general-case method for recovering topology works well for Platonic and Archimedean objects, the general-case method for finalising the geometry is particularly slow, since the “quick” logical operations for parallelism and perpendicularity apply to none of the faces, and all relationships must be determined by the “slow” operations for rotational and mirror symmetries. The general-case method is too slow to be considered interactive for these objects.

In addition, the general-case distance-optimisation method assumes that at least one vertex on every face is visible. This is not the case, for example, for the completed dodecahedron: the position of the back face of this can only be determined by special-purpose code using the symmetry of the object.

Since special-case code is needed, we recommend that this take the form of choosing the appropriate finalised geometry from the known finite set of Platonic and Archimedean objects, bypassing general-case geometry altogether.

3.7 Summary

The special-case methods listed above are successful in proportion to the extent to which they bypass the general case: finishing the geometry of axis-aligned objects is quick and produces accurate results, and the performance (and sometimes the geometrical accuracy) of semi-axis-aligned objects is improved too.

The principal disadvantage is the lack of generality. Different methods are used for different types of objects, and a new special-case category would require new methods. We believe that axis-aligned and semi-axis-aligned objects are common in engineering practice, and a survey supports this view [14], but this is not a guarantee that these will be a common feature of the sketches input by any particular user.

4 Future Work

It is planned to attempt to extend the range of sketches which can be interpreted by allowing non-trihedral vertices. These currently fail line-labelling, being identified and rejected in those cases where the non-trihedral vertex is fully visible.

It is also planned to attempt to extend the range of permitted input to allow sketches of objects with through holes and hole loops, and to extend the scope of reconstruction by partitioning difficult sketches into smaller, categorisable sub-sketches.

We are considering methods for inferring the presence of non-trihedral vertices at the sketch boundary or in the hidden

part of the object where doing so provides the most plausible interpretation of the sketch.

Further in the future, a system which can deal with simple curved objects may be attempted.

5 Acknowledgements

The authors are grateful to Unigraphics Solutions Inc. for providing Parasolid to the authors for use in this research.

References

- [1] L.Bauer, *Elimination with Weighted Row Combinations for Solving Linear Equations and Least Squares Problems*, Handbook for Automatic Computation **II**, Linear Algebra, eds. J.H.Wilkinson and C.Reinsch, Springer-Verlag 1971
- [2] A.Bowyer and J.Woodwark, *Introduction to Computing with Geometry*, Information Geometers, 1993
- [3] I.J.Grimstead, *Interactive Sketch Input of Boundary Representation Solid Models*, PhD Thesis, Cardiff University, 1997.
- [4] D.L.Jenkins, *The Automatic Interpretation of Two-Dimensional Freehand Sketches*, PhD Thesis, Cardiff University, 1992.
- [5] T.Kanade, *Recovery of the Three-Dimensional Shape of an Object from a Single View*, *Artificial Intelligence* **17** 409–460, 1981.
- [6] G.A.Kramer, *Solving Geometric Constraint Systems*, MIT Press, 1992.
- [7] R.S.Latham, *Combinatorial Algorithms for the Analysis and Satisfaction of Geometric Constraints*, PhD Thesis, Brunel University, 1996.
- [8] W.H.Macaulay, *Solid Geometry*, Cambridge University Press, 1930.
- [9] R.R.Martin and D.Dutta, *Tools for Asymmetry Rectification in Shape Design*, *Journal of Systems Engineering* **6** 98–112, 1996.
- [10] J.A.Nelder and R.Mead, *A Simplex Method for Function Minimization*, *Computer Journal* **7** 308–313, 1965.
- [11] J.C.Owen, *Algebraic Solution for Geometry from Dimensional Constraints*, in eds. J.Rossignac and J.Turner, Proc. Symposium on Solid Modeling Foundations and CAD/CAM Applications, 397–407, 1991.
- [12] D.N.Perkins, *Cubic Corners*, Quarterly Progress Report 89, 207–214, MIT Research Laboratory of Electronics, 1968.
- [13] W.H.Press, S.A.Teukolsky, W.T.Vetterling and B.P.Flannery, *Numerical Recipes in C, The Art of Scientific Computing*, Second Edition, Cambridge University Press, 1994.
- [14] M.M.Samuel, A.A.G.Requicha and S.A.Elkind, *Methodology and Results of an Industrial Parts Survey*, Technical Memorandum 21, Production Automation Project, University of Rochester NY, 1976.
- [15] K.Sugihara, *Machine Interpretation of Line Drawings*, MIT Press, 1986.
- [16] P.A.C.Varley and R.R.Martin, *A System for Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch*, Proc. GMP 2000, IEEE Press, 2000.
- [17] P.A.C.Varley and R.R.Martin, *Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch — Sketch Categorisation and Frontal Geometry*, 1st Korea-UK Joint Workshop on Geometric Modeling and Computer Graphics, 2000.
- [18] P.A.C.Varley and R.R.Martin, *Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch — Topology of Hidden Parts*, 1st Korea-UK Joint Workshop on Geometric Modeling and Computer Graphics, 2000.
- [19] C.E.Weatherburn, *Elementary Vector Analysis*, Bell, 1921.
- [20] W.Whitely, *Matroid and Rigid Structures*, in ed. N.White, *Matroid Applications*, Cambridge University Press, 1991.
- [21] H.W.Yankee, *Engineering Graphics*, Prindle, Weber and Schmidt, 1985.