Automatic Creation of Boundary-Representation Models from Single Line Drawings

Peter Ashley Clifford Varley

Department of Computer Science
University of Wales College of Cardiff
December 2002

A thesis submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy.

DECLARATION

is not being concurrently submitted in candidature for any degree.
Signed (candidate) Date
STATEMENT 1
This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by explicit references. A bibliography is appended.
Signed (candidate) Date
Signed (supervisor) Date
STATEMENT 2
I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.
Signed (candidate) Date

This work has not previously been accepted in substance for any degree and

Abstract

This thesis presents methods for the automatic creation of boundary-representation models of polyhedral objects from single line drawings depicting the objects. This topic is important in that automated interpretation of freehand sketches would remove a bottleneck in current engineering design methods. The thesis does not consider conversion of freehand sketches to line drawings or methods which require manual intervention or multiple drawings.

The thesis contains a number of novel contributions to the art of machine interpretation of line drawings. Line labelling has been extended by cataloguing the possible tetrahedral junctions and by development of heuristics aimed at selecting a preferred labelling from many possible. The "bundling" method of grouping probably-parallel lines, and the use of feature detection to detect and classify hole loops, are both believed to be original. The junction-line-pair formalisation which translates the problem of depth estimation into a system of linear equations is new. Treating topological reconstruction as a tree-search is not only a new approach but tackles a problem which has not been fully investigated in previous work.

Acknowledgements

I wish to thank my supervisor, Prof. Ralph Martin, firstly for proposing such an interesting research project, then for his invaluable advice throughout the course of my research, and finally for helping to translate the results (this thesis) into comprehensible English.

I also wish to thank my colleagues in the vision laboratories, Ana, Bruce, Chunhua, Elena, Frank, Gavin, Gundon and Julia, for their many stimulating discussions of ideas of mutual interest and for reassuring me that a fascination with computer vision is not a sign of madness.

I am grateful to the Computer Science Department of Cardiff University for providing the funding for my research, and to Unigraphics Solutions Inc. for providing Parasolid for use in this research.

Contents

A	Abstract Acknowledgements			
\mathbf{A}				
1	Introduction			
	1.1	Introduction and Context	1	
	1.2	Terminology	3	
	1.3	Previous Investigations	5	
	1.4	Discussion of Aims	9	
	1.5	Thesis Structure	12	
	1.6	New Ideas in this Thesis	13	
2	Pro	olem Statement and Proposed Solution Overview	15	
	2.1	Problem Statement	15	
	2.2	Alternatives and Possible Extensions	16	
	2.3	Solution Overview	17	
	2.4	Component: Line Labelling	18	
	2.5	Components: Subgraphs and Regions	22	
	2.6	Component: Parallel Lines	25	
	2.7	Component: Two-Dimensional Tidying	25	
	2.8	Component: Feature Identification	26	
	2.9	Component: Inflation	27	
	2.10	Component: Validation of Labelling	28	
	2.11	Component: Local Symmetry	30	
	2.12	Component: Classification	31	
	2 12	Component: Topological Possenstruction	29	

	2.14	Component: Face Loops	33
	2.15	Component: Geometric Finishing	35
	2.16	Component: Splitting and Recombination	36
	2.17	Component: Intersecting Faces	37
	2.18	Component: Quality Control	37
	2.19	Chosen Components: Order and Control Structure	38
3	Bac	kground Ideas	40
	3.1	Sketch to Drawing	40
	3.2	Searching and Heuristics	41
	3.3	Constraints and Optimisation	44
	3.4	Least Squares Fit	46
	3.5	Planar Geometry	46
	3.6	Solid Geometry	47
	3.7	Dual Space	51
	3.8	Miscellaneous	52
4	Line	e Labelling	53
	4.1	Introduction	53
	4.2	History	54
	4.3	Tetrahedral Junction Catalogue	62
	4.4	Two Labelling Approaches	70
	4.5	Results and Conclusions	83
5	Para	allel Lines	91
	5.1	Introduction	91
	5.2	History	92
	5.3	Reproduction of Bucketing	93
	5.4	Partitioning into Bundles	94
	5.5	Corners and Face Planes	96
	5.6	Results and Recommendations	97

6	Fea	tures	100
	6.1	Introduction	. 100
	6.2	History	. 101
	6.3	Implementation	. 102
	6.4	Underslots and Valleys	. 103
	6.5	Cofacial Configurations	. 104
	6.6	Hole Loops from Cofacial Configurations	. 107
	6.7	Results	. 109
7	Infl	ation	113
	7.1	Introduction	. 113
	7.2	History	. 115
	7.3	Compliance Functions	. 116
	7.4	Grimstead's Linear System Approach	. 129
	7.5	Depth from Labelling	. 130
	7.6	Results and Conclusions	. 133
8	Loc	eal Symmetry Detection	146
	8.1	Introduction	. 146
	8.2	History	. 147
	8.3	Compatibility, Pairing, Propagation	. 150
	8.4	Figures of Merit	. 152
	8.5	Data Identified	. 153
	8.6	Mirror Chains	. 157
	8.7	Results	. 158
9	Cla	ssification	160
	9.1	Introduction	. 160
	9.2	History	. 161
	9.3	Classes	. 161
	9.4	Combining Classes	. 168
	0.5	Results	160

10	Reconstruction of Hidden Topology	175
	10.1 Introduction	175
	10.2 History	176
	10.3 Number of Possible Completions	178
	10.4 Control Mechanism	184
	10.5 Move Types	189
	10.6 Hypothesising Moves	195
	10.7 Hypothesis Adjudication	208
	10.8 Special-Case Recovery of Topology of Hidden Parts $\dots \dots$.	212
	10.9 Results	214
11	Geometric Finishing	224
	11.1 Introduction	224
	11.2 History	226
	11.3 Constraints	238
	11.4 Face Normals—Simple Downhill Optimisation	245
	11.5 Face Normals—Enhanced Downhill Optimisation	249
	11.6 Face Normals—Geometric Optimisation	257
	11.7 Face Normals using a Genetic Algorithm	259
	11.8 Face Distances—Simple Downhill Optimisation	262
	11.9 Face Distances—Enhanced Downhill Optimisation	265
	11.10Intersecting Faces	268
	11.11Special Classes	268
	11.12Results	273
12	Results	282
	12.1 Axis-Aligned Extrusion	282
	12.2 Axis-Aligned Non-Extrusion	284
	12.3 Grimstead's Block	284
	12.4 Hole Loop	286
	12.5 Extended Trihedral Normalon	287
	12.6 Non-Trihedral Semi-Normalon	289
	12.7 Semi-Normalon	290
	12.8 Sami-Normalon	202

	12.9	Non-Trihedral Semi-Normalon	. 293
	12.10	0Non-Trihedral Bracket	. 295
	12.1	1 Conclusions	. 296
	12.13	2Timing	. 296
13	Con	aclusions	298
	13.1	Line Drawing Interpretation	. 299
	13.2	Sketch to Line Drawing	. 301
	13.3	Curves	. 302
	13.4	Features	. 302
	13.5	Psychology	. 303
\mathbf{A}	Glo	ssary	305
В	Test	t Drawings	307
	B.1	Trihedral Genus Zero Polyhedra	. 307
	B.2	Non-Trihedral Genus Zero Polyhedra	. 315
	В.3	Objects with Through Holes	. 324
	B.4	Multiple Polyhedra	. 327
	B.5	Figures based on Collections	. 327
\mathbf{C}	Tun	ing Constants	333
	C.1	Tuning: Configurable Constants	. 333
	C.2	Tuning: Introduction	. 336
	C.3	Tuning for the Labelling Problem	. 339
D	Figu	ares of Merit	344
${f E}$	Jun	ction Catalogue Illustrations	350
	E.1	Trihedral Catalogue	. 350
	E.2	Tetrahedral Catalogue	. 351
\mathbf{F}	Geo	ometric Analysis	360
	F.1	Rotation Axis from Start and End Points and Angle	. 360
Bi	bliog	graphy	364

Chapter 1

Introduction

1.1 Introduction and Context

This thesis describes the automatic creation of boundary-representation models of polyhedral solid objects from single line drawings depicting the objects. This topic is important in that automated interpretation of freehand sketches would remove a bottleneck in current engineering design methods. The thesis does not consider conversion of freehand sketches to line drawings (this is already well covered in the literature), or methods which require manual assistance or multiple drawings.

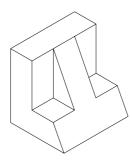
Textbooks on engineering drawing (e.g. [20]) emphasise the importance of free-hand sketching in the design process. Studies such as Jenkins [59] have shown that engineers and architects, when creating a new design, start by sketching ideas free-hand on paper, and follow this, once a satisfactory concept has been found, by manually copying the design to a CAD package. Automating this process would remove a bottleneck. In order to achieve this, a freehand sketch must be converted into a boundary representation solid model of the most plausible 3D interpretation of the sketch, and in a reasonable time. Manual intervention is undesirable—the engineer will wish to concentrate on creating an idea, not on the mechanics of using a computer package. The problem of automatic conversion of a 2D drawing to a 3D object forms the subject of investigation of this thesis and is stated more precisely in Chapter 2.

The theoretical impossibility of perfect conversion of a single 2D view of an object to a full 3D model is both obvious and well-known, but attempts to relate

this geometric computation problem to the philosophical debates of past centuries can be overstated. For example, Mill's refutation of Hamilton's philosophy [109] is sometimes cited as historical background, but that discussion considered only perception of things, not with perception of pictures as representing things, and it seems certain that all parties involved in the controversy were well aware that what is perceived when the eye sees and the mind interprets is not "the thing in itself". One of Mill's points remains noteworthy: that interpretation of any visual scene is a practical skill learnt from experience, not an arcane art requiring the intervention of mystical forces.

During the course of the investigations which led to this thesis, it has become clear that human interpretation of line drawings is similarly a skill which has to be learnt (Lipson [90] reached the same conclusion). This has two important consequences.

Firstly, an application domain must be defined. Engineers do not necessarily see the same things in line drawings as do architects or geometers, and certainly make assumptions (based on experience) when viewing a line drawing which differ from those made by people without that experience. Even Figure 1.1 can be ambiguous to those lacking any experience of interpreting line drawings (for example, the "obviously" concave Y-junction may be interpreted as convex, and vice versa), and interpretation of Figure 1.2 depends on what the viewer perceives the function of the object to be (does the square hole at the top of the object indicate a through hole, with the object being designed to slide up and down a square bar, or does it indicate a pit, with the object being designed to hold the bar in place?).



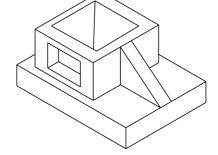


Figure 1.1: Line Drawing from [194]

Figure 1.2: Line Drawing from [128]

Secondly, the rules underlying any skill can in principle be elucidated, and it is

upon these rules that any attempt to program the skill into a machine should be based. Problems should be solved by means which correspond as closely as possible to geometric intuition. Draper [23], in advocating sidedness reasoning, makes the interesting statement that it is "more intuitively correct" than gradient and dual space algorithms, which it displaced. My aim in this thesis is also for methods which are intuitively correct. Although the methods outlined here sometimes fail, this should be taken, not as a recommendation for less-intuitive methods, but as indicating that engineers are subtle, and accustomed to applying more rules, or more complex rules, than I have been able to identify in the time available.

1.2 Terminology

A 3D object has faces, edges where pairs of faces meet¹, and vertices where edges meet. Faces, edges and vertices are here called the atoms of the object. It is polyhedral if all faces are planar. It is a normalon [17] if all face normals are aligned with one of three mutually-perpendicular axes.

A line drawing is a 2D pictorial representation of an object. A natural line drawing [163] is a line drawing where only the object's visible edges and parts of edges are shown. Lines in the drawing represent the object's edges (sometimes, in a natural line drawing, visible parts of edges). Lines intersect at junctions, and cycles of lines subdivide the drawing into regions. A junction where two lines meet is biconnected; a junction of three lines is triconnected. Regions, lines and junctions are the atoms of the drawing.

A drawing is from a *general viewpoint* if no small change in viewpoint changes the topology of the drawing. From a general viewpoint, no pair of vertices is collinear with the viewpoint, and no face is coplanar with the viewpoint. Note that some (e.g. [163]) use a stricter definition of general viewpoint, requiring also that no pair of edges is coplanar with the viewpoint—such a requirement is intolerant of freehand sketching errors and cannot reasonably be enforced (see Chapter 5).

A *sketch* is a freehand drawing. Lines in a sketch may be duplicated for emphasis. Figures 1.3 and 1.4 are example sketches; Figures 1.5 and 1.6 are the corresponding line drawings. Figures 1.1 and 1.2 are further examples of line drawings from general

¹This thesis considers only manifold objects

viewpoint.

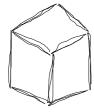


Figure 1.3: Sketch of Cube



Figure 1.4: Sketch of House

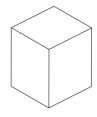


Figure 1.5: Line Drawing of Cube

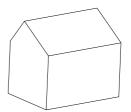


Figure 1.6: Line Drawing of House

Appendix B shows the complete set of general-viewpoint line drawings used as test data in this thesis. For example, Figures 1.1 and 1.2 can be found therein as Figures B.503 (page 329) and B.518.

Each vertex in the object has an underlying vertex type, which depends on the number, type and configuration of the edges meeting at that vertex; this is discussed further in Chapter 4. Polyhedral objects are trihedral if exactly three faces meet at each vertex, extended trihedral [120] if exactly three planes meet at each vertex (there may be four or more faces provided that some are coplanar) and tetrahedral if no more than four faces meet at any vertex. For example, Figure 1.7 is trihedral, Figure 1.8 is extended trihedral, and Figure 1.9 is tetrahedral.

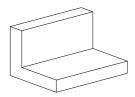


Figure 1.7: Trihedral

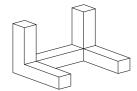


Figure 1.8: Extended Trihedral

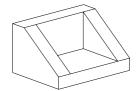


Figure 1.9: Tetrahedral

A drawing is *recognised* if the computer uses it to choose one of a finite set of candidate objects. It is *interpreted* if the computer uses the drawing to create a new object from an infinite set of constructible objects. This thesis is concerned only with interpretation.

1.3 Previous Investigations

Although Roberts [139] aims for recognition rather than interpretation, his program is capable of interpreting "compound" objects as assemblies of primitive objects it recognises (cuboids and triangular wedges), and can thus be considered the first in the field of machine interpretation of drawings. The program is aware that the "join" between two primitives produces no lines, and since it can interpret Figure B.29 correctly, it must also know that a complete face of one primitive may match part of a face of another.

Guzman's program SEE [42] takes another approach to Roberts's problem, using heuristics rather than numerical analysis to identify both known primitive objects and the spatial relationship between them. Falk's program INTERPRET [26] illustrates an advantage of this approach. Since it breaks a scene down into occurrences of a small number of primitives, it requires merely a good match, not a perfect match, and is thus tolerant of drawing errors.

Despite these early successes, subsequent approaches to line drawing interpretation followed different ideas. Wang and Grinstein [184] describe and assess seventeen approaches to interpreting 2D drawings as 3D objects, of which seven are based on single drawings. The earliest of these is the Clowes-Huffman line-labelling [14, 56], described in more detail in Chapter 4. Both Clowes and Huffman were more interested in the problem of whether a line drawing had a polyhedral interpretation than that of finding the best interpretation; Huffman's original idea [56] was not formulated as an algorithm, and Clowes's implementation, OBSCENE [14], was intended to explore the idea of picture grammars. Malik's extension to line-labelling [100] is also described in more detail in Chapter 4.

Waltz [181] extends Clowes-Huffman line-labelling to allow interpretation of shadows and cracks. This is useful for processing drawings produced from camera pictures, but less useful for interpreting drawings produced from freehand sketches.

Mackworth's program POLY [97, 96] builds on Huffman's use of dual space [56] (see Chapter 3.7) and introduces the idea of gradient space. By analysing and checking consistency in gradient space, POLY can not only detect as invalid some drawings which OBSCENE regards as valid but can also obtain some spatial information (relative orientations of visible object faces) from the drawing. Although it

remains limited to error-free drawings of trihedral polyhedra, the trihedral limitation does not appear to be inherent in Mackworth's method and it is also reported [184] that Wei [188] extended Mackworth's method to allow for non-perfect input.

Sugihara observes [159] and proves [162] that the necessary and sufficient condition for a line drawing to have a valid geometric interpretation is that a consistent set of vertex depth coordinates and face equations exist for which (i) all non-occluding vertices lie exactly on their faces and (ii) at each occluding junction, the occluding face is nearer the viewer than the occluded face². Sugihara himself observes that testing this apparently straightforward condition may prove problematic in practice, as even roundoff errors may make the system of depth coordinates and face equations "inconsistent".

Lamb and Bandopadhay [77] start by attempting to identify three bundles of lines (see Chapter 5) which correspond to three perpendicular axes of the object. After choosing a reference vertex, it is in many cases then possible to determine relative spatial locations of the other vertices by propagating distances along lines in the three chosen bundles. They report that their approach makes semi-normalons too square, but this is presumably a consequence of their bundling algorithm rather than a fault inherent in their method. More seriously, the method relies on being able to determine unambiguously which three bundles of lines correspond to the three axis directions.

Lamb and Bandopadhay [77] also report the existence of a method for determining hidden topology, but do not describe it.

Wang and Grinstein [183] produce a CSG representation of the object depicted in the drawing. Their method was originally restricted to normalons (which are inherently trihedral or extended trihedral), with the single CSG primitive being a cuboid. It was later [182] extended to non-normalon polyhedra with the addition of a second CSG primitive, a tetrahedron. Since this approach requires a labelled line drawing, the implementation of which used the trihedral catalogue, it is unclear whether or not these methods can be extended to non-trihedral polyhedra.

Wang also differs by taking an unusual approach to drawing errors. Whereas the usual assumption is that junctions in the drawing may be slightly misplaced, Wang assumes that if the drawing has no geometrical interpretation, the error is that a

²Some unexpected drawings turn out to be provably realisable by this criterion—see [160, 164].

line is missing.

More recently, Qin et al [137] describe another system which constructs solid objects from CSG-style primitives which must be entered individually. They assume exact isometric projection. It is difficult to regard this as an advance on Wang's approach.

Lipson and Shpitalni [91, 92] have developed a method for inflating a wireframe drawing (a drawing where hidden lines are shown) into a 3D object. It is tolerant of freehand sketching errors, and allows planar and cylindrical faces. It assumes a single polyhedral object and general viewpoint. Ideas from this approach are discussed in Chapters 7 and 11. The Regeo project [16, 15, 17] has developed a similar implementation as part of their investigations into a language of geometry.

Pugh [133] not only requires all lines to be drawn but requires the user to specify which are hidden and which are visible. His sketches are subject to the same restrictions as ours, except that he has a labelling method, arc-labelling, which uses a junction catalogue for tetrahedral objects [132]. He considers extension to pentahedral and higher junction types to be straightforward, incorporation of hole loops to be possible but probably not worth the additional processing overhead, and curved surfaces to be incompatible with the underlying assumptions of the system. In Pugh's system, topology is determined before the geometry is adjusted to meet user-specified constraints. The user must also specify which vertices may be moved and which remain fixed. The system is interactive in that vertices can be added and deleted, and even separate objects added and merged to create more complex objects. The resulting user interface departs from the requirement for a natural, easy-to-use sketching tool.

Grimstead [38] provides a prototype system based on input of natural line drawings of single trihedral polyhedra. This comprises five stages: incremental line labelling; two-dimensional drawing tidying; conversion to 3D using a linear system; recovery of hidden parts; and three-dimensional drawing tidying. Since Grimstead's system makes similar assumptions to those listed in Chapter 2, several comparisons will be made later between Grimstead's methods and those described in this thesis.

Moving away from freehand drawing, Barrow and Tenenbaum [2] obtain line drawings by preprocessing greyscale pictures into region boundaries. Their objective is to produce depth information on a per-pixel basis, thereby obtaining a " $2\frac{1}{2}D$ " sketch [103]. The system can handle semicircles, cylinders and spheres, using Chakravarty's junction catalogue [13]. However, if the input sketch contains curves, the iterative algorithms used converge too slowly to be useful for an interactive system.

Similar methods are also used in analysis of aerial photographs, although as Mayer's survey [105] shows, this field has subsequently become more specialised. Early work analysed single pictures by modelling buildings, firstly as rectangular prisms [48] and later by grouping rectangular regions and parallel lines [113]. Artefacts such as skewed symmetry [63] (see Chapter 7.3.9) are equally applicable to such pictures and to interpretation of sketches.

Nagendra and Gujar [115] list eleven algorithms reported between 1973 and 1984 for recovering 3D objects from three orthographic 2D views. They make the point that even given three orthographic 2D views (the ideal starting-point), perfect recovery is not always possible.

As a recent example of work in this field, the two-stage extrusion process of Shum et al [150] is worth noting. Initially, the interiors of each 2D view are extruded, and a solid obtained by intersecting the resulting volumes. Usually, there will be lines in one or more of the drawings which this initial solid would not produce; a second stage of extrusion is used to account for these.

Reconstruction from two orthogonal images is not always straightforward even if there is a template available for the object being viewed. Lee et al [86] use semi-automatic rather than fully automatic methods for generating models of human heads (however, their main problem is texture rather than shape).

More distant fields also produce results of interest. One such is the detection of motion of an object in an image, or of egomotion (movement of the viewpoint with respect to the image). Useful mathematical results can be found in Kanatani [64].

While previous work in this field has produced useful results, several problems remain unsolved:

 Most previous work assumes that the drawing is trihedral (some follow Waltz [181] in allowing a small subset of common non-trihedral junction types)—no previous work allows for all possible non-trihedral junction types.
 As well as being an inconvenient restriction, this means that the validity of much previous work when applied to the domain of non-trihedral objects remains unproven.

- All previous work on single line drawings assumes that there are no hole loops. Again, as well as being an inconvenient restriction, this means that the validity of much previous work when applied to the domain of objects containing cofacial loops remains unproven.
- No satisfactory solution has been found to the problem of determining which lines in an imperfect drawing are intended to be parallel. Indeed, much previous work assumes perfect drawings.
- No previous work addresses identification of common machining features in single line drawings (see, for example, Figure B.445, where unambiguous interpretation of the implied slot feature requires domain-specific knowledge).
- No satisfactory solution has been found to the problem of deducing the hidden topology of the object.
- No previous work makes use of potential symmetries implied by the line drawing when attempting to deduce the hidden topology of the object.
- Progress in the field of geometric constraint satisfaction has not been applied to the problem of determining a geometry for the deduced topology of the object.

This thesis attempts to address all of these problems.

1.4 Discussion of Aims

A point which will be repeated several times in this thesis is that, since the ultimate aim is to interpret the user's intended object, the information entered by the user should be preserved throughout the interpretation process. It should not be "tidied" in any way before the final stage of matching the program's interpretation to the original line drawing. Intermediate "correction" or other manipulation of user input is undesirable, and choice between methods will in a number of cases be made on this criterion.

Previous line drawing analysers took various types of drawing as input. Some required drawings showing shaded faces, others simply lines representing edges, and in the latter case the drawing might be required to show hidden lines, or only the visible lines. Some assume a single object; others allow *cracks* (lines indicating a discontinuity of material); others still allow *scenes* (multiple objects). Some assume a parallel projection, while others assume a perspective projection. Many are limited to drawings of polyhedra (no curved surfaces), and most are limited to drawings of trihedral objects.

The disadvantage of using sketches with hidden lines visible, that of ambiguity (Necker reversal), is in principle insoluble, although frequently the symmetry of the drawn object makes the two potential interpretations the same. There are other disadvantages of using wire-frame sketches rather than sketches without hidden lines: it is more natural to draw only that which can be seen, and it is quicker and easier to draw a smaller rather than a larger number of lines.

A further advantage of interpreting sketches with only visible lines is that this makes it easier to incorporate work deriving from attempts to recognise real-world objects. Naturally, if the two-dimensional data is obtained from a photograph of a real object rather than from a sketch, hidden lines will not be visible. Interpretation of line drawings derived from photographs is inevitably more complex than interpretation of line drawings derived from sketches, as it cannot enforce assumptions on the real world in the way that a sketching interface can enforce assumptions on its user. It must be able to handle multiple objects, only some of which are of interest; it must be able to convert shading indicating differently-oriented faces into lines representing edges; it must be able to cope with noisy input. Nevertheless, it has one advantage which analysis of sketch input does not: it can assume that the input is valid, that it genuinely represents a real object, and (after allowing for lens distortion) that the projection is "drawn" correctly.

It is considered that in a quick, natural sketch input system the advantages of drawing only the visible lines outweigh the disadvantages.

Similarly, the requirement for a quick, natural system mandates that the system must allow for freehand sketching errors. The idea of snap-to-grid (as used by, for example, Pugh [133]) is not only less natural than freehand sketching, but presuppose that a suitable grid has been specified beforehand, and limit the designer's

subsequent creative freedom. The same requirement rules out menu-based correction of drawing errors, which although less limiting (the program does not intervene until asked to) is even further away from the ideal of a natural sketching interface.

Interpretation of drawings containing curved lines presents several unsolved problems even for perfect drawings. This thesis only investigates interpretation of drawings of polyhedra, and (as will be seen in Chapter 13), considerable further work is necessary before this problem can be considered solved. Interpretation of drawings containing curved lines is a far harder problem—many of the simplifying assumptions made in this thesis (such as that if two edges meet the same two faces, the edges must be collinear) do not hold if curved objects are permitted.

Even in the domain of polyhedra, it is found that some facts have non-local consequences. However, in the domain of curved line drawings, this becomes a serious obstacle to interpretation, as can be seen by considering Yonas's curves (Figure 1.10), in which turning the bottom line from a straight line to a curve changes the perception of the curved top lines.

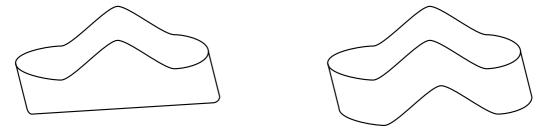


Figure 1.10: Yonas's Curves [2]

Another example given by Barrow and Tenenbaum [2] illustrates the problem of distinguishing similar drawings which depend on obscure mathematical points for their interpretation. Both the slice of cake (Figure 1.11 and the rocket nose-cone (Figure 1.12) are valid drawings; they are distinguished by the "obscure 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 remains valid when applied to the domain considered by this thesis).



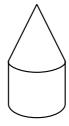


Figure 1.11: Slice of Cake

Figure 1.12: Rocket Nose Cone

Thus, the problem investigated in this thesis is to take as input a natural line drawing of a single manifold polyhedral object, and to produce from it a boundary-representation model of the object portrayed. The drawing must be from a general viewpoint and must be topologically correct, but need not be geometrically perfect. No artificial restrictions (such as snap-to-grid) are imposed on the freehand drawing process other than that all lines are straight and all lines terminate at junctions of at least two lines. No user input is required other than the line drawing itself, and at no stage will the user be prompted for further information.

1.5 Thesis Structure

The problem, as stated, is complex and requires subdivision in order to make it more tractable. The constituent subproblems are identified in Chapter 2 (which also covers some subproblems considered but rejected as part of the overall approach) and described in more detail in Chapters 4–11.

Between them, Chapter 3 gives general overviews of "imports": results from outside the field of line-drawing interpretation which are used in the thesis. These include symmetry (what it is, and why it is relevant) and solid geometry (listing the results of which use is made).

Except where specifically noted otherwise, ideas described in this thesis have been implemented in a computer program, RIBALD (Reconstructs Interactively B-Reps by Analysing Line Drawings). In practice, several other groups have attacked this or similar problems, and there exists an expanding set of drawings for which conversion can be achieved; RIBALD is claimed to be the most flexible so far, in that by incorporating the ideas presented in this thesis, the set of drawings which it can process is larger than previous programs.

Experimental results are presented in Chapter 12, and conclusions (and recommendations for possible future work) are summarised in Chapter 13.

Test drawings shown in Appendix B have been accumulated from a variety of sources, including line-labelling literature [38, 148, 163], all of the solid objects (but not the paper objects or wire-frames) from [83] and all of the planar objects from two engineering drawing textbooks [128, 194]. There has been no selection other than that if a drawing looks like a polyhedron, it has been included.

Throughout the thesis, timings are in seconds and were obtained on a Sun Ultra 10. Where problem size is quoted without other explanation, it is the number of lines in the drawing.

1.6 New Ideas in this Thesis

The thesis contains the following novel contributions to the art of machine interpretation of line drawings. Some of these original contributions have been published or accepted for publication in journals or conference proceedings; where this is the case, it is indicated by a citation. Where no citation is given, the ideas are described for the first time in this thesis.

The tetrahedral junction catalogue shown in Appendix E and my automated method for deriving it described in Chapter 4.3.1 have been published in a journal [175]. Although partial tetrahedral junction catalogues have been used in prior work, and Huffman [58] suggested a general method for cataloguing junctions, it is believed that this is the first complete tetrahedral junction catalogue.

An abbreviated description of the heuristics described in Section 4.4.1, used to choose between alternative valid labellings, was included in a conference paper [177]. These heuristics, and their inclusion in line-labelling algorithms, are believed to be original. The labelling algorithms given in Chapters 4.4.2 and 4.4.3 are otherwise straightforward extensions of standard computer vision techniques to the non-trihedral labelling problem. The results of a comparison between them, given in Chapter 4.5, and the inferences drawn from these results, appeared in the same conference paper [177] and are original.

The "bundling" method of grouping probably-parallel lines, described in Chapter 5, is believed to be original and has been described in a conference paper [172].

The use of feature detection to solve many of the problems associated with hole loops is believed to be original, as is the choice of features (the "cofacial configurations") in Chapter 6.5. This has appeared in a conference paper [177].

Although the idea that junction labels of neighbouring junctions imply relative depths is not new, its "algorithmisation" as a set of linear equations in Chapter 7.5 is believed to be original. A preliminary version of this idea, restricted to trihedral junction labels, appeared in a conference paper [172], and a truncated description of later work appeared as a conference paper [176].

The ideas in Chapter 8 include several incremental improvements on the state of the art but no original contribution.

Almost all of Chapter 10 is believed to be original (a few ideas are inherited from Grimstead [38]; these are indicated by citations). A preliminary version of Chapter 10, restricted to trihedral polyhedra, has appeared as a conference paper [173].

Precedence is difficult to establish for the ideas in Chapter 11. A preliminary version, restricted to trihedral polyhedra, appeared as a conference paper [174]. The ideas of a two-stage process fitting face normals and face distances separately, and of using successive iterations of downhill optimisation to decide whether constraints can be accommodated, were believed at the time to be original, but essentially the same ideas appear in [75] and [32] respectively. Chapter 11 includes incremental improvements on all three, as well as comparative material which does not appear elsewhere.

Chapter 2

Problem Statement and Proposed Solution Overview

2.1 Problem Statement

The problem studied is that of converting a two-dimensional freehand sketch, with hidden lines removed, of a single polyhedral object with no cracks or shadows, into a boundary representation solid model.

Demonstration systems exist which convert freehand sketches to line drawings (e.g. those in [25, 112, 137]). The problem of deducing the solid object which the line drawing represents is more difficult, and this is the subject of this thesis.

Some assumptions concerning the drawing are required. It is assumed that the drawing is of a single manifold polyhedral object—all faces are planar. The object is assumed to be viewed from a general viewpoint. It is assumed that the object has been drawn from the "most informative viewpoint"—there is nothing at the rear of the object which could not reasonably be inferred from the visible part of the object. It is further assumed that the user is ambidextrous—left-handed and right-handed versions of chiral objects have equal merit.

For the system to be useful, it must perform the conversion in a "reasonable" time. A second or less on a powerful personal computer is a reasonable target figure.

There are practical limits to what can be drawn—it is unlikely, for example, that anyone would draw Figure B.132 (page 313), which has 98 lines. It is therefore reasonable to assume that drawings will have fewer than 100 lines. As a result,

order of algorithms in terms of the problem size (generally the number of lines) is not of great importance—even $O(e^n)$ would be acceptable if, when implemented, the algorithm were to run to completion in less than a second with a problem size of 100. Orders—actual or expected—of algorithms are sometimes noted in this section, as they influenced choice of which methods to investigate, but they are not in themselves a criterion for acceptability.

2.2 Alternatives and Possible Extensions

Alternative input methods and formats are possible. For example, freehand drawing errors could be eliminated by using a "snap-to-grid" approach when creating the initial line drawing. This idea has been rejected for the time being, partly because the objective is to provide as simple and natural an interface as possible, and partly because until the problems caused by freehand drawing errors are known, and known to be insoluble, eliminating them is of no proven benefit.

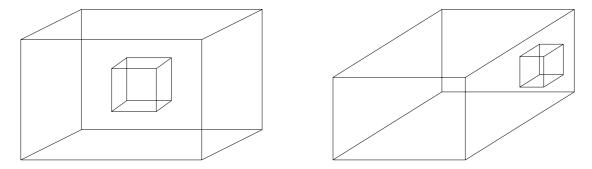


Figure 2.1: Ambiguous Wireframe Drawings

Wireframe line drawings, showing all edges, are a possible alternative input format to natural line drawings, and would eliminate the need for reconstructing hidden topology. However, balancing this, there are problems in interpreting wireframe line drawings which do not occur for natural line drawings. The best-known and most obvious is that of resolving Necker ambiguity [116], but another and more serious problem is illustrated in Figure 2.1. While there is little doubt that, for example, Figure B.422 shows an object with a boss, and Figure B.438 shows an object with a shallow pocket, it is far from clear what features are present in the drawings in Figure 2.1, or even which faces contain the features [130]. Given that

either choice for the type of input drawing leaves problems to be solved, the preference for a simple interface is decisive—natural line drawings have fewer lines and are thus easier to draw.

The choice of output format—boundary representation (B-rep) model, rather than constructive solid geometry (CSG) model, is determined by the fact that both commonly-used CAD kernels, ACIS [18] and Parasolid [168], are based on B-rep. This choice is more flexible as it imposes no restrictions on the methods used: since conversion of CSG models to B-rep is straightforward, but not vice versa [5], methods which use either representation can be used to produce B-rep output. However, the choice may limit future flexibility, as use of methods which use B-rep representation (all in this thesis are in this category) precludes production of CSG models.

I suggest that as the large majority of curves in engineering objects are cylindrical through holes (88% according to [143]), a simple approach to handling these would be of considerable practical benefit (although not of any great theoretical interest). Input of ellipses could be added to a freehand drawing package, either by specifying axes or by fitting the best ellipse to a freehand curve. The ellipse would be interpreted by the system as a cylindrical through hole (as a bonus, the face normal of the face in which the hole is drilled could be estimated by applying the skewed symmetry method [63] to the axes of the ellipse). This suggestion has not been implemented.

An alternative method of inputting cylindrical through holes could be implemented as a postprocessing stage. For display purposes, RIBALD allows the completed 3D object to be rotated interactively. A drill axis could be specified by rotating the object to the correct orientation and then indicating the axis and drill radius. This idea has not been implemented.

2.3 Solution Overview

The remainder of this chapter identifies possible components of a solution to the problem outlined above, and concludes with a list of the components actually chosen.

For each component, inputs and outputs are listed. Inputs may be either required (the component cannot function without them) or preferred (the component can function without them, but performance or reliability are improved if they are

available). Outputs are either primary (the component exists in order to produce this information) or secondary (the component produces this information as a sideeffect, or other ways of obtaining it are as good, if not better).

By matching inputs of one component with outputs from another, it is possible to determine how a solution can be built from the blocks available.

In many cases, there may be several plausible alternatives during reconstruction. A figure of merit is assigned to each. This is a numerical value intended to represent the plausibility of a particular choice; higher values represent more plausible choices. Out of a set of alternatives, the one with the highest figure of merit will be considered first. Figures of merit are justified in Chapter 3.2 and described in more detail in Appendix D.

Some components use heuristics. Balancing the numerical values of competing heuristics presents problems. In the program RIBALD, used to test the ideas in this thesis, several heuristics make use of tuning constants designed to make the balancing process easier. Tuning constants are listed and described in Appendix C.2.

2.4 Component: Line Labelling

Machine vision systems often use the technique of *line-labelling* [14, 43, 56] to reduce the possible 3D interpretations of a line drawing to a processable number. Line-labelling attempts to identify each line in the drawing as either convex, concave or occluding. A Clowes-Huffman-style line-labeller following the algorithm in Kanatani [64] is outlined here and described in more detail in Chapter 4. Line labelling is preceded by the minor task of junction type identification.

As shown in Chapter 4, deterministic labelling algorithms are theoretically $O(4^n)$ in the worst case, although often almost O(n) in practice for drawings of trihedral objects. For this reason, a faster but potentially less reliable alternative labelling approach based on relaxation labelling has also been investigated; this too is described in Chapter 4. Junction type identification takes low-order polynomial (in principle, O(n)) time.

2.4.1 Notation

A standard notation for trihedral junction labels is emerging—this thesis follows Wang [182]. Junctions of two lines are termed L-junctions. Junctions of three lines are T-junctions, W-junctions or Y-junctions according to shape. Notation of non-trihedral junction labels is not standardised as yet—this thesis adapts notation from Waltz [181] and Chakravarty [13]. Junctions of four lines are K-junctions, M-junctions, X-junctions and Z-junctions, again according to shape. See Figure 2.2. (It will be found that in some cases there is also an "invisible" junction implied by the structure of the object but not visible in the line drawing; this is termed an I-junction.)

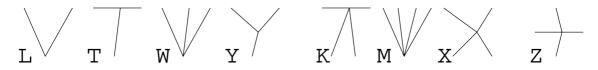


Figure 2.2: Junction Types

Lines are convex, concave or occluding. In diagrams, convex lines are shown as +, concave lines as -, and occluding lines with an arrow directed such that the occluding face is on the right and the occluded face on the left when following the direction of the arrow.

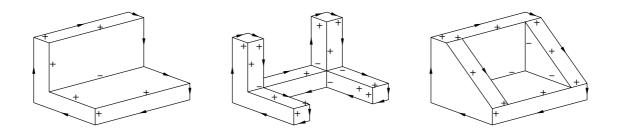


Figure 2.3: Labelled Line Drawings

In text descriptions it is more convenient to have an alphanumeric label: a for an arriving arrow (the occluding face is anticlockwise from the edge), b for a departing arrow (the occluding face is clockwise from the edge), c for a convex line and d for a concave line. In junction labels, these are read clockwise; in the cases of L-, W- and M-junctions, the leftmost line when all lines point upwards is the first; in the cases of T- and K-junctions, the leftmost line when the straight line through the

junction is uppermost is the first; in the cases of Y- and X- and Z-junctions, the ordering is arbitrary. For example, the legal junction labels in a trihedral drawing are Yccc, Yddd, Yabd (Ybda, Ydab), Wbca, Wdcd, Wcdc, Lac, Lab, Lcb, Lbd, Lba, Lda, Tbaa, Tbab, Tbac and Tbad.

Vertex types are also classified using this scheme: extended trihedral vertices are Z-type; tetrahedral vertices are X-type, M-type or K-type, depending on their appearance when all four faces are visible in the drawing. (The illustrative solids in Appendix E.2.18 and E.2.19 do not fit this scheme, as there is no viewpoint from which all four faces at the tetrahedral vertex are visible. They are termed K^* -type vertices and are included with the K-type vertices: they were obtained as part of the process which produced the K-type vertices. The illustrative solids are built from the same blocks as those in the immediately preceding sections, and, as with the K-type vertices, two of the incident edges are collinear.)

2.4.2 Junction Type Identification

This subcomponent labels each junction as L, W, Y, T, M, X, K or Z (see Figure 2.2). The junction type can be identified purely by considering the number and relative angles of lines meeting at each junction, so requires no information other than the vertex-edge graph and the vertex x and y coordinates. This preliminary labelling simplifies implementation of region identification and should ideally precede it. T-junctions must also be labelled as such before the outline of the object can be identified with confidence. See [178] for a description of the algorithm implemented.

The required inputs are: a list of junctions, with 2D coordinates; a list of lines, with the junctions they join; a list of intercepts, relating T-junctions and K-junctions to the lines they intercept.

There are no optional inputs.

The primary output is a junction type for each junction.

As a secondary output, for each junction, a list of the lines meeting at this junction is produced. This list is ordered clockwise.

2.4.3 Line Labelling

Line labelling attempts to identify each line in a line drawing as either convex, concave or occluding. Figures 2.4 and 2.5 show labelled line drawings.

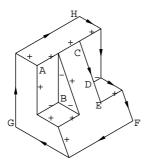


Figure 2.4: Labelled Drawing (Object B.503)

Figure 2.5: Labelled Drawing (Object B.518)

Successful labelling provides useful information about the object drawn. Firstly, the line labels indicate which edges bound the visible faces or partial faces of the object and which merely occlude them. In Figure 2.4, it is evident that two of the internal regions of the drawing correspond to faces, a part of which in each case is hidden by the occluding line, and the other regions correspond to fully-visible faces.

Secondly, the junction labels can be used to obtain a depth ordering of visible vertices [172]. It is, for example, immediately apparent in Figure 2.4 that the Y-junction A is nearer to the viewer than its neighbours, and the Y-junction B further away than its neighbours.

Thirdly, the underlying vertex types implied by the junction labels constrain the possibilities when attempting to reconstruct the hidden topology of the object. In the example of Figure 2.4, the tetrahedral junction catalogue [175] requires that a single concave line must be added to complete the vertex at T-junction C, and that this line is occluded by the occluding line \rightarrow . Deducing that this meets the concave edge occluded by the T-junction D to form a quadrilateral face is straightforward.

The four L-junctions (E, F, G, H) each require at least one more edge to complete the vertex (there could be more); these edges are convex at F, G and H and concave at E. The simplest (and best) interpretation of the drawing can be obtained by using the same methods of deduction that were used for completing the

T-junctions.

In addition to providing a strong suggestion of the correct topological interpretation, the labelling gives clues about the object's geometry—nothing in the labelling in Figure 2.4 contradicts the idea that there is a mirror plane bisecting the octagonal face. If the assumption is added that parallelograms in the drawing are rectangular faces of the object, the geometry is effectively determined.

Interpretation of more complex drawings, such as Figure 2.5, is naturally less straightforward, but it remains evident that these would be much harder to interpret without the several clues provided by labelling.

The required inputs to line labelling are the inputs to and outputs from junction type identification and the loop of lines forming the object boundary—if the trihedral catalogue is used, most trihedral drawings can be labelled consistently and uniquely given only the assumption that every line on the object boundary occludes the background.

Secondary inputs to line labelling are lists of candidate features and cofacial configurations, as described in Section 2.8. It will be seen in Chapter 4 that some of the heuristics useful for choosing between alternative valid labellings require knowledge of features which might be present in the drawing.

The primary outputs of line labelling are a junction label for each junction and a line label for each line. Some drawings have no valid labelling; in such cases, the only output is an error message to the user.

The secondary outputs of line labelling are "runner-up" labellings, which could be used as alternatives if it proves impossible to reconstruct an object on the basis of the first chosen labelling; and merit figures (see Chapter 3.2) for each labelling.

2.5 Components: Subgraphs and Regions

Two minor tasks assist in analysing the front of the object: subgraph identification and region identification. They are outlined here. The techniques for performing these tasks are straightforward, and are not further considered in later chapters. Subgraph identification and region identification take low-order polynomial time, with small constants.

2.5.1 Subgraph Identification

A subgraph is a subset of a drawing in which all junctions correspond to vertices and in which a path exists between any pair of vertices in the subgraph which does not involve jumping from the occluded to the occluding line (or vice versa) at an occluding T-junction. Subgraphs are a helpful clue in detecting hole loops (Chapter 6), although other considerations must also be taken into account.

Figure 2.6 contains a single subgraph. Figure 2.7 contains two subgraphs, and it is the presence of the second subgraph which provides an initial clue that the drawing contains a hole loop (in this case, the mouth of a pocket). Figure 2.8 also contains two subgraphs, but there are no hole loops present; the presence of a second subgraph merely gives warning that inflation (Chapter 7) may require extra care.

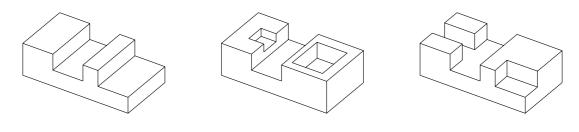


Figure 2.6: Figure 2.7: Figure 2.8:

Identification of subgraphs is trivial for trihedral objects, since all T-junctions are occluding (the algorithm is given in [178]). For non-trihedral objects, some T-junctions are occluding T-junctions but others are projections of K-vertices. Which are which is not always obvious—see for example Figure B.318, where there are two sensible labellings, one in which the rear chimney reaches the top of the roof (in which case there is only one subgraph) and one in which it does not (in which case there are two subgraphs).

Since some of the heuristics used in labelling lines require knowledge of the (potential) number of subgraphs present, it seems preferable to perform an initial subgraph count before line labelling, in order that these heuristics are still available. A revised subgraph count following labelling may be (and in the example given, is) necessary; in view of the simplicity of the task, this additional overhead is acceptable.

The required inputs to subgraph identification are the inputs to junction type identification plus the output from junction type identification (the junction type for each vertex).

A preferred input to subgraph identification is a labelling. If this is unavailable, it is assumed that all *T*-junctions are occluding.

The primary output from subgraph identification is a subgraph label for each junction and each line in the drawing. There are no secondary outputs.

2.5.2 Region Identification

Region identification divides the drawing into regions bounded by closed loops of lines, as a preliminary stage in identifying the visible faces of the object. It starts by enumerating all half-edges in the drawing. Lines have two half edges, plus one for each T-junction which intersects the line (occluding and non-occluding T-junctions need not be distinguished)—for example, the line marked * in Figure 2.9 contributes fourteen half-edges. Region identification then creates regions, one at a time, allocating a clockwise loop of half-edges to a region, until no unused half-edges remain. The "background region" is identified using the same technique.

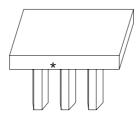


Figure 2.9:

The required input to region identification is the original line drawing data.

Preferred inputs are: a clockwise-ordered list of lines at each junction (this simplifies implementation); a labelling, which makes it possible to distinguish those lines and junctions which correspond to edges and vertices of the face corresponding to the region from those lines and junctions which merely occlude that face.

The primary outputs are a list of regions (a region is a cyclically-ordered list of lines and junctions), and cross-references between junctions, lines and regions.

The secondary output, for which a labelling is required, is an indication for each line and junction in each region whether that line or junction bounds or occludes the corresponding face.

In RIBALD, region identification is split into two. Enumeration of half-edges and identification of the background region, which need no additional information,

precede line labelling. Identification of other regions of the drawing can make use of labelling information and follows line labelling.

2.6 Component: Parallel Lines

In order to make hypotheses based on the symmetries and regularities of a drawing, it is useful to be able to identify which lines in the drawing are believed to correspond to edges which are parallel in 3D, and to be able to assign an indication of confidence in this belief. Chapter 5 describes a method for partitioning lines into bundles, where any pair of lines in the same bundle is probably intended to correspond to parallel edges in 3D.

The discussion in Chapter 5 concludes that the bundling process is $O(n^2)$ in theory.

The primary inputs are: the original drawing data, plus knowledge of the underlying vertex type of each junction (produced by line labelling).

One preferred input, if used, improves the robustness of the output: the output of common feature identification (Section 2.8).

The primary output from the component is a list of bundles, as described above.

The secondary outputs from the component are: the mean 2D angle of the bundle to which the line has been allocated; an indication of which three of the bundles (if any) are most likely to be aligned with the 3D axes.

2.7 Component: Two-Dimensional Tidying

Sugihara's method [159] for determining whether a labelling can be realised geometrically suggests a mechanism for identifying and correcting misplaced vertices [160]. Although the initial outline of this method assumed that the object portrayed is a trihedral polyhedron, the formal proof [162] does not rely on this assumption and is valid for any general-viewpoint line drawing. This has not been investigated, as this thesis prefers to leave the user's input unchanged until a final geometric realisation is produced for the entire reconstructed object.

After identifying intended 2D line parallelism, Grimstead [38] attempted to adjust the drawing to improve it. The purpose was to enhance this regularity in order

to make other artefacts which depend on parallelism easier to detect.

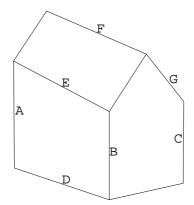


Figure 2.10: House

I do not recommend 2D drawing tidying, and RIBALD does not include it, for two reasons. Firstly, as noted in Chapter 1.1, any distortion by intermediate processes of the user's input is undesirable. Secondly, regularity may be lost, not gained, by two-dimensional tidying. For example, consider Figure 2.10. Even if it can be correctly determined that edges D, E and F should be parallel, problems can occur. Suppose that lines A, B and C are well-drawn, being both parallel and of matching lengths, but that line F is misdrawn. A least-squares fit, requiring D, E and F to be parallel while trying to keep A, B and C parallel too and to preserve existing vertex locations, will change the orientations of lines D and E, and therefore change the lengths and possibly the orientations of A and B, while C remains unaffected.

2.8 Component: Feature Identification

Certain local configurations of lines in a drawing have a natural (to a human) interpretation—it is not necessary to go through a complex process of reasoning to see, for example, slots in Figure B.549 or holes in Figure B.420. Any such features in an object which can be identified in advance from the drawing will simplify the process of topological reconstruction. Although this is a departure from the ideal of "reconstruction rather than recognition", some features occur so often in engineering components that this can be justified. It is also possible that some features might be so regularly misinterpreted by the topological reconstruction process that identifying them in advance is necessary.

The methods of identifying features discussed in Chapter 6 take $O(n^3)$ time.

The required input is a list of junction types.

Preferred inputs which, if used, may improve the reliability of the output are: the output of line-labelling, and a list of groups of bundles of probably-parallel lines.

The primary output from the feature identification component is a list of candidate features.

The secondary output from the component is a figure of merit for each candidate feature.

It can be noted that there is potential for mutual dependence here. Some of the heuristics for identifying preferred labellings make use of identified features, but identification of features is more reliable if it follows line labelling. Identification of features is also more reliable after bundles of parallel lines have been identified. A resolution to this problem is suggested in Chapter 6.

2.9 Component: Inflation

Calculation of final spatial locations of vertices must take account of the entire object structure if it is to consider symmetry and regularities, so must be deferred until after the topology of the hidden part of the object has been deduced. However, intermediate stages of processing may find provisional depth coordinates for the visible part of the object useful or necessary when assessing the merits of their hypotheses. The geometry produced need not be particularly accurate, but it is desirable that the depth-ordering of the visible vertices is correct, and necessary that adjacent vertices are ordered correctly.

Chapter 7 describes an approach for producing such provisional depth values which is straightforward and intuitively plausible. It creates and solves a system of equations linear in vertex depth coordinates. In the simplest version of this approach, the number of equations generated is O(e) for a drawing with e edges. Assuming that black-box linear system solvers are $O(nu^2)$ for a system of n equations and u unknowns, and that u is also O(e), the method has an overall performance of $O(e^3)$. To maintain this performance for more complicated variants of the approach, the number of equations must be limited to O(e) for large drawings.

The required inputs to the inflation component are: a list of start and end

junctions for each line in the drawing, a line label for each line in the drawing, and a junction label for each junction in the drawing.

Preferred inputs which, if used, may improve the quality of the output are: a list of which visible vertices lie on each visible face, and a list of "bundles" of probably-parallel lines.

The primary output from the inflation component is a depth (z-) coordinate estimate for each visible vertex.

The secondary outputs from the component are: a z-coordinate estimate for the point at which each partially-occluded edge disappears from view at an occluding T-junction; a 3D vector for each bundle of parallel lines, coordinates for the centre of each face, and a 3D normal vector for each face.

Since the approach requires line and junction labels, this component must follow line labelling. Additionally, if edges are to be made parallel in 3D where lines are bundled together (this is recommended), inflation must follow bundling. Since cofacial loops must be made (approximately) coplanar, inflation must follow identification of face loops and identification of configurations indicating cofacial loops.

2.10 Component: Validation of Labelling

It can happen that labellings which are valid according to the junction catalogue do not lead to realisable geometry. This occurs, albeit rarely, even in the world of trihedral drawings, where such exceptions are uncommon and well-known: Figure B.146 shows Sugihara's Box [163] and Figure B.148 shows Huffman's Combs [56], both of which can be labelled using the Clowes-Huffman catalogue but cannot be interpreted as polyhedra. Appendix B includes other examples.

Geometric realisation is a greater problem when the non-trihedral catalogue is used. Figure 2.11 is a labelling of Figure B.466 in which all of the junction labels appear in the tetrahedral or Clowes-Huffman catalogues. It is also clearly the wrong labelling—the line marked * should evidently be occluding. Many similar examples could be given. In this particular case, the labelling shown would be rejected by the heuristic of minimising the number of non-trihedral junction labels, used by some but not all of the methods described in Chapter 4.

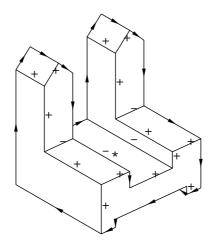


Figure 2.11: Mislabelled Line

An attempted solution to such problems is outlined here. After inflation, approximate unit face normals are estimated for each face. The sum of the two face normals at an edge can be used to determine whether the edge is concave or convex. The algorithm as currently implemented in RIBALD is listed in [178]. Since it uses 3D information, it must follow inflation.

Experimentation suggests that this method is good at choosing the correct interpretation of drawings containing hole loops (for example, it correctly allows Figure 6.2 on page 103 and rejects Figures 6.3), but poor at selecting the correct interpretation of non-trihedral drawings without hole loops. It is particularly poor for drawings such as the Archimedean solids, where there are numerous edges where the two faces are only just non-coplanar (this poor performance probably results from the poor quality of depth estimates available for such drawings). In view of its strengths and weaknesses, RIBALD only uses this method for drawings with more than one subgraph.

No recovery action has been incorporated in RIBALD—if an error is detected, it is reported and processing aborted. A more practical remedy would be simply to discard the labelling and use another one—this is a strong argument in favour of a labelling method which produces a small set of candidate "good" labellings rather than a single "best" one.

The method outlined here is unsatisfactory, less because of its occasional failures than because of its requirements. Geometric validation of a labelling should follow as soon after labelling as possible (ideally, consideration of geometry would be incorporated in the labelling process itself), but the method outlined here requires the non-trivial intervening stages of parallel line identification and inflation.

An improved geometric validation component should be based on an extension of Draper's "sidedness reasoning" [23, 24], which uses the topology of the drawing, and not on the more traditional dual-space/gradient-space methods [56, 58, 96] which depend on the geometry of the drawing being correct.

2.11 Component: Local Symmetry

It is reasonable that if the user draws an object which is not intended to be symmetrical in any way, the asymmetry will be evident in the drawing. Thus, if the drawing portrays an object which could have a symmetry, any reconstruction which breaks this symmetry is probably incorrect. It has already been noted that nothing in Figure 2.4 (page 21) contradicts the idea that the object is mirror-symmetric, and that topological reconstruction should proceed on this assumption. Similarly, topological reconstruction from Figure B.449 should give preference to hypotheses which preserve the mirror symmetry implied by the drawing.

Since the drawing does not show the entire topology, it is not possible to detect full (whole-object) symmetries at this stage. Instead, local symmetries are sought: clues localised to part of the object (a single face, the two faces meeting at an edge, or the faces meeting at a single vertex) from which the presence of whole-object symmetry can be extrapolated. Chapter 8 describes detection of such clues: faces which are rotationally symmetric about their centres or which are mirror-symmetric about a line; edges where the two faces are equivalent under a rotation or reflection; and vertices where all faces are equivalent under rotation. It also describes how face and edge mirror planes are chained across all or part of the drawing when reasoning shows them to be clues to the same global symmetry.

Local symmetry detection as described in Chapter 8 takes O(n) time. Chaining of mirror planes, also described in Chapter 8, takes $O(n^2)$ time. Propagation of symmetry, used in assessing the merit of mirror chains, takes $O(n^4)$ time as implemented in RIBALD (this could in principle be reduced to $O(n^3)$ by adapting Sugihara's recommended solution [161] to a similar problem), so assuming that there are O(n)

candidate mirror chains to assess, the overall process is $O(n^5)$.

The required inputs to the local symmetry detection component are the loops of edges for each face or partial face.

The preferred inputs to this component are a labelling (knowledge of which edges are convex and which are concave is important), bundles of parallel edges (some local symmetries imply that certain edges will be parallel) and the depth coordinate estimates (in order that figures of merit can be based on 3D rather than 2D information).

The primary outputs from this component are: a list for each face of the possible local rotational symmetry seeds about the face centre; a list for each edge of the possible local rotational symmetry seeds at its midpoint; a list on each vertex of the possible local rotational symmetry seeds centred on the vertex; a list for each face of the possible local mirror planes bisecting the face; a list for each edge of the possible local mirror planes running along the edge; a list of the possible chains of mirror planes crossing all or part of the drawing.

The secondary outputs from this component are figures of merit for each of the hypotheses (rotational symmetry, mirror plane or mirror chain) identified.

There are advantages to be gained if symmetry detection were to precede line labelling. For example, in the case of Figure B.318, the single-subgraph interpretation with non-trihedral vertices should be preferred in order to preserve the symmetry of the object. However, the advantages of placing symmetry detection after inflation are greater as parallel edge bundles will be available (a major consideration), and 3D rather than 2D geometric information can be used in assessing figures of merit.

Identification of mirror chains must, of necessity, follow identification of local mirror symmetry. If mirror symmetry or skewed symmetry information is used to refine the estimates of frontal geometry, this refinement must obviously be postponed until after identification of mirror planes.

2.12 Component: Classification

There are advantages to classifying the object portrayed in the drawing into one of several classes of special shape, such as extrusions or normalons—a successful classification considerably improves performance and reliability of both topological

reconstruction (Chapter 10) and final geometric fitting (Chapter 11). Chapter 9 describes methods for performing such a classification.

Classification takes low-order polynomial time, all parts analysed taking O(n) time.

The required inputs for the classification component are the original drawing information plus the outputs of line labelling (Chapter 4), region identification (Section 2.5.2), and bundling (Chapter 5), the last being used in identifying normalons and semi-normalons.

The preferred inputs for this component are the lists of local symmetry elements as identified by the methods of Chapter 8, which increases the choice of heuristics available for estimating figures of merit.

The primary output of this component is a set of special-case shape classes which the drawing matches.

The secondary output of this component is a set of figures of merit, describing how well the drawing matches any special-case class for which it qualifies.

2.13 Component: Topological Reconstruction

The central stage of this thesis, to which most of the previous stages are preparatory, is reconstruction of the hidden topology of the object intended by the user. The aim of this component is to reconstruct the complete vertex/edge framework of the object; the list of face loops need not be complete as an effective and reliable algorithm for adding faces to an existing vertex-edge framework is known (see Section 2.14).

If backtracking is permitted, this component takes exponential time. If backtracking is not permitted, with lookahead used to avoid illegal situations, the component takes $O(n^6)$ time.

The required inputs are the initial user information plus the outputs of the region identification, labelling (Chapter 4) and inflation (Chapter 7) components.

The preferred inputs are the outputs of parallel line bundling (Chapter 5), local symmetry detection (Chapter 8) and object classification (Chapter 9).

The primary output of topological reconstruction is a complete vertex-edge framework: a number of extra vertices; a number of extra edges; and the connectivity information to provide a boundary-representation topological structure. The secondary outputs of topological reconstruction are: new faces, or completions of visible partial faces, and estimated 3D coordinates for each hidden vertex added (these may be even less trustworthy than the provisional depth coordinates of visible vertices, but simplify implementation in that the same data are available for both visible and hidden vertices).

2.14 Component: Face Loops

Adding additional faces to a complete vertex/edge framework until all vertices have the same number of faces as edges is straightforward and is not described in a separate chapter. This completes the topology of the object. A known algorithm exists for this:

- Repeat, while any vertex is connected to fewer faces than edges
 - Choose an unused half-edge (see note 1)
 - Find the least expensive loop of unused half-edges including the chosen half-edge (see note 2)
 - Create a new face using this loop of half-edges

Note 1: when possible, RIBALD uses as the starting-point a half-edge for which the co-half-edge is already part of a face loop, in order to minimise the possibility of choosing an incorrect loop of edges. In practice, this avoids the internal face problem noted by Bagali and Waggenspack [1].

Note 2: RIBALD uses Dijkstra's algorithm [21]; any algorithm for minimum-cost cyclic paths will do. It is, however, important to choose a good cost function. An earlier version of RIBALD [173] used a cost function of 1 for each edge in the loop; this is inadequate, as shown below. Bagali and Waggenspack [1] outline a similar algorithm for identifying the faces of a trihedral wireframe object; their cost function is based on line lengths (shorter lines are preferred)—this suffers from similar problems, albeit less frequently.

The inadequacy of this approach can be illustrated by the completed framework of Figure B.103 shown in Figures 2.12 and 2.13. A number of face loops must be added to complete the topology—possibly one pentagonal hidden face and two

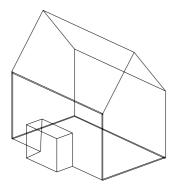


Figure 2.12: Wrong

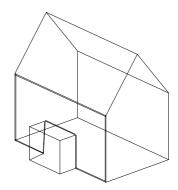


Figure 2.13: Right

quadrilateral hidden faces at the back of the object (these may or may not have been created as part of the topological reconstruction process), and the base face, the small hidden face at the back of the extension built onto the front of the house, and the front wall of the house (the original visible partial face has been deleted). In forming the loop for the last of these, with all edges costing the same, RIBALD finds the topologically-shorter path of unused half-edges shown in Figure 2.12 rather than the correct path shown in Figure 2.13. Bagali and Waggenspack's cost function would choose the correct path in this case, but it is easy to visualise that with a longer, thinner house, it too would fail.

Instead, a cost function is required which weights choice of half-edges geometrically, preferring half-edges which are as close as possible to the plane of half-edges preceding it in the path. After experimenting with a number of similar cost functions, I chose as the most promising the figure of merit for perpendicularity between the half-edge BA and the normal to the plane BCD where BA is the new half-edge being added to the path and CB and DC are the two preceding half-edges in the path.

Using this cost function, detection of face loops is fast $(O(n^2))$ time, and reliable when the output of topological reconstruction is correct. As will be seen in Chapter 10, topological reconstruction sometimes produces vertex-edge frameworks for which no valid geometry exists; face loop detection does not identify these erroneous frameworks and can produce peculiar results when processing them.

Other algorithms for detection of face loops are known. One such is that of Liu and Lee [93], which detects faces directly from a 2D wireframe drawing. This improves on a previous method of Shpitalni and Lipson [149]. In both cases, depth

information is not available and therefore cannot affect the results. The algorithm used in the current version of RIBALD is to be preferred as taking account of provisional depth information.

An alternative idea, worthy of investigation, arises from recent work by Grosjean et al [40], who show that adding or deleting faces to maintain a valid solid model as edges are added to an object is very fast. On this basis, partial faces and the background region would initially be treated as genuine, albeit non-planar, faces of the object; extra vertices and edges would be added, and existing faces split, until all faces of the object are planar. This would be incorporated within topological reconstruction, removing the need for a separate loop completion component at the cost of added complexity in the (already complex) topological reconstruction component. Their method, as described, requires accurate geometry, but appears easy to adapt to use provisional, potentially inaccurate geometry. This idea has not been investigated.

2.15 Component: Geometric Finishing

Although topological reconstruction produces a complete topology, the vertex locations will not be accurate. A final geometric fitting process is required in order to ensure, firstly, that the object has a geometric realisation (vertices must lie on faces, which must be planar) and, secondly, that wherever desirable, identified symmetry constraints are enforced geometrically. Solutions to this problem are described in Chapter 11. The preferred general-case solution takes $O(n^6)$ time.

The required inputs to geometric finishing are the depth information derived in Chapter 7 and the completed topology as output by topological reconstruction as described in Chapter 10.

The preferred inputs are bundles of parallel lines (Chapter 5), seeds for local symmetry (Chapter 8) and drawing classification (Chapter 9).

The primary output of geometric finishing is a list, for each vertex, of vertex coordinates. The only secondary outputs are error reports produced to the user when previously-accepted hypotheses are contradictory and cannot be resolved.

2.16 Component: Splitting and Recombination

For several reasons, it may be desirable to split a drawing into pieces, reconstruct the topology (and perhaps fit a geometry) separately for each piece, and finally recombine the pieces:

- a CSG-style approach has been shown to be effective in reconstructing hidden topology [182, 183]
- the correctness of topological reconstruction (Chapter 10) is noticeably poorer with more complicated drawings than with simple drawings
- the methods described in Chapter 10 take exponential-order time—halving the problem size would clearly be beneficial
- the most natural way of processing objects with bosses is as separate objects
- processing an object as two pieces may allow one or both pieces to be processed as a special-case class even though the whole object does not meet the requirements for such a class.

However, splitting is not, in general, useful for trihedral objects with no hole loops. In such objects, the presence of junctions of two concave and one convex edge, such as Wdcd, would be the clue to identifying a natural point of cleavage [51]. Most of the cases for which this idea might be thought useful, including Figures B.101 and B.6, can be interpreted correctly without this idea. The idea was not pursued when only objects without hole loops were considered.

The idea should clearly be incorporated as the preferred means of processing objects with bosses. This being so, it is worth reconsidering its use for objects including junctions of two or more concave edges, and perhaps also worth searching for reliable methods of splitting large objects at a single concave edge (see Figure B.74 for an example where this would prove useful). There has not been time to investigate this.

2.17 Component: Intersecting Faces

The final stage of Grimstead's system [38] is a three-dimensional tidying process in which the x-, y- and z-coordinates of each vertex are recalculated from the equations of the three faces on which it lies.

The equivalent process in RIBALD is included within the geometric finishing component, and is described in Chapter 11.10.

2.18 Component: Quality Control

Various ideas for a final stage of processing which performs some sort of quality check on the completed B-rep model have been considered. By analogy with a human quality inspector, such a component might:

- simply report problems, offering no clue as to their cause or solution (possible problems include vertices which do not lie exactly on the appropriate face planes, visible vertices which have moved unacceptably far from their locations in the line drawing, and hidden vertices, edges and faces which are in locations from which they would be visible)
- tidy up as much as is possible while ignoring problems for which there is no simple fix (for example, if the original drawing implied mirror symmetry and the object has topological but not geometric mirror symmetry, enforce geometric mirror symmetry if it is possible to do so by moving one or two faces, but otherwise do nothing)
- reject the final model, requiring a second attempt (try to identify hypotheses which have proved incorrect, remove them, and re-run the algorithm from the appropriate point).

In principle, such quality control is unnecessary—correct methods should not make mistakes. However, consideration of the drawing in Figure B.421 makes it clear that a final check of some sort may be a practical necessity—the simplest way of determining whether the features are holes or a pockets is to reconstruct the object on the assumption that the features are pockets and then test how thin the bottoms of the pockets are.

When separation (Section 2.16) is investigated, a component which puts the two halves of the object back together again will be required. Since this component would have to perform some geometric and topological validation, performing other checks at the same time would not be out of place.

Quality control has not been investigated. RIBALD only checks for two problems, vertices not lying on the appropriate face planes, and visible vertices which have moved significantly from their original coordinates, and reports these problems rather than attempting to fix them.

Identifying which hypotheses caused the problem is non-trivial. Further research is needed to determine to what extent this is possible. There has not been time in the course of the work for this thesis to do this.

2.19 Chosen Components: Order and Control Structure

A control structure is required in order to assemble the available components defined above into a system. A sequential structure has been chosen (iterative structures were rejected for reasons of processing speed, and it will be seen in later chapters, particularly Chapter 10 and Chapter 11, that even the simple sequential structure is not always fast enough in the general case). The idea of "lazy evaluation" was rejected as adding unnecessary complexity—it makes the system less predictable and increases development time (and it will also be seen in later chapters, particularly Chapter 11, that even with the simpler sequential control structure, some ideas remain untested).

The following sequential control structure appears to match outputs of earlier components to inputs of later ones as well as any:

- Analyse the line drawing:
 - Identify Junction Types (see Section 2.4.2)
 - Identify Subgraphs (see Section 2.5.1)
 - Identify the Background Loop (see Section 2.5.2)
 - Identify Candidate Features (see Chapter 6)

- Label the Lines and Junctions (see Chapter 4)
- Identify Subgraphs again (see Section 2.5.1)¹
- Identify the Face Loops (see Section 2.5.2)
- Identify Parallel Lines (see Chapter 5)
- Identify Genuine Features from Candidates (see Chapter 6)
- Inflate the 2D Coordinates (see Chapter 7)
- Identify Rotational Symmetry Elements (see Chapter 8)
- Identify Mirror Planes and Mirror Chains (see Chapter 8)
- Classify the Object (see Chapter 9)
- Reconstruct the hidden topology framework (see Chapter 10)
- Fill in any missing faces (see Section 2.14)
- Beautify the resulting object (see Chapter 11)
- Perform a quality check (see Section 2.18)

Geometric validation (Section 2.10) should follow immediately after (or be part of) line labelling; the current component, placed after inflation, is unsatisfactory.

Splitting objects into pieces and later recombination of the pieces (Section 2.16) has not been investigated in sufficient depth to identify the points at which these should be included.

 $^{^{1}}$ Labelling may identify some T-junctions as non-occluding, and all lines at a non-occluding T-junction must be in the same subgraph

Chapter 3

Background Ideas

This section lists ideas, algorithms and formulae obtained from fields other than interpretation of line drawings which are used in this thesis or in the RIBALD program. None of these are new—where no citation is given, it indicates that the idea is either well-known or obvious.

3.1 Sketch to Drawing

The distinction between a *sketch* and a *drawing* was made in Chapter 1.1. Conversion of sketches to drawings is well-covered in the literature.

Jenkins [59] does not interpret sketches, but describes a 2D sketch input and tidying package. Symmetry is detected automatically, but parallelism and other constraints are entered via menu options. Close points and tiny lines are removed automatically.

Mitani [112] has produced a freehand sketching program, JMSketch, with the capability of interpreting sketches as line drawings. It was this program which was used to draw Figures 1.3 and 1.4 (page 4) in Chapter 1.1 and convert them to the line drawings in Figures 1.5 and 1.6.

The program of Pavlidis and Wyk [122] performs 2D "beautification" of a drawing (which may include points as well as lines) by enforcing constraints. The constraints which may be enforced are: equality of side lengths; equality of side slopes; collinearity of sides; and vertical and horizontal alignment of points. They stress the

importance of negative constraints, which they describe as necessary to avoid various pitfalls. For example, if two lines cross, a negative constraint may be required in order that their slopes remain different however close they may be. They use a simple equation solver developed by one of the authors.

Eggli et al [25] also have a good, full-featured 2D sketching interface. They discuss user preferences in some detail—they have established different user settings, which weight constraints differently, according to the profession and skill of the user. They consider that "Interpreting an arbitrary 2D input as a 3D object is too ambiguous, in general". They limit 3D features to specific menu options, the only current one being extrusions, although freehand curves, as well as polygons, may be extruded, and 2D input may be extruded along freehand curves as well as along lines. Features can be added on faces of existing objects. Holes can be made in existing objects.

In the remainder of this thesis, it is assumed that converting sketches to line drawings is a proven technique.

3.2 Searching and Heuristics

In many parts of this thesis, it will be necessary to choose between alternatives. Levy [87] discusses this as two distinct subproblems: position evaluation and search strategy.

3.2.1 Position Evaluation

Position Evaluation is the process of determining the merit of a position, a static situation. Any choice between two or more positions will require that the competing merits of the positions are assessed numerically.

In considering interpretation systems, Hinton [49] argues in favour of "tentative hypotheses", where a number of competing hypotheses are maintained. The alternatives he rejects are: "hypothesise and test", in which hypotheses made on the basis of local cues are tested against the sketch as a whole and accepted or rejected immediately; the "principle of least commitment", which requires a large number of vague classifications; and less convincingly against "feature semantics"—he accepts

that propagation of constraints generated by interdependent local cues can in practice be more efficient than "tentative hypotheses", citing line labelling as an obvious example (the method of inflation described below is another). Each "tentative hypothesis" is assigned a numerical value; Hinton calls these "probabilities" and uses the range 0 to 1. The idea of competing and complementing hypotheses has been adopted in this thesis, as has the convention of assigning numerical values in this range, but the term figure of merit is preferred as the actual numerical values are subjective rather than statistical—it is a numerical value intended to represent the plausibility that the artefact was intentional on the part of the user.

Numerically, figures of merit are manipulated as probabilities:

- a hypothesis with a figure of merit F = 1.0 can be accepted immediately
- a hypothesis with a figure of merit of F = 0.0 can be rejected immediately
- multiplying two figures of merit $F_{A \cap B} = F_A \times F_B$ reduces the merit
- mutual reinforcing of two figures of merit $F_{A \cup B} = 1.0 (1.0 F_A) \times (1.0 F_B)$ increases the merit

3.2.2 Search Strategy

Except where the number of alternatives to be considered is small, a *search strategy* is required: a method of finding those static situations whose merits are worthy of assessment. Formally, the static situations are the terminal nodes of a *directed acyclic graph*; however, less formal and more intuitive descriptions help to clarify the problem.

It is conventional, if occasionally misleading, to use arboreal terms to describe a search space: the starting position is the *root*, alternatives (when alternatives become possible) are *branches*, the various terminating static situations to be evaluated are *leaves*, and the overall structure is a *tree*. The metaphor becomes misleading when it is possible to reach a leaf or branching-point via two or more sequences of branches (such a structure is clearly not a tree), and in Chapter 10 this is often the case.

An alternative metaphor is that of a single-player game. As already seen, positions in the game are nodes of a graph. The possible moves in the game in any

position are the arcs of the graph leaving the corresponding node, and the *rules* of the game determine which arcs exist.

In general, this thesis will use the tree metaphor.

One simple method of choosing a leaf is the *greedy algorithm*. Whenever a branching-point is reached, the merits of the available branches are assessed as if they were static situations, and the branch with the highest merit is followed. Provided only that each branch ends either at a leaf or by dividing into "thinner" branches, the *greedy algorithm* guarantees that a solution will be found, and it is likely to be a good one (although there can be no guarantee that it will be the best).

If some branches are dead ends (as happens in Chapter 10), the strategy must be refined: whenever a dead end is reached, one returns to the last branching-point at which there was a branch which has not yet been explored, and follows this instead. This strategy is known as *backtracking*, and it will always find a solution if there is one, provided only that there are no loops in the structure (branching-points divide only into "thinner" branches, never back into "thicker" ones).

There are numerous variants on these two methods described in standard text-books on searching, e.g. [123]. In general (and in Chapter 10 in particular), searches are conducted either using a *greedy algorithm* or with *backtracking*.

One alternative considered was Stilman's [155] idea of partitioning a starting situation into *subgames*, and determining subgoals for each subgame. This idea is suspect even in the context of chess endgames, for which it was originally developed. Stiller [154] points out that with very few exceptions the chessboard is "small"—it cannot usually be partitioned into subgames, as moves made in one subgame affect other subgames. The "games" of three-dimensional topology and geometry can be considered even smaller—changing the topology or geometry of one atom of an object will inevitably have consequences elsewhere in the object—and the idea of partitioning into subgoals can be rejected.

Other ideas from the chess literature could usefully be investigated in the context of search strategies for Chapter 10, but for reasons of time have not been taken further in this thesis. Levy [87] considers:

• The *Killer Heuristic*: if a particular change is found to have the highest merit in one branch of the search tree, that change should be tried first when traversing

other branches of the tree;

- Forward Pruning: a branch of the search tree is "obviously" wrong and need not be traversed;
- Razoring: a variant of forward pruning where a branch of the search tree is rejected because the change initiating that branch lowers the overall merit;
- Transposition Tables: after evaluating in full a branch of the search tree, the results are assigned to intermediate nodes in the branch, so that if the same node occurs by transposition in another branch, it need not be re-investigated.

3.3 Constraints and Optimisation

A constraint is a relationship between variables expressed as a function (an equation or inequality) of those variables [80]; constraints may be discrete or continuous. A constraint satisfaction problem (CSP) attempts to find values for the variables which provide a solution to a system of constraints. Both discrete and continuous constraint satisfaction problems occur in the approaches considered in this thesis: line labelling (Chapter 4) is a discrete CSP, and fitting a geometry to a given topology (Chapter 11) is a continuous CSP.

As well as general solutions, there are specific solutions to the line-labelling problem; these are discussed in Chapter 4. Of the more general solutions to discrete CSPs, Mackworth [98] recommends node consistency, arc consistency and path consistency and deprecates backtracking, whereas Kumar's survey [76] recommends backtracking and arc consistency.

Node consistency, arc consistency and path consistency can be illustrated by reference to the junction labelling problem described in Chapter 4. In this problem, each junction must have a label selected from a limited number of available labels (each *node* must satisfy a unary predicate) and each edge must have the same label at both end vertices (each *arc* must satisfy a binary predicate relating two nodes). A path is a more complex predicate relating two or more arcs—path consistency is not used in standard line-labelling algorithms, but the error in Figure 2.11 (page 29) is a path consistency error caused by not satisfying the three-arc predicate described on Page 90.

Backtracking has already been described in Section 3.2.2.

It may be noted that, although both Kumar and Mackworth consider general CSPs, Kumar's discussion is limited to *node* and *arc* consistency, while Mackworth also allows *path* consistency.

Notwithstanding Mackworth [98], the methods described above do not transfer well to solutions to continuous CSPs. Various approaches to the general continuous CSP, and the specific geometric CSP, have been reported in the literature and are summarised in Chapter 11.2. There, I conclude that geometric CSPs are best handled using numerical optimisation approaches.

3.3.1 Downhill Optimisation

RIBALD uses the downhill simplex method, generally known as *amoeba* [117, 131] for the numerical optimisations required by Chapter 11, which cannot be solved as linear systems.

According to Press et al [131], there are no theoretical reasons for preferring either *amoeba*, variable metric algorithms such as BFGS, or conjugate gradient algorithms for a small to medium number of variables (for a large number of variables, conjugate gradient algorithms are preferred). BFGS can work better for functions whose distant behaviour matches their local behaviour, and *amoeba* can work better for functions where the distant behaviour differs markedly from the local behaviour. In principle, any new optimisation process should be tested with all three methods to determine empirically which works best.

However, use of a single downhill optimisation method has practical advantages, and RIBALD uses *amoeba* as (a) I have used it before [170] and found it to be reliable, (b) the algorithm is compact, and (c) the algorithm is based on geometric concepts, and thus meets the preference for methods which are intuitively correct.

3.3.2 Genetic Algorithms

Genetic algorithms provide a non-deterministic method for solving CSPs for which no deterministic algorithm is known, but for which a reliable *position evaluation* function is available. Genetic algorithms were first elaborated by Holland [52],

although (like neural networks) they were prefigured by Selfridge's ideas of Pandemonium [145]. Goldberg [35] introduced refinements such as mutation. Where versions of RIBALD have incorporated genetic algorithms (for example, in testing the ideas of Chapter 11.7), they follow Goldberg's outline algorithm [35, 84].

3.4 Least Squares Fit

RIBALD uses the black-box routine Ortholin2 [3] for all least-squares fitting, and in particular depth estimation (Chapter 7) and fitting planes through more than three points. Initially, two routines were considered, Ortholin2 and SVD [131]. Initial comparisons showed Ortholin2 to be more robust and significantly faster in all realistic cases tested. Ortholin2 was not robust to three specific user errors: insufficient equations, duplicated equations, and unknowns not referred to in any equation, but these are easily avoided.

Ortholin2 comprises two stages, an initial calculation which is $O(nu^2)$ for n equations and u unknowns, and an iterative refinement which is O(nu) for each iteration.

3.5 Planar Geometry

The area A(a, b, c) of a parallelogram three of whose corners are the points (a_x, a_y) , (b_x, b_y) , (c_x, c_y) :

$$A(a, b, c) = (a_x(b_y - c_y) + b_x(c_y - a_y) + c_x(a_y - b_y)).$$

Although obvious (most basic geometry books give the area of a triangle, $\frac{1}{2}A(a,b,c)$), the area function A(a,b,c) must be defined as it is used on Page 49.

2D Lines
$$(x,y) \cdot \hat{\mathbf{n}}_{\mathbf{A}} + d_A = 0$$
 and $(x,y) \cdot \hat{\mathbf{n}}_{\mathbf{B}} + d_B = 0$ cross at

$$\frac{(n_{By}d_B - d_A n_{Ay}), (d_A n_{Bx} - n_{Ax} d_B)}{n_{Ax} n_{By} - n_{By} n_{Bx}}$$

If the denominator is tiny, the lines are parallel or collinear. RIBALD uses this method throughout in order that parallelism/collinearity can be detected easily.

3.6 Solid Geometry

Results from solid geometry are used throughout the thesis.

3.6.1 Euler's Formula

RIBALD uses the Poincaré version of Euler's formula to validate completed topologies (Chapter 10):

$$V + F - E = L + 2N - 2H$$

V is the number of vertices, E is the number of edges, F is the number of faces, E is the number of hole loops, E is the number of through holes, and E is the number of objects.

3.6.2 Vectors

Although vectors were introduced over a century ago by Gibbs [34] and Heaviside, their notation remains inconsistent. This thesis follows Weatherburn [185]: a vector \mathbf{a} from the origin to point A has components (A_x, A_y, A_z) , its modulus is a, and a unit vector in the same direction is $\hat{\mathbf{a}}$. The scalar product of two vectors is $\mathbf{a} \cdot \mathbf{b}$, the vector or area product of two vectors is $\mathbf{a} \times \mathbf{b}$, and the volume product of three vectors is $[\mathbf{abc}]$. Except where specifically noted otherwise, vectors are 3D.

By extension, for normalising a vector expression (\mathbf{v}) , this thesis uses the operator notation $\hat{\mathbf{v}}$. There seems to be no established notation for the common operation "choose either \mathbf{i} or $-\mathbf{i}$, whichever is closer to \mathbf{n} ". In this thesis, the notation $(\mathbf{i}) \hookrightarrow \mathbf{n}$ is used.

The nearest vector ${\bf n}$ to a reference vector ${\bf r}$ which is perpendicular to ${\bf p}$ is given by

$$\mathbf{n} = (\mathbf{p} \times \mathbf{r}) \times \mathbf{p}$$

The nearest two mutually perpendicular vectors \mathbf{i}' and \mathbf{j}' , both lying in a plane perpendicular to $\hat{\mathbf{n}}$, to two non-collinear vectors \mathbf{i} and \mathbf{j} , are obtained by setting

$$\mathbf{i} = (\mathbf{i} \times \hat{\mathbf{n}}) \times \mathbf{i}$$

$$\mathbf{j} = (\mathbf{j} \times \hat{\mathbf{n}}) \times \mathbf{j}$$

and then iterating

$$\mathbf{p} = (\mathbf{i} \times \hat{\mathbf{n}}) \hookrightarrow \mathbf{i}$$

$$\mathbf{q} = (\mathbf{j} \times \hat{\mathbf{n}}) \hookrightarrow \mathbf{j}$$

$$\mathbf{i}' = \hat{\ } (\mathbf{i} + \hat{\mathbf{p}})$$

$$\mathbf{j}' = \hat{\ } (\mathbf{j} + \hat{\mathbf{q}}).$$

Note that iteration is required in order to allow for the possibility that the original vectors **i** and **j** do not lie in the plane. The method converges quickly when **i** and **j** are close to the plane—RIBALD always uses two iterations, which is adequate for demonstration purposes (more may be required in some cases to achieve the accuracy required by CAD).

The nearest three perpendicular vectors \mathbf{i}' , \mathbf{j}' and \mathbf{k}' to three non-coplanar vectors \mathbf{i} , \mathbf{j} and \mathbf{k} are obtained by iterating

$$\mathbf{p} = (\mathbf{j} \times \mathbf{k}) \hookrightarrow \mathbf{i}$$

$$\mathbf{q} = (\mathbf{k} \times \mathbf{i}) \hookrightarrow \mathbf{j}$$

$$\mathbf{r} = (\mathbf{i} \times \mathbf{j}) \hookrightarrow \mathbf{k}$$

$$\mathbf{i}' = \hat{}(\mathbf{i} + \hat{\mathbf{p}})$$

$$\mathbf{j}' = \hat{}(\mathbf{j} + \hat{\mathbf{q}})$$

$$\mathbf{k}' = \hat{}(\mathbf{k} + \hat{\mathbf{r}}).$$

RIBALD always uses four iterations (avoiding the overhead of testing for convergence).

3.6.3 Planes

RIBALD stores a plane **P** as normal and distance: $r \cdot \hat{\mathbf{n}}_{\mathbf{P}} + d_P = 0$.

Given three points A, B and C, the plane \mathbf{P} through them is found by setting $\hat{\mathbf{n}}_{\mathbf{P}} = \hat{\mathbf{n}}((\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a}))$ and $d_P = -\mathbf{a} \cdot \hat{\mathbf{n}}_{\mathbf{P}}$.

Fitting a plane through four or more points uses a least-squares fit, as described in Section 3.4.

The point of intersection of three planes, \mathbf{P} , \mathbf{Q} and \mathbf{R} , each defined by a normalised normal vector $(\hat{\mathbf{n}}_{\mathbf{P}}, \hat{\mathbf{n}}_{\mathbf{Q}}, \hat{\mathbf{n}}_{\mathbf{R}})$ and a distance (d_P, d_Q, d_R) , is obtained using Cramer's Rule: set a vector $\mathbf{d} = (d_P, d_Q, d_R)$ and then calculate

$$\mathbf{p_{PQR}} = \frac{\left(\left[\mathbf{d}\hat{\mathbf{n}}_{\mathbf{Q}}\hat{\mathbf{n}}_{\mathbf{R}}\right], \left[\hat{\mathbf{n}}_{\mathbf{P}}\mathbf{d}\hat{\mathbf{n}}_{\mathbf{R}}\right], \left[\hat{\mathbf{n}}_{\mathbf{P}}\hat{\mathbf{n}}_{\mathbf{Q}}\mathbf{d}\right]\right)}{\left[\hat{\mathbf{n}}_{\mathbf{P}}\hat{\mathbf{n}}_{\mathbf{Q}}\hat{\mathbf{n}}_{\mathbf{R}}\right]}$$

If the bottom volume product is zero, two or more of the planes are parallel.

The intersection point of four or more planes is implemented as intersection point of the three planes whose normalised normals have the largest volume product.

The perpendicular distance r from a point Q to a plane \mathbf{P} is given by [5]:

$$r = \mathbf{q} \cdot \hat{\mathbf{n}}_{\mathbf{P}} + d_P$$

A 2D point $D = (D_x, D_y, 0)$ can be made coplanar with the plane through three 3D points U, V and W (using the area function defined in Section 3.5 above) as follows:

- Set $x = \frac{A(U,D,W)}{A(U,V,W)}$
- Set $y = \frac{A(U,V,D)}{A(U,V,W)}$
- $\bullet \ \operatorname{Set} \ D_z = (1 x y)U_z + xV_z + yW_z$

3.6.4 3D Lines

The nearest point P on a line $\mathbf{u} + s\hat{\mathbf{v}}$ to a general point G is given by:

$$\mathbf{p} = \mathbf{u} + (\hat{\mathbf{v}}.(\mathbf{g} - \mathbf{u}))\hat{\mathbf{v}}$$

The distance r from any point G to the line $\mathbf{u} + s\hat{\mathbf{v}}$ is given by:

$$r = |(\mathbf{u} + (\hat{\mathbf{v}}.(\mathbf{g} - \mathbf{u}))\hat{\mathbf{v}} - \mathbf{g})|$$

The nearest point P on line $\mathbf{a} + u\hat{\mathbf{m}}$ to line $\mathbf{b} + v\hat{\mathbf{n}}$ is calculated by [27]:

- Set c = b a
- Set $J = \hat{\mathbf{m}} \cdot \hat{\mathbf{n}}$
- ullet If $1-J^2$ is tiny, the lines are close to parallel report an error

•
$$\mathbf{p} = \mathbf{b} + \hat{\mathbf{n}} \left((\mathbf{c} \cdot \hat{\mathbf{m}}) - (\mathbf{c} \cdot \hat{\mathbf{n}}) J \right) / (1 - J^2)$$

The shortest perpendicular distance r from line $\mathbf{a} + s\hat{\mathbf{m}}$ to line $\mathbf{b} + t\hat{\mathbf{n}}$ follows from this [27]:

•
$$r = \left| \left(\left(\mathbf{a} + \hat{\mathbf{m}} \left(\left(\mathbf{c} \cdot \hat{\mathbf{n}} \right) - \left(\mathbf{c} \cdot \hat{\mathbf{m}} \right) J \right) / \left(1 - J^2 \right) - \left(\mathbf{b} + \hat{\mathbf{n}} \left(\left(\mathbf{c} \cdot \hat{\mathbf{n}} \right) J - \left(\mathbf{c} \cdot \hat{\mathbf{m}} \right) \right) / \left(1 - J^2 \right) \right) \right|$$

3.6.5 General Rotation about an Axis

Chapter 11.6 requires calculation of the vector $\hat{\mathbf{M}}$ which is obtained when a vector $\hat{\mathbf{N}}$ is rotated by a known angle about an axis $\hat{\mathbf{R}}$. From [5]:

 $\hat{\mathbf{M}} = \Re(\rho, \hat{\mathbf{R}})\hat{\mathbf{N}}$ where \Re is the rotation matrix for rotating through an angle ρ about $\hat{\mathbf{R}}$:

out **R**:
$$\Re(\rho, \hat{\mathbf{R}}) = \begin{pmatrix} R_x^2 + (R_y^2 + R_z^2)c & R_x R_y v - R_z s & R_x R_z v + R_y s \\ R_x R_y v + R_z s & R_y^2 + (R_x^2 + R_z^2)c & R_y R_z v - R_x s \\ R_x R_z v - R_y s & R_y R_z v + R_x s & R_z^2 + (R_x^2 + R_y^2)c \end{pmatrix}$$
where $s = \sin \rho$, $c = \cos \rho$ and $v = 1 - \cos \rho$.

3.6.6 Spherical Triangles

Chapter 11.6 requires manipulation of unit face normals around the Gaussian sphere. Such manipulation makes use of various standard results for spherical triangles.

From [95, 186]:

Consider three points A, B and C on the surface of a sphere of unit radius centred at point O. Draw the great arcs BC, CA and AB, such that all arcs are less than π . This divides the surface of the sphere into two; the smaller of the two subdivisions is a *spherical triangle*.

- the angle subtended at O by the arc BC is denoted a;
- the angle subtended at O by the arc CA is denoted b;
- the angle subtended at O by the arc AB is denoted c;
- the angle between the planes OCA and OAB is denoted α ;
- the angle between the planes OAB and OBC is denoted β ;
- the angle between the planes OBC and OCA is denoted γ .

From [186]:

Defining $\hat{\mathbf{l}}$ as the vector \vec{OA} , $\hat{\mathbf{m}}$ as the vector \vec{OB} and $\hat{\mathbf{n}}$ as the vector \vec{OC} , it is known that $\cos a = \hat{\mathbf{m}} \cdot \hat{\mathbf{n}}$, $\cos b = \hat{\mathbf{n}} \cdot \hat{\mathbf{l}}$ and $\cos c = \hat{\mathbf{l}} \cdot \hat{\mathbf{m}}$.

The cosine rule for spherical triangles [186, 95]:

$$\cos a = \cos b \cos c + \sin b \sin c \cos \alpha$$
$$\cos b = \cos c \cos a + \sin c \sin a \cos \beta$$
$$\cos c = \cos a \cos b + \sin a \sin b \cos \gamma$$

to which [95] adds:

$$\cos \alpha = \cos a \sin \beta \sin \gamma - \cos \beta \cos \gamma$$
$$\cos \beta = \cos b \sin \gamma \sin \alpha - \cos \gamma \cos \alpha$$
$$\cos \gamma = \cos c \sin \alpha \sin \beta - \cos \alpha \cos \beta$$

The sine rule for spherical triangles [186, 95]:

$$\frac{\sin \alpha}{\sin a} = \frac{\sin \beta}{\sin b} = \frac{\sin \gamma}{\sin c}$$

[95] adds a further result of use:

$$\sin^2{(a/2)} = (1 - \cos{a})/2$$

[95] also quotes the rule for the area of a spherical triangle:

$$(\alpha + \beta + \gamma - \pi)$$

3.7 Dual Space

Dual Space [57] is a 3-dimensional space (u, v, w) in which a point (x = a, y = b, z = c) in normal 3-dimensional (x, y, z) space maps to the plane -au - by + z + c = c0. The mapping is symmetrical—the plane Px + Qy + z + D maps to the point (u = -P, v = -Q, w = D). The line x = mz + g, y = nz + h maps to the line u = -f(nw + h), v = f(mw + g) where f = 1/(h(m + g) - g(n + h)). All points on the normal space line map to planes which pass through the dual space line, and vice versa.

The coordinates of a vertex map to a dual-space plane. The lines of all edges meeting at the vertex map to dual-space lines which are in this plane. The planes of all faces which intersect that the vertex map to dual-space points which are in this plane [57].

Dual Picture Space [57], also known as Gradient Space [97], is a 2D space obtained by projecting dual space onto the (u, v) plane. One of its more interesting properties is that lines in (x, y) drawing space map to perpendicular lines in dual picture space [57].

3.8 Miscellaneous

Random numbers, when required, are generated using $ran\theta$ from Press et al [131].

Chapter 4

Line Labelling

4.1 Introduction

The technique of line-labelling [14, 43, 56] attempts to identify each line in a line drawing as either convex, concave, or occluding, in order to reduce the possible interpretations of a line drawing to a processable number. The best-established method of labelling line drawings is by means of a catalogue of permissible junction labels—any combination of labelled lines meeting at a junction which does not produce a junction label listed in the catalogue can be rejected. The Clowes-Huffman catalogue for line drawings of trihedral polyhedra is well-established—although the limitation to trihedral vertices is somewhat restrictive, Clowes-Huffman line-labelling has been used successfully in a number of applications, including the interpretation of natural line drawings [147] and of freehand sketches [38]. A similar catalogue for line drawings with hidden lines shown [157] has been used in interpreting such drawings [91, 17]; other catalogues are listed in Section 4.2.

It is not always possible to achieve a unique labelling, but neither is it necessarily desirable [163]—it may be the case that several labellings correspond to sensible interpretations, and all of these should be retained. However, finding a "best" labelling remains useful—for example, while there are several reasonable interpretations of Figure 1.2 (page 2), they all correspond to the same labelling.

Section 4.2 describes the history of line labelling, and indicates where previous work fails to meet the requirements of reconstruction of engineering objects.

In particular, a junction catalogue is required for tetrahedral objects. Section 4.3

describes a method for generating this catalogue, and the catalogue itself. Both the method and the catalogue are new.

Section 4.4 describes two labelling approaches using these catalogues, a deterministic approach which adapts Kanatani's algorithm [64], and a novel probabilistic approach based on relaxation labelling [140]. Both use heuristics to choose between alternative valid labellings, and the deterministic approach also uses heuristics to speed up its search process. These heuristics are also new.

Section 4.5 summarises the results of investigations into these two approaches.

4.2 History

4.2.1 Junction Catalogues

Although attempts to identify the junctions which may appear in valid line drawings date back at least as far as Guzman [43], the first systematic catalogues were produced by Clowes [14] and Huffman [56] (they are identical apart from nomenclature). They assume polyhedra with trihedral vertices. By considering three perpendicular planes intersecting at the origin, and analysing the line drawings produced in all 28 possible combinations of solid and empty regions, they showed that the twelve junction types illustrated in Appendix E.1 are the only possible views of trihedral vertices.

In addition to the twelve junction types obtained by this procedure, Clowes and Huffman also listed the four occlusive T-junctions. These occur when an occluding face occludes an edge, and are artefacts of line drawings, not features of objects. Since they need not correspond to vertices, they are independent of the types of vertex which appear in a sketch, and thus should appear in any junction catalogue for natural line drawings.

The process used to derive the Clowes-Huffman catalogue produces, as a side-effect, the complete catalogue of junctions possible in drawings of extended trihedral objects. By adding the six junction types shown in Figures 4.1–4.6 it is possible to label drawings of any extended trihedral polyhedron [120]. Note that although three planes intersect at the vertices shown, more than three faces meet at the vertex, and also more than three edges meet at the vertex (a single line entering and leaving the

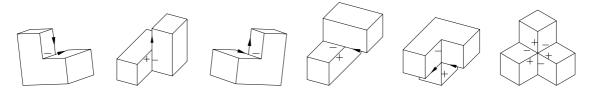


Figure 4.1: Figure 4.2: Figure 4.3: Figure 4.4: Figure 4.5: Figure 4.6: Tbda Tbda Zcbda Zcdcdcd

vertex must be counted as two edges, since such a line may be concave in one part and convex (as in Figures 4.5 and 4.6) or occluding (as in Figures 4.1–4.4) in the other).

As Sugihara [163] points out, given independent knowledge of the world of objects which may appear in line-drawings, a complete catalogue of the junction labellings of that world can in principle be produced. A similar catalogue to the Clowes-Huffman catalogue for wireframe drawings [157] has been used in applications to interpret them [91, 17]. Turner's catalogue [167], which considers curved objects, reportedly lists 3000 physical interpretations of trihedral junctions; it includes curved lines derived from parabolic and elliptical surfaces. Chakravarty's catalogue [13] handles general curved objects but remains limited to trihedral vertices. It describes V-, W-, Y-, T-, A- and S-junctions—the A- and V-junctions are subcategories of L-junction; understanding the name of the S-junction takes some imagination (it is a tangent meeting a curve)—and the "invisible" C-junction, obtained by following a concave curve. Malik [99] has listed errors in this catalogue, and in a simplified version of it by Lee et al [85], and provided a correct catalogue [100]. Catalogues have also been produced for interpreting drawings of Origami objects [62] and for assisting in interpretation of scenes with shadows [181] and of range data [158].

Grimstead [39] investigated the possibility of an incremental line-labelling algorithm which updated labellings whenever a new line was added to a drawing. He notes that the extra complexity introduced by allowing for incremental line labelling does not translate into a significant benefit in terms of processing time—line labelling takes a small proportion of the total time needed to interpret a drawing. This proportion would drop further if more "intelligence" were to be built into the later stages.

A full catalogue of valid tetrahedral junction labels would clearly be considerably larger than the equivalent Clowes-Huffman catalogue for trihedral junction labels.

Its very size may make it less useful in practice, as may the fact that it is far harder to obtain an unambiguous labelling for a line drawing explicitly or implicitly containing tetrahedral vertices. This has prompted some to use cut-down catalogues of common tetrahedral junctions [181] rather than the full catalogue, and others to investigate other methods of labelling the lines in drawings of objects with tetrahedral vertices (one of these is Pugh's arc labelling method [133]).

Others still follow Malik's work [100], which extends trihedral line-labelling to curves and also suggests a method, the "local minimum complexity rule", for dealing with cases where more than three faces meet at a vertex: use the trihedral catalogue for L-, W- and Y-junctions, assume that T-junctions are occluding, and accept whatever this produces at visibly tetrahedral junctions. However, this does not address the more difficult problem of inferring that a boundary junction where only two or three lines are visible is non-trihedral in the completed object. This procedure can find an inferior labelling (as with Figure 4.7, where the topmost junction should be Wbda, not Wbca) or fail to find any valid labelling (as with Figure 4.8) (there are many other examples below; see in particular Appendix E.2.18 and E.2.19) while also failing to avoid the combinatorial explosion when attempting to label drawings such as Figure 4.9. More subtly, it may obtain the best labelling for Figure 4.10 but fail to identify that the topmost W-junction should be non-trihedral.

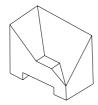


Figure 4.7: Non-trihedral W [194]

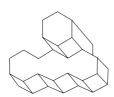


Figure 4.8: Non-trihedral Y



Figure 4.9: Archimedean

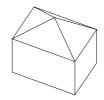
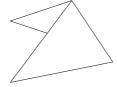


Figure 4.10: House

Malik has shown [99] that one necessary consequence of the local minimum complexity rule is that there is no more than one hidden face at any vertex. Figures 4.11–4.13 show counter-examples where two hidden faces are required by the best interpretation, further undermining the validity of this method.

Malik [100] also suggested the following procedure for generating junction catalogues: starting with a valid fully-visible vertex type, replace any adjacent pair of





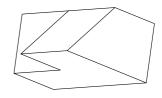


Figure 4.11: L

Figure 4.12: W

Figure 4.13: T

concave edges by an arriving-departing pair of occluding edges. In the trihedral domain, this method generates a correct catalogue of W- and Y-junctions (but not L-junctions). In the tetrahedral domain, it suffers from the problem that identifying valid fully-visible vertex types is non-trivial.

Furthermore, the geometric reasoning used to develop this approach is implicitly trihedral, and does not extend infallibly to the tetrahedral domain. When applied to the 19 underlying vertex types identified in Table 4.1 on page 65, Malik's method would identify as valid an additional 13 junction labels: Xabdc, Xabcd, Xabdc, Mbcca, Mbcca, Mbcca, Mbcca, Mbcca, Mbcca, Kabcd, Kbdca, Kabdc and Kdabc (it would not, and was not intended to, generate implicitly-tetrahedral L-, W-, Y- and T-junctions). Eight of these, Xabdc, Xabcd, Xabdd, Mbcca, Mbcca, Mbcda, Kabcd and Kabdc, are also identified by the methodology given below, and with the 19 full-view tetrahedral junction labels comprise the complete set. The remaining five, Mcabc, Mabcd, Mdcab, Kbdca and Kdabc, are either implicitly pentahedral (see Figure 4.14), excluded from the tetrahedral catalogue since they can only be generated by including hinges (Figures 4.16 and 4.17), or both (Figures 4.18 and 4.19).

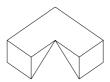


Figure 4.14: Mcabc

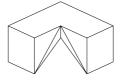
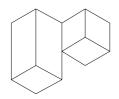


Figure 4.15: The same object tilted

Huffman [58] advocated analysis of *cut sets*, the set of edges incident at either a single vertex or a tight group of neighbouring vertices. For example, in Figure 4.20, the convex edge between faces A and B implies that at point Φ , the plane of face A is further away than the plane of face B ($z_A > z_B$). The concave edge between faces



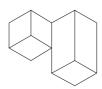






Figure 4.16: Kbdca

Figure 4.17: Kdabc

Figure Mabcd

4.18:

Figure 4.19: Mdcab

B and C implies that at point Φ , $z_B > z_C$. Similar reasoning shows that $z_C > z_D$ and that $z_D > z_A$. The existence of such a *cyclic inequality* demonstrates that this particular labelling is invalid, and, more generally, the non-existence of any such Φ -point is a necessary condition for the labelling to be realisable [58].

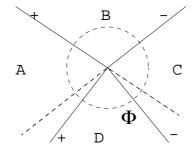


Figure 4.20: Cut Set, from [58]

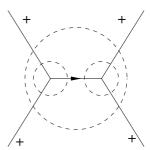


Figure 4.21: Cut Set Counterexample, from [58]

It is not clear how this can be translated into a fast algorithm (since cut sets may surround groups of neighbouring vertices as well as single vertices, a natural implementation would be significantly slower than the algorithm described in Section 4.4.2). Huffman's original idea made use of dual space [57] (see Chapter 3.7), but this is artificial and unnecessary. Although, as Huffman [58] notes, his idea can be implemented in such a manner as to make it tolerant of small numerical errors, it is not naturally tolerant of drawing inaccuracies. Furthermore, Huffman [58] also showed that the absence of a Φ -point is not a sufficient condition for the labelling to be realisable, providing Figure 4.21 as a counter-example—all three cut sets pass the test, but the labelling is nevertheless clearly incorrect (catalogue-based labelling using the tetrahedral junction catalogue would also accept this labelling as valid).

Huffman's idea can be used in other ways, for example as a way of generating or validating junction catalogues. For generating catalogues, I consider it inferior to the method outlined in Section 4.3.1 below, which provides not only the catalogue but knowledge of the underlying vertex types implied by each entry. For validating catalogues, it is correct—it can be noted that the discrepancies between Huffman's non-occluding T-junction catalogue [58] and that in Table 4.6 on page 69 are caused by differences in assumptions. Both include the 16 non-occluding tetrahedral T-junctions. Table 4.6 also includes the four occluding T-junctions (Tbaa, Tbab, Tbac and Tbad) whereas Huffman also includes seven non-occluding pentahedral T-junctions (Tbaa, Tbab, Tbac, Tcaa, Tbcb, Tdaa and Tbdb).

Draper's sidedness reasoning [23, 24] is, in essence, an extension of Huffman's cyclic inequality idea across an entire drawing, avoiding any need for a junction catalogue. It is, as Draper claims, intuitively correct, but it is not informative, in that the junction catalogue provided additional information which line labels alone do not.

Kanatani [64] has suggested that non-trihedral vertices can be decomposed into trihedral vertices by splitting them, and if necessary adding a zero-length line. For example, Figure 4.22 would be processed as if it were Figure 4.23.

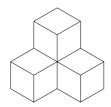


Figure 4.22: Drawing

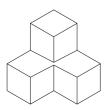


Figure 4.23: After Decomposition

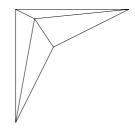


Figure 4.24: Quadrilateral Pyramid

This would require care—in order to avoid side-effects in subsequent stages of processing, the "pretend edge" must be removed as soon as possible after completion of line labelling.

However, the major concern here is how the splitting would be performed. It would seem natural, for example, when attempting to split a tetrahedral vertex to attempt to divide space into four quadrants, and treat the zero-length line as being in the direction of one of the dividing axes. Consider Figure 4.24. The central vertex can be split into two trihedral vertices either by a horizontal line or a vertical line. In either case, two Y-junctions will be produced, and there will be no valid trihedral labelling. It is possible to find directions for the zero-length line which would achieve

the desired result (one Yccc junction and one Wcdc junction), but it is not clear how to arrange for this to be the first choice.

It seems likely that this problem is soluble and that Kanatani's idea is an improvement on Malik's suggestion of allowing anything at visible non-trihedral junctions, but the problem of inferring that apparently-trihedral junctions correspond to non-trihedral vertices remains (one valid interpretation of Figure 4.24 is an irregular bipyramid).

4.2.2 Algorithms

Once a junction catalogue has been created, it can be used to label drawings. Combinations of labelled lines meeting at a junction which do not produce a junction label listed in the catalogue can be rejected. The task is translated into a discrete constraint satisfaction problem, with the constraints that each line must have the same label throughout its length, and each junction must be allocated a labelling listed in the catalogue. Several effective algorithms for this discrete constraint satisfaction problem have been proposed.

It has been apparent for some time that although such constraint satisfaction problems are NP-complete in the worst case [69], in practice many line drawings can be labelled correctly in almost linear time using deterministic algorithms such as Waltz's propagation algorithm [181]. This surprising result is generally believed to derive from the sparsity of the trihedral junction catalogue and the resulting lack of ambiguity in the chosen labelling. Fixing the label of one junction will usually fix the labels of its neighbours.

After testing this conjecture, Parodi et al [120] report that a median-case performance of O(n) can be achieved in practice if random objects are chosen. This investigation would have been more useful if it had been coupled with an analysis of structure, as some features (e.g. holes, pockets and bosses) commonly give rise to line drawings with multiple valid labellings. Since it can be shown that any line drawing with a single valid labelling can be labelled in low-order polynomial time (Kanatani's set-intersection method [64] is demonstrably $O(n^2)$ for drawings with no more than a single valid labelling), it is the possibility of numerous alternative valid labellings which makes the problem show NP-complete behaviour in the worst

case.

By [11] reports that implementing junction and line labels as Prolog body goals and using the built-in unification of a Prolog interpreter (Quintus 3.1.1) to propagate constraints is faster than an interpreted implementation of Waltz's original constraint propagation algorithm [118, 181]. No mention is made of the order of this method as problem size increases, and the reported timings are difficult to explain in terms of a polynomial-order algorithm. The timings of By's method are reasonable for drawings of realisable trihedral objects but excessive for drawings with no realisable trihedral interpretation. This cannot be considered an improvement on Kanatani's algorithm.

An attempt to improve on Waltz's and Kanatani's algorithms using genetic algorithms was unsuccessful [153].

Since line labelling is theoretically NP-complete, attempts have been made to determine whether or not a sketch is labellable. One such is Kirousis's algorithm [68], which is O(n). This is limited to trihedral polyhedra (in particular, it requires that there is at least one non-occluding line at every Y-junction) and makes use of a stronger definition of "general projection" (it requires that lines parallel in 2D correspond to edges parallel in 3D).

An alternative method of labelling lines, also proposed by Kanatani [64], assumes not only that vertices are trihedral but that the object is a normalon. It labels lines according to their concavity/convexity and also uses their orientation to classify them by axis as i, j or k lines. It is more successful in rejecting impossible objects, and provides additional useful information for later stages of the recognition process. The assumption of object orthogonality is, however, a major limitation.

The junction labels computed as part of the line labelling process are themselves useful, and this may explain why other methods of line-labelling such as gradient space [96], cut sets [58] and sidedness reasoning [23] have not superseded junction catalogue approaches (other advantages and disadvantages of these two methods are discussed elsewhere [24, 163]). For example, junction labels determine (sometimes uniquely) the underlying vertex types, simplifying topological reconstruction (Chapter 10), and pairs of junction labels can be used in inflation (Chapter 7).

It is Kanatani's set-intersection method [64] which forms the basis of the algorithm described in Section 4.4.2 below.

4.3 Tetrahedral Junction Catalogue

Despite the initial success of line labelling methods, the limitation to polyhedra with trihedral junctions has proved a problem. Real engineering objects are often not trihedral (Figures 1.1 and 1.2 on page 2 are not). Of the 85 test drawings adapted from engineering drawing textbooks [128, 129, 194], 53 are trihedral, 29 are tetrahedral, one (Figure B.478, page 329) is pentahedral and one (Figure B.471) is hexahedral (it is also extended trihedral). In practice, pentahedral vertices encountered in engineering objects are usually all-convex, as in Figure B.177. Hexahedral vertices are more common but are usually either all-convex or alternating as in Figure 4.6 (page 55). Figure B.402 is the only occurrence of an extended tetrahedral vertex in the test drawings. It is therefore worthwhile investigating the full tetrahedral vertex catalogue, while recognising that the pentahedral, hexahedral and extended tetrahedral vertex catalogues are of little interest in this application domain.

4.3.1 Generating the Tetrahedral Catalogue

My aim here is twofold: to derive the complete catalogue of junctions which may appear in line drawings of polyhedra with tetrahedral vertices, and to arrange this catalogue such that all of the possible junction types which can occur in views of a single underlying vertex type are grouped together. The method used is to divide space into regions by creating four planes which intersect at the origin, consider all sensible combinations (see below) of full and empty regions, and observe the central vertex from all empty regions. This is first expressed as an algorithm, with implementation details being described subsequently.

- Split the Gaussian sphere into regions by creating four planes through the origin.
- Repeat for each combination of solid and empty regions which meet the criteria for a valid single polyhedral object
 - View the central vertex from one viewpoint located in each empty region.
 For each view:
 - * Count the number of visible edges (if the ray from the intersection of the

edge with the Gaussian sphere to the viewpoint passes through a solid region, the edge is not visible)

- * Determine the orientation of the visible edges, and order them clockwise
- * For each visible edge, determine whether it is concave, convex, clockwise-occluding or anticlockwise-occluding.
- Derive the junction label from the number of visible edges and the edge types
- Output the set of junction labels for the different viewpoints of this vertex as a single group

As with the Clowes-Huffman procedure for trihedral vertices, it is the topology of the chosen planes which matters, not the geometry. For simplicity, three of the planes can be chosen to be the three axial planes. There are two distinct possibilities for the fourth plane: its normal may be in one of the axial planes (this subdivides space into twelve regions and creates K-type junctions), or not (this subdivides space into fourteen regions and creates X-type junctions). The two cases are treated separately. For simplicity, the fourth planes can be chosen to be X + Y = 0 for the former case and X + Y + Z = 0 for the latter.

A combination of solid and empty regions is valid providing it meets the following criteria:

- at least one region must be solid
- at least one region must be empty
- the solid regions must be contiguous, in order for the solid to be manifold
- the empty regions must be contiguous (to exclude degenerate vertices such as those which could be produced by interpreting Figure B.81)
- points and edges may not be degenerate (e.g. the four regions viewed cyclically about an edge may not be solid-empty-solid-empty, as this would produce two degenerate edges)
- none of the planes may divide the sphere into an entirely solid part and an entirely empty part (there would be no vertex to see)

If the object is not viewed from a general viewpoint, an accidental collinearity of lines could cause (for example) W-junctions and Y-junctions to be misclassified as T-junctions. As a precaution against this, the viewpoints chosen for each empty region are offset from the centre of the region by different amounts along the three axes.

In the same way that the method for cataloguing trihedral vertices also produces some, but not all, tetrahedral and higher vertices, the approach described above produces some, but not all, pentahedral and higher vertices. In particular, the most useful of these, the ones likely to appear in line drawings (all-convex and all-concave), are not generated. The extra vertices generated were discarded and are not listed below. In order to provide reassurance that the program implementing this approach was working correctly, the trihedral vertex results were kept and are listed also below; they correspond exactly to the Clowes-Huffman trihedral junction catalogue.

4.3.2 The Tetrahedral Junction Catalogue

Table 4.1 shows the complete tetrahedral junction catalogue (for comparison, Table 4.2 shows the equivalent trihedral junction catalogue). The columns list: the vertex type; the junction labels which can be produced by viewing this type from different viewpoints; and an appendix section which contains drawings illustrating the vertex type. The results are listed in the following order:

- X-type underlying vertex
- *M*-type underlying vertex
- K-type underlying vertex (and K^* —see Chapter 2.4.1)

Within each section, the groups of junction types are listed in order of increasing concavity of the underlying vertex type.

In some cases, the underlying vertex type has chirality. In such cases, the groups of junction types for the "left-handed" and "right-handed" versions are shown separately. Where the underlying vertex type has no chirality only one group is shown even where the illustrative solid is chiral—it may be reversed without affecting the

Vertex Type	Junction Types	Section
Xcccc	Xcccc, Mbcca, Wbca, Lba	E.2.1
Xcccd	Xcccd, Mbcda, Mbdca, Yabc, Yabd, Yacc,	E.2.2
	Ybcc, Wbaa, Wbba, Wbca, Wbda, Lba	
Xcdcd	Xcdcd, Yacd, Ybdc, Yabd	E.2.3
Xcddd	Xcddd, Yadd, Ybdd	E.2.4
Xdddd	Xdddd	E.2.5
Mccdc	Mccdc, Xabcd, Yaab, Yabd, Wcab,	E.2.6
	Wcac, Wccb, Lac, Lcb, Lab	
Mcdcc	Mcdcc, Xabdc, Yabb, Yabd, Wabc,	E.2.7
	Webe, Wace, Leb, Lac, Lab	
Mcddc	Mcddc, Xabdd, Wadc, Wcdb, Lac, Lcb, Lab	E.2.8
Mdccd	Mdccd, Wbcd, Wdca, Lba, Lbd, Lda	E.2.9
Mcdcd	Mcdcd, Yabd, Yacd,	E.2.10
	Wcda, Wcbd, Wacd, Wabd, Lbd, Laa	
Mdcdc	Mdcdc, Yabd, Ybdc,	E.2.11
	Wbdc, Wdcb, Wdab, Wdac, Lda, Lbb	
Mddcd	Mddcd, Yadd, Wdbd, Wdda, Lbd	E.2.12
Mdcdd	Mdcdd, Ybdd, Wdad, Wbdd, Lda	E.2.13
Kcccd	Kcccd, Kabcd, Yabd,	E.2.14
	Taba, Tbca, Tbcc, Tcca, Iab	
Kccdc	Kccdc, Kabdc, Yabd,	E.2.14
	Tabb, Tcab, Tcac, Tccb, Iab	
Keded	Kcdcd, Ybdc, Yabd, Tbda, Tbdc, Tcda	E.2.16
Kdcdc	Kdcdc, Yacd, Yabd, Tdab, Tdac, Tdcb	E.2.17
Kcdcd*	Wdcb, Wdab, Tbda, Tbdc, Lda, Lbb	E.2.18
Kdcdc*	Wacd, Wabd, Tdab, Tdac, Lbd, Laa	E.2.19
Kddcd	Kddcd, Tdda, Ybdd	E.2.20
Kdcdd	Kdcdd, Tddb, Yadd	E.2.21

Table 4.1: Junction Catalogue

Vertex Type	Junction Types	Section
Yccc	Yccc, Wbca, Lba	E.1.1
Wede	Wcdc, Yabd, Lcb, Lac, Lab	E.1.2
Wdcd	Wdcd, Lda, Lbd	E.1.3
Yddd	Yddd	E.1.4

Table 4.2: Trihedral Junction Catalogue

results. Most M-type underlying vertices and all K-type are chiral; two M-type and all X-type are non-chiral.

Tables 4.3–4.9 on page 67 list all junction labels and underlying vertex types identified by RIBALD—the full trihedral, extended trihedral and tetrahedral junction catalogues, and all-convex and all-concave pentahedral and hexahedral vertices.

The method described above not only gives the catalogue, but (since it lists the combination of solid and empty regions which gave rise to each junction label) it also gives the information necessary to create an example solid which illustrates each vertex type and junction label. These illustrative solids are shown in Appendix E. The solids illustrating the trihedral catalogue are built from cubes, although non-axially-aligned trihedral polyhedra generate the same junction types. The solids illustrating the X-type and M-type tetrahedral vertices are built from cubes, triangular pyramids (obtained by choosing a pyramid vertex and slicing a cube in the plane of the three vertices adjacent to the pyramid vertex—Figure 4.25 is built from a cube and such a pyramid) and shapes termed oojits (that which is left after removing a triangular pyramid from a cube—see Figure 4.26). The solids illustrating the K-type tetrahedral vertices are built from cubes and wedges (obtained by slicing cubes along the plane of diagonally-opposed edges).



Figure 4.25: Cube plus Pyramid



Figure 4.26: Oojit

Junction Label	Vertex Types	Junction Label	Vertex Types
Kabcd	Keeed	Kabdc	Keede
Keed	Keeed	Keede	Keede
Keded	Keded	Kdcdc	Kdcdc
Kddcd	Kddcd	Kdcdd	Kdcdd

Table 4.3: K-type Junction Labels

Junction Label	Vertex Types
Laa	Mcded, Kdede
Lab	Wede, Meede, Medde
Lac	Wede, Meede, Medde
Lba	Yecc, Xecce, Xeccee, Xecced, Mdeed
Lbb	Mdcdc, Kcdcd
Lbd	Wdcd, Mdccd, Mcdcd, Mddcd, Kdcdc
Lcb	Wede, Meede, Medde
Lda	Wded, Mdeed, Mdedd, Keded

Table 4.4: L-type Junction Labels

From the catalogue, it can be seen that whereas there are only 4 basic trihedral vertex types, there are 19 basic tetrahedral vertex types (5 X-type, 8 M-type and 6 K-type). In the trihedral case, excluding T-junctions, there are 12 possible valid junction labellings (14 if different viewpoints of equivalent Y-type junctions are counted separately); in the tetrahedral case, there are 91 possible valid junction labellings (127 if different viewpoints of equivalent Y-type and X-type junctions are counted separately). Of the 64 conceivable labellings for a Y-junction, only 5 are valid in line drawings of trihedral objects, but 32 are valid in line drawings of tetrahedral objects. The number of valid W-junction labellings increases from 3 to 28, and the number of valid L-junction labellings increases from 6 to 8. In the trihedral case, each junction label identifies the underlying vertex type (the number of concave edges) directly, but in the tetrahedral case, 73 of the junction labels identify the underlying vertex type (the total number of edges, the number of concave edges and the chirality) unambiguously but 18 do not, the worst being the Y-abd junction type, for which there are 11 possible interpretations.

Junction Label	Vertex Types
Mbcca	Xcccc, Xccccc, Xcccccc
Mbcda	Xcccd
Mbdca	Xcccd
Mccdc	Mccdc
Medee	Mcdcc
Mcddc	Mcddc
Mdccd	Mdccd
Mcdcd	Mcdcd
Mdcdc	Mdcdc
Mddcd	Mddcd
Mdcdd	Mdcdd
Mbccca	Xeccec

Table 4.5: M-type Junction Labels

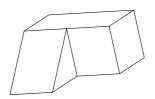


Figure 4.27: Non-Trihedral

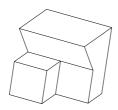


Figure 4.28: Non-Trihedral

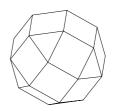


Figure 4.29: Non-Trihedral

The level of ambiguity has effects both on the speed of applications using the tetrahedral junction catalogue and its usefulness in identifying three-dimensional structure. To illustrate the former, RIBALD takes about 12 seconds to produce all of the valid labellings for the line drawings in Appendix B.2.3, compared with 1/3 of a second to label the same number of trihedral line drawings. To illustrate the problem, the line drawings in Figures 4.27, 4.28 and 4.29 have, respectively, 16, 240 and 10206 valid tetrahedral labellings, whereas the apparently more complex

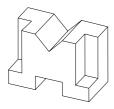


Figure 4.30: Trihedral

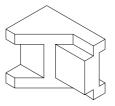


Figure 4.31: Trihedral

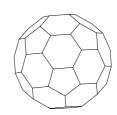


Figure 4.32: Trihedral

Junction Label	Vertex Types	Junction Label	Vertex Types
Taba	Keced	Tabb	Kccdc
Tbaa	occlusion	Tbab	occlusion
Tbac	occlusion	-	-
Tbad	occlusion	-	-
Tbca	Keced	Tcab	Keede
Tbcc	Keced	Tcac	Keede
Tbda	Keded, Keded*, Zeded*	Tdab	Kdcdc, Kdcdc*, Zcdcd*
Tbdc	Keded, Keded*, Zeded*	Tdac	Kdcdc, Kdcdc*, Zcdcd*
Tcca	Keced	Tccb	Keede
Tcda	Keded	Tdcb	Kdcdc
Tdda	Kddcd	Tddb	Kdcdd

Table 4.6: T-type Junction Labels

trihedral Figures 4.30, 4.31 and 4.32 can be labelled unambiguously.

A final problem, not found in the trihedral object world, is illustrated in Figure 4.33. K-type tetrahedral junctions can vanish if viewed from the wrong orientation! There is no way of detecting the presence of a vertex from the junction-line graph (although this does not affect the labelling itself). No such phenomenon can occur with trihedral polyhedra.

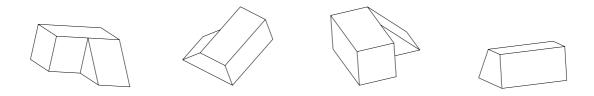


Figure 4.33: Vanishing K-type tetrahedral junction [175]

A candidate set of 16 possible labellings can be processed in a reasonable time in interactive applications such as RIBALD, and even 240 might not be considered excessive for some applications. If a first-choice interpretation is required, the problem of picking the "correct" one is not insoluble. For drawings such as Figures 4.27 and 4.28, where the number of possible interpretations is moderate, it seems reasonable to attempt to validate each possible interpretation by attempting to create the corresponding frontal geometry and validating the result geometrically.

Junction Label	Vertex Types	Junction Label	Vertex Types
Wabc	Mcdcc	Wcab	Mccdc
Wabd	Mcdcd, Kdcdc*	Wdab	Mdcdc, Kcdcd*
Wacc	Mcdcc	Weeb	Mccdc
Wacd	Mcdcd, Kdcdc*	Wdcb	Mdcdc, Kcdcd*
Wadc	Medde	Wcdb	Mcddc
Wbaa	Xcccd	Wbba	Xcccd
Wbca	Yccc, Xcccc, Xcccd	-	-
	Xccccc, Xcccccc		
Wbcd	Mdccd	Wdca	Mdccd
Wbda	Xcccd	-	-
Wbdc	Mdcdc	Wcda	Mcdcd
Wbdd	Mdcdd	Wdda	Mddcd
Weac	Mccdc	Webe	Mcdcc
Webd	Mcdcd	Wdac	Mdcdc
Wede	Wede	-	-
Wdad	Mdcdd	Wdbd	Mddcd
Wdcd	Wdcd		-

Table 4.7: W-type Junction Labels

4.4 Two Labelling Approaches

With the tetrahedral catalogue, line labelling can in principle be used with equal success for drawings of trihedral and tetrahedral objects. However, the dramatic increase in the number of valid labellings noted in the previous section presents practical problems. Since it was the sparsity of the junction catalogue which led to practical low-order timings in the trihedral case, and since the tetrahedral catalogue is no longer sparse (at least in the cases of L-, W- and Y-junctions), exponential rather than low-order polynomial behaviour is observed in practice: for example, Figure B.503 has more than 200000 valid labellings, and evaluating all of them is clearly impractical.

The approach in this thesis starts with one advantage: since, by assumption, a drawing shows a single, entire object, all outer boundary lines in the drawing must be occluding. This assumption is not valid for applications such as identification of buildings from aerial photographs: in such cases, the potential for ambiguous labellings is greater, and thus the labelling problem would be even harder.

Junction Label	Vertex Types
Xabcd, Xbcda, Xcdab, Xdabc	Mccdc
Xabdc, Xbdca, Xdcab, Xcabd	Mcdcc
Xabdd, Xbdda, Xddab, Xdabd	Mcddc
Xeecc	Xcccc
Xcccd, Xccdc, Xcdcc, Xdccc	Xcccd
Xcdcd, Xdcdc	Xcdcd
Xcddd, Xdddc, Xddcd, Xdcdd	Xcddd
Xdddd	Xdddd
Zcbda	Zededed
Xccccc	Xccccc
Xddddd	Xddddd
Xccccc	Xccccc
Xdddddd	Xdddddd
Zededed, Zdedede	Zededed

Table 4.8: X-type and Z-type Junction Labels

Furthermore, whereas in the trihedral case, each junction label determines unambiguously the underlying vertex type of the corresponding vertex, this does not extend to the non-trihedral case. In the worst case, a single junction label can be interpreted as any of seven underlying vertex types: two convex and one concave edges; three convex and one concave edges, in K-, M- or X-configuration; two convex and two concave edges, in K-, M- or X-configuration [175]. The pairs of drawings in Figure 4.34 show three of these interpretations (in each pair, the left-hand drawing shows the ambiguous junction label and the right-hand drawing shows the revealed vertex type).

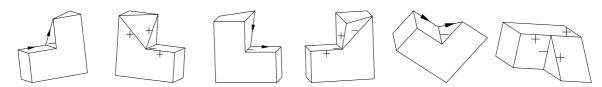


Figure 4.34: Three objects illustrating different interpretations of a single junction label [175]

A deterministic algorithm [64] which has been used successfully to label trihedral drawings [172] has been adapted for the case where multiple valid labellings are the norm, not the exception. As expected, there are drawings for which its time performance is unacceptable.

Junction Label	Vertex Types
Yaab, Yaba, Ybaa	Mccdc
Yabb, Ybba, Ybab	Medee
Yabc, Ybca, Ycab	Xcccd
Yabd, Ybda, Ydab	Wede, Xeced, Meede, Medec, Keeed, Keede,
	Xeded, Meded, Mdede, Keded, Kdede
Yacc, Ycca, Ycac	Xcccd
Yacd, Ycda, Ydac	Xeded, Meded, Kdede
Yadd, Ydda, Ydad	Xeddd, Mdded, Kdedd
Ybdc, Ydcb, Ycbd	Xeded, Mdede, Keded
Ybdd, Yddb, Ydbd	Xeddd, Mdedd, Kdded
Yccc	Yecc
Yddd	Yddd

Table 4.9: Y-type Junction Labels

Junction	Tri-	Extended	Tetra-	Total
Type	hedral	Trihedral	hedral	Valid
L	6	0	2	8/16
Τ	4	4	12	20/64
W	3	0	25	28/64
Y	5	0	27	32/64
K	0	0	8	8/256
M	0	0	11	11/256
X	0	1	24	25/256

Table 4.10: Number of Entries in Junction Catalogues

The problem can be considered as a search problem, where the space to be searched is the set of valid labellings, and the search criterion is a heuristic measure of the merit of a particular labelling. Non-deterministic algorithms have been used with some success in NP-complete search problems. To investigate this possibility, a non-deterministic algorithm has also been tested to see if it improves on the deterministic algorithm.

The non-deterministic algorithm chosen is based on probabilistic relaxation labelling [140]. To avoid confusion, the deterministic method, sometimes called "discrete relaxation labelling", is called *set-intersection labelling* in this thesis.

4.4.1 Heuristics

There are two distinct purposes for which heuristics can be used in improving the performance of any search: *choice* heuristics assist in choosing the best interpretation when there are many valid interpretations, and *pruning* heuristics help to speed up the search by lopping off unlikely branches of the search tree.

Heuristics used for line labelling can also be subdivided into *global* heuristics, based on measures derived from the drawing as a whole, *feature* heuristics, measures derived from a part of the drawing, and *local* heuristics, measures based on individual lines or junctions.

RIBALD assigns a figure of merit to each labelling, and the labelling with the highest figure of merit is the one preferred. This overall figure of merit is the product of the figures generated by each heuristic.

Global Labelling Heuristics

It seems plausible that the object should be as "simple" as possible. If the object represented in a drawing has any self-similarity (repetitiveness or symmetry), the number of different underlying vertex types (the numbers and types of edges incident at the vertices corresponding to junctions) in the object will be small. One reasonable heuristic is that, as far as possible, the labelling should minimise the number of different vertex types.

Since some junction labels determine uniquely the underlying vertex type, while others do not, the merit figure is assessed in a two-stage process. Firstly, those junction labellings which correspond unambiguously to a single underlying vertex type are noted, and the minimum set of underlying vertex types established. The final count is then the sum of the number of underlying types in this set and the number of ambiguous junction labellings which cannot be interpreted as any of the labellings in this set (in practice, the second number is almost always zero).

Numerically, this preference is quantified as $(L_u + L_s)^{-k_L}$, where L_s is the number of different underlying vertex types required by those junction labels which correspond to unique underlying vertex types, L_u is the number of junction labels which cannot be interpreted as one of the unique underlying vertex types already required and counted in L_s , and the sum $(L_u + L_s)$ provides an estimate of the number of

distinct underlying vertex types in the object; k_L is a tuning constant.

In order to identify an optimum value of the tuning constant k_L (and other tuning constants described below) these were input as parameters to a downhill optimisation process where the objective function being minimised was the number of incorrectly-labelled edges in my set of test drawings (see Appendix C). Results suggest that the optimum value of k_L is between 0.5 and 0.6, implying that this heuristic is reasonably useful in identifying preferred labellings.

It is also plausible that if a drawing is of a single object, it should be as "connected" as possible¹—occluding lines in the interior of the sketch could be considered undesirable (for example, in Figure 4.35 on page 78, the line marked * should not be occluding). This is modelled by assigning a figure of merit $(1 - \frac{E_o}{E_t})^{k_E}$ to the labelling, where E_o is the number of occluding lines, E_t the total number of lines, and k_E a tuning constant. However, investigations suggest that the optimum value of k_E is close to zero, implying that this heuristic is of little use in identifying preferred labellings.

Feature Labelling Heuristics

RIBALD tests for the presence of certain features—pockets, bosses and slots—as described in Chapter 6. Each such hypothesised feature has an associated figure of merit M_h and requires the lines forming part of that feature to be labelled in a particular way.

RIBALD calculates a feature merit figure for a labelling: the product of the merit figures M_c for each feature, where $M_c = 1$ for labellings which match the expectations for the hypothesised feature, and $M_c = (1 - M_h)^{K_c}$ for labellings which do not match the hypothesis (K_c is another tuning constant).

Local Labelling Heuristics

Some junction labels are "better"—more common, or more plausible—than others. Preliminary investigations show that even simple heuristics such as "interpret as many T-, W- and Y-junctions as trihedral as possible" identify the favoured interpretation in about half of the cases tested, including those shown in Figures 4.27

¹Mackworth's POLY [96] is an extreme example of this, generating all possible interpretations of a drawing in decreasing order of connectedness

and 4.28 on page 68, and in most other cases left the favoured interpretation in the first few when arranged in a preference order based on this heuristic.

RIBALD uses a more sophisticated model, assigning a constant figure of merit to each junction label (determined using the optimisation process described above and normalised so that the "best" label for any junction type has a figure of merit of 1.0). The contribution of this heuristic to the overall figure of merit for the labelling is the product of the figures of merit for each individual junction label.

4.4.2 Deterministic Labelling

As a particular deterministic labelling method, RIBALD implements a set-intersection labelling approach derived from Kanatani's iterative constraint propagation method [64]. At its core is the following algorithm:

- (Initialisation)
- For each junction, candidate label set = all valid labels for that junction type;
- For each boundary line, candidate label set = {occluding such that outside is occluded}
- For each non-boundary line, candidate label set = {occluding to left, occluding to right, convex, concave}
- Set of junctions to be processed $S_j = \{all junctions\};$
- Set of edges to be processed $S_l = \{all edges\};$
- (Processing)
- Loop
- - For each junction in S_j
 - * Eliminate from the candidate label sets for neighbouring lines any line labels inconsistent with the remaining candidate labels for this junction;
 - * If the junction label is unique, remove the junction from S_j ; (see Note 2 on Page 78)

- For each line in S_l

- * Eliminate from the candidate label sets for the neighbouring junctions any junction labels inconsistent with the remaining candidate labels for this line;
- * If the line label is unique, remove the line from S_l
- Exit the loop if S_j and S_l are both empty (a unique labelling has been obtained)
- Exit the loop if the set of candidate labels for any junction or line is empty
 (no valid labelling can be obtained given the starting conditions)
- Exit the loop if no candidate labels were eliminated in this iteration

End Loop

This algorithm is demonstrably $O(n^2)$ (with n being the number of lines in the drawing). In the worst case, each iteration of the loop removes a single candidate junction or line label; the number of these is proportional to the number of lines.

In practice, the algorithm is sufficient to obtain a unique labelling for the majority of drawings which meet the simplifying assumptions that the object drawn is a trihedral polyhedron with no through holes or hole loops (Parodi [120] reports the same result with a different deterministic algorithm and a far larger set of test drawings). In cases where these assumptions do not hold, it is likely that there will be several valid labellings, and further processing required to identify the preferred one.

The algorithm has three exit conditions: no valid labelling, a unique labelling, or no further progress. In the first two cases, it need only be called once. In the last case, it requires a surrounding control structure. At least one junction and at least one line still have multiple possible labels. It is likely that there will be multiple valid labellings compatible with the starting conditions (although this is not certain, as one or more of the remaining possible junction or line labels may be a "dead end").

To allow for ambiguities, the following control structure is added:

If the algorithm terminated ambiguously

- Choose any junction or line where the set of candidate labels L contains more than one element; (see Note 1 later)
- Choose any element l of this set;
- Create a labelling A which is identical to the original labelling except that the label set for the chosen junction or line is $\{l\}$
- Label the rest of A by reentering the algorithm at (*Processing*) above;
- Create a labelling B which is identical to the original labelling except that the label set for the chosen junction or line is $L-\{l\}$
- Label the rest of B by reentering the algorithm at (*Processing*) above;

• End If

In practice, this extension to the original algorithm is adequate when the number of alternative labellings is small, as is commonly the case with drawings of trihedral polyhedra with through holes or hole loops. However, it appears that the worst case (where each disambiguation step fails to propagate to neighbouring junctions or lines) is $O(4^n)$ and this pessimistic prediction is nearer the truth if the non-trihedral junction catalogue is used. If it were possible to identify in advance which apparently-trihedral junctions corresponded to trihedral vertices and which to tetrahedral vertices, so that the full non-trihedral catalogue is used only for the latter, the problem would be alleviated considerably. This does not seem to be possible. For example, it might be possible to infer on the basis of symmetry that T-junction A in in Figure 1.1 (page 2) is a reflected K-junction, but deducing that the object is symmetrical before obtaining the labelling is hard (and not all symmetrically-related junctions are as close to one another as this pair).

In an attempt to speed up practical labelling, RIBALD (a) tries to search the most promising branches of the tree of valid labellings first, and (b) tries to lop off unpromising branches of the tree without processing them.

Note 1: In order to search the most promising branches of the tree first, the choice of which junction or line to disambiguate, and which candidate label to choose to investigate first, is made according to a priority list. Prior to labelling, a list of the "most desirable" junction labels is generated; this is based both on fixed priorities for the most common junction labels and on the merits of candidate features (Chapter 6)

and the resulting junction labels they imply. When the control structure requires a choice, this list is searched, and the highest-priority junction label which would produce a disambiguation is chosen.

Note 2: In order to lop off unpromising branches of the tree, the local contribution to the labelling merit is calculated as each unambiguous junction label is identified. If the current labelling merit of the branch under investigation is already seriously worse than the merit of the best labelling so far, the branch is lopped off. ("Seriously worse" is implemented as $L_n < 2L_0 - 1$, L_n being the current labelling merit and L_0 the best so far; if a single "best" labelling is wanted rather than a choice of reasonably good ones, performance could be improved further by using $L_n < L_0$ instead.) (Note that slightly-unpromising branches will not be lopped off: L_n is based solely on the local contributions to the labelling merit, whereas L_0 also includes the global and feature measures.)

Even this approach can be unacceptably slow for interactive response times, so RIBALD forces the algorithm to return the best labelling it can find in a specified time by limiting the number of tree nodes examined. In any branch of the tree, at most N labellings are examined; these are subdivided whenever the tree branches again, so that the most promising branch is allocated pN nodes and the remaining alternatives (1-p)N nodes. (The results in Section 4.5 were obtained using p=0.7 and an initial N=2000.)

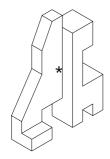


Figure 4.35: One Object or Two? [194]

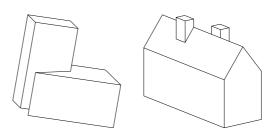


Figure 4.36: The Tetrahedral Catalogue must be used to label these drawings correctly

For several test drawings there is a valid labelling using a more restrictive junction catalogue, but a better labelling can be obtained using a less restrictive catalogue. For example, Figure 4.35 can be labelled using the trihedral catalogue (as two unconnected objects), but the labelling obtained using the extended trihedral

catalogue is clearly preferable. Similarly, the drawings in Figure 4.36 have valid labellings using the extended trihedral catalogue, but a clearly superior labelling can be obtained using the tetrahedral catalogue. While it is possible to create heuristics to choose the non-trihedral interpretation once it has been generated, it seems impossible to know, without generating non-trihedral interpretations, that the trihedral interpretation is not the best.

However, it is also observed that the less restrictive junction catalogue may result in an inferior labelling when the superior labelling can be obtained using the more restrictive catalogue—for example, the superior labelling may be in an unpromising branch of the tree which has been lopped off. It is not intuitively obvious which is the more common occurrence, so for purposes of comparison, RIBALD implements various options for the junction catalogue (the labels in brackets refer to rows of Table 4.13 on page 86):

- (SI-Full) use the full catalogue (all trihedral, extended trihedral and tetrahedral junction labels plus the common symmetrical 5-hedral and 6-hedral labels);
- (SI-LWY) use the trihedral catalogue for L-, W- and Y-junctions, and the full catalogue for other junction types; if no valid labelling is obtained, use the full catalogue instead (several drawings, including Figures 1.1 and 1.2, can be labelled using this method);
- (SI-X3h) use the extended trihedral catalogue; if no valid labelling is obtained, use the full catalogue instead;
- (SI-3h) use the trihedral catalogue; if no valid labelling is obtained, try again using the extended trihedral catalogue and if necessary the full catalogue.

If non-trihedral junctions occur in drawings of plausible engineering objects, rather than in the simple illustrative solids shown here, some method of localising the non-trihedrality would prevent the generation of large numbers of implausible labellings which will inevitably be discarded. Ideally, the initial set of valid labellings should be the non-trihedral set only for those junctions "close" to the centre of non-trihedrality; elsewhere, the trihedral set should be used. My initial investigations

showed that defining "close" as one or two edges away from a visibly non-trihedral vertex was inadequate (for example, it takes no account of object symmetry), and this idea has not been pursued.

4.4.3 Probabilistic Labelling

As a particular non-deterministic labelling method, RIBALD implements a relaxation labelling algorithm. Relaxation methods have been used successfully in several machine vision processes from scene labelling [140] to object recognition in robotic systems [136], and probabilistic relaxation can be viewed as a natural extension of constraint propagation, so probabilistic relaxation was therefore my first choice.

Alternative approaches are possible. Genetic algorithms might be worth revisiting. The disappointing results in [153] were obtained several years ago. More recently, Myers [114] reports successful results in labelling trihedral scenes with genetic algorithms; he stresses the advantage of producing a population of valid labellings, rather than a unique labelling. He gives no timings but indicates that the order of the algorithm is the same as the order of the fitness function, i.e. polynomial. Ant systems [22] also appear worthy of investigation, although it is not clear at this point whether an implementation would differ significantly from probabilistic relaxation labelling, and if the analogy with crystallisation used below is valid, simulated annealing [67, 108] might also be a method worth investigating.

The algorithm as implemented is as follows:

- (Initialise)
- For each junction, allocate a probability (see text) for each candidate junction label for junctions of that type, such that each probability is greater than 0 and the sum of all probabilities at the junction is 1.
- For each boundary edge, set the probability that the edge occludes the outside to 1 and the probabilities that the edge occludes the inside, is convex, or is concave, to 0.
- For each non-boundary edge, allocate a probability for each candidate label (occluding to left, occluding to right, convex or concave), such that each probability

is greater than 0 and the sum of all probabilities is 1.

- Set of junctions to be processed $S_j = \{all junctions\};$
- Set of edges to be processed $S_l = \{\text{all edges}\};$
- (Processing)
- Loop
- ullet For each junction in S_j
 - * Multiply the probability of each candidate label by each of the neighbouring line label probabilities which support this label
 - * Re-normalise the probabilities
 - * If the probability of any label for this junction exceeds a threshold (0.9999), set the probability for this label to 1 and the probabilities for all other labels to 0, and remove the junction from S_j ;
 - For each line in S_l
 - * Multiply the probability of each candidate label by each of the sums of the neighbouring junction label probabilities which support this label
 - * Re-normalise the probabilities
 - * If the probability of any label for this line exceeds a threshold (0.9999), set the probability for this label to 1 and the probabilities for all other labels to 0, and remove the line from S_l
- ullet Exit the loop if S_j and S_l are empty (a unique labelling has been obtained)
- Exit the loop if a specified maximum number of iterations has been exceeded (see text)
- End Loop

One theoretical problem with relaxation is that if there is no limit to the number of iterations it cannot be proved to converge [195]. To overcome this, RIBALD abandons the relaxation labelling algorithm if it has not achieved a unique labelling after a fixed maximum number of iterations. On this basis, the algorithm as described above is O(n); the exact order in practice will depend on how set operations

(and in particular "remove member from set") are implemented, and is likely to be better than $O(n^2)$.

If relaxation labelling fails to converge, labelling then proceeds using the setintersection method described in the previous section, preset with any unambiguous junction or line labels obtained during relaxation. This is not ideal (if O(n) relaxation fails, RIBALD must resort to $O(4^n)$ set-intersection), but the timings given in Section 4.5 below suggest that even when relaxation fails to converge it makes enough progress to leave set-intersection with a manageable task.

My initial experimentation showed that in the majority of cases where relaxation converged, it did so in four to six iterations, and in almost all cases where it converged, it did so in fewer than thirteen iterations. I therefore collected results for differing fixed maximum numbers of iterations (the labels in brackets refer to rows of Table 4.13)

- (Rel-6) six (whereupon several drawings dropped through to the set-intersection method);
- (Rel-10) ten (a few drawings dropped through to the set-intersection method);
- (Rel-20) twenty (only those drawings where relaxation was unlikely to converge dropped through to set-intersection).

There remains the difficulty of identifying the initial probabilities to assign to each candidate junction label. There appears to be no way of deriving these from any theoretical principle; instead, the chosen set of probabilities was obtained by optimising the number of correctly-labelled lines in the set of test drawings.

This optimal set of probabilities contains surprises: for example, the most common L-junction label has the lowest initial probability, and the least common edge label (concave) has the highest initial probability. It may be possible to explain this by analogy with a crystallisation process: if actual frequencies of occurrence are used as probabilities, some parts of drawings will crystallise too quickly, before information from more distant parts of the drawing has arrived.

It seems plausible that a more scientific method of generating the initial probabilities could improve the performance of this method (for example, by basing them on the actual frequency of occurrence of the various junction labels). Also, for

comparison with set intersection, I wished to try labelling L-, W- and Y-junctions with the trihedral catalogue and others with the full catalogue. To test these ideas, RIBALD implements three further options:

- (Rel-Junc) the initial junction probabilities are based on the actual frequency of occurrence of junction labels when the drawings from Appendix B are correctly labelled; the initial line probabilities are as in (Rel-20);
- (Rel-Line) the initial junction probabilities are as in (Rel-20); the initial line probabilities are based on the actual frequency of line labels for the particular pair of junction types joined by the line when the drawings from Appendix B are correctly labelled;
- (Rel-LWY) all initial probabilities are as in (Rel-20), except that non-trihedral junction label probabilities for L-, W- and Y-junctions are zero.

Results suggest that none of these is an improvement. Option (Rel-LWY), which ignores a large part of the tetrahedral catalogue, is significantly worse than any other option.

4.5 Results and Conclusions

For the purposes of this thesis, good labellings must be obtained in interactive time. This section therefore compares the algorithms using two criteria, timing and correctness (an ideal method would be both faster and more correct than the alternatives). These results were obtained using the 535 test drawings of the 558 in Appendix B which can be labelled correctly using only trihedral, extended trihedral and tetrahedral labels.

4.5.1 Timings

The timings in Table 4.11 are those which version (SI-Full) of the algorithm in Section 4.4.2 takes to terminate, i.e. either one candidate labelling is identified as "best" and a number of reasonable runners-up are stored, or it is reported that the drawing has no valid labellings. The timings in Table 4.12 are those version (Rel-20) of the labelling process takes to terminate, i.e. either relaxation labelling

No. of Lines	Minimum	Median	Maximum
1–8	0.00	0.00	0.02
9–16	0.00	0.04	0.31
17–24	0.04	0.18	0.82
25–32	0.06	0.65	6.10
33–40	0.03	1.68	4.13
41–48	1.66	3.20	6.41
49–56	2.16	6.30	9.13
57+	4.89	6.18	28.48

Table 4.11: Set Intersection Labelling: Summary of Average Timings (s)

identifies a "best" labelling within 20 iterations, or the process drops through to set-intersection, as above. All are in seconds.

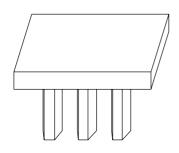
No. of Lines	Minimum	Median	Maximum
1–8	0.00	0.00	0.01
9–16	0.00	0.00	0.01
17–24	0.00	0.00	0.01
25–32	0.00	0.01	0.03
33–40	0.00	0.01	0.05
41–48	0.01	0.03	0.07
49–56	0.01	0.02	0.18
57+	0.02	0.08	0.41

Table 4.12: Relaxation Labelling: Summary of Average Timings (s)

Drawings are grouped into batches according to the number of lines in the drawing. Each table shows the minimum, median and maximum times taken by the labelling process for drawings in each batch.

Relaxation labelling is very much quicker for all but simple trihedral drawings. The timings in Table 4.12 are similar to (perhaps slightly less than) the time taken by deterministic methods using the Clowes-Huffman catalogue to label drawings of trihedral objects. For example, both variants (SI-3h) and (Rel-20) label Figure B.91 in less time than can be measured (i.e. significantly less than 0.01 seconds) on the test machine. For comparison, Grimstead [38], using Waltz's algorithm and the trihedral catalogue, labelled Figure B.91 in 0.018 seconds on a machine slower by about a factor of 8.

(It may be noted, as an aside, that the apparent outlier at 30 edges for set intersection labelling occurs for Figure 4.37. The chain of occluding T-junctions along a single line acts as a propagation boundary.)



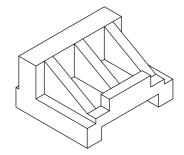


Figure 4.37: Original Drawing

Figure 4.38: From [194]

4.5.2 Correctness of Labelling

Tests have also been performed to determine how often the methods described above produce correct results. In Table 4.13, $\sqrt{}$ indicates the number of drawings where the desired labelling was found, X indicates the number of drawings in which a wrong (but valid) labelling was found, and – indicates that no valid labelling was found. It is clear from the results that option (SI-Full) is more often correct than the other variations of set-intersection (it is also considerably slower), and that option (Rel-20) is at least as good as any other variation of relaxation labelling. Set-intersection is clearly superior at obtaining the desired labelling. However, the number of incorrectly-labelled lines is much closer between the two most promising variants—relaxation often gets a single line label wrong, whereas set intersection errors are clumped.

Given the large number of possible labellings for some of the line drawings in the test set, it is unrealistic to expect any method to identify the preferred labelling for all of them. Ideally, the method should produce the preferred labelling for the majority of cases and a reasonable (if sub-optimal) interpretation of the rest. In practice, even this is not achieved. For example, set-intersection does not find a valid labelling for Figure 4.38—all valid labellings are in branches of the search tree which were lopped off as "unpromising" (Figure B.507 is the simplest drawing in which this happens). Relaxation also does not find a valid labelling for Figure 4.38,

Method		X	_	
SI-Full	428	104	3	
SI-X3h	384	92	59	
SI-3h	367	77	91	
SI-LWY	400	85	50	
Rel-6	387	129	19	
Rel-10	384	130	21	
Rel-20	388	128	19	
Rel-Junc	355	69	111	
Rel-Line	321	135	79	
Rel-LWY	314	68	153	

Table 4.13: Summary of Correct Labellings Achieved

although the reason is different—isolated subgroups of junctions and lines begin to be labelled unambiguously in different parts of the drawing. It is only when these subgroups expand so as to overlap that it is discovered that the partial local labellings are globally incompatible (Figure B.520 is another, simpler example).

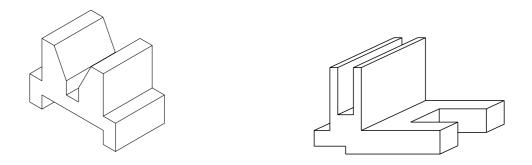


Figure 4.39: Both methods label these drawings correctly

Some generalisations can be made concerning which drawings can be labelled with set intersection and which with relaxation labelling. All variants of both methods succeed in labelling correctly the drawings in Figure 4.39, and most other drawings of trihedral polyhedra. Trihedral junctions are both the most common in drawings (this determines pruning heuristics) and the highest-merit (this determines choice heuristics). The set intersection algorithm takes 1–2 seconds for these two, and relaxation labelling takes 10–20 milliseconds.

All variants of set intersection succeed, and all variants of relaxation fail, for

the drawings in Figure 4.40. Extended trihedral junctions, while uncommon (occluding T-junctions are more common), are high-merit when they do occur (many engineering objects are extended trihedral). The reason for the relative success of deterministic methods with drawings containing mixed-vexity M-junctions is less well-understood.

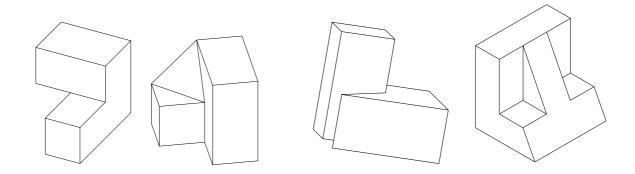


Figure 4.40: Set intersection labels these drawings correctly

Figure 4.41: Relaxation labels these drawings correctly

All variants of the relaxation method succeed, and all variants of set intersection fail, for the drawings in Figure 4.41. In the left-hand drawing, there is one unusual L-junction, all the other junction labels being from the trihedral catalogue. The success of relaxation methods can be explained here by the crystallisation analogy: two separate crystals start to form, and the point where they meet must fit both, however strange it looks when viewed in isolation. The generally superior performance of relaxation methods in labelling drawings containing mixed-vexity K-junctions is significant but less well-understood.

There are also drawings which are not labelled correctly by any of the variants tested. A common source of failure is illustrated by the modified cubes in Figure 4.42, where there are two pairs of topologically-identical figures distinguishable only by geometry. It is evident to the eye which lines should be convex in the left-hand figure of each pair and concave in the right-hand one, but any algorithm based purely on topology will get one or the other wrong. Mislabelling a single line in this manner does not have a serious impact on further processing.

More damaging are the occasions when an occluding line is labelled as non-occluding or vice versa, as with the drawing in Figure 4.43. The rotational symmetry of the implied object suggests that the line segment marked * should be concave,

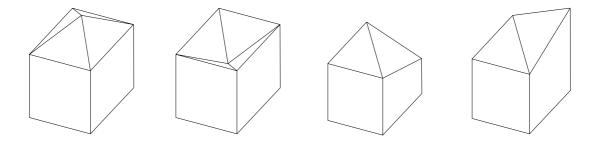


Figure 4.42: Variants of a Cube

not occluding, but neither algorithm takes any account of symmetry. Similarly, in Figure 2.11 (page 29), it is geometrically impossible for the edge marked * to be concave—it must be occluding—but two of the relaxation labelling variants make this mistake. The mislabellings will have the effect of introducing spurious vertices when the hidden parts of the objects are reconstructed.

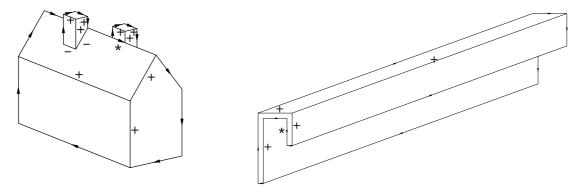


Figure 4.43: Both methods produce this incorrect labelling

Figure 4.44: Occluding in wrong direction

Another problem, observed less frequently, occurs when a line is labelled as occluding the wrong region. The obvious incorrectness of the label marked * in Figure 4.44 does not translate into a simple heuristic which could be applied to prevent such faults occurring.

4.5.3 Conclusions, Recommendations and Future Work

The labelling problem remains non-trivial, especially when non-trihedral vertices are allowed, and no perfect solution has been found. Two approaches have been presented, set-intersection labelling, where only discrete information is propagated

from a junction to its neighbouring edge and vice versa, and relaxation labelling, where probabilistic information is also propagated.

Set-intersection can be slow for larger drawings unless the trihedral catalogue is used for the majority of junctions, and cannot be recommended for drawings with more than 50–70 lines. A method (algorithm or heuristic) for determining which junctions require the full catalogue and which require merely the trihedral catalogue would improve performance considerably, but no such method has as yet been identified.

Relaxation labelling is quick, but the output is too often incorrect for it to be recommendable. Nevertheless, it succeeds in some cases where set-intersection fails. If general characteristics of drawings which work with one method or the other can be identified, the appropriate method can be used, increasing the likelihood of a correct labelling. Some such general characteristics have been identified here, but as yet not enough to form the basis for a reliable choice.

For relaxation labelling, the set of seed probabilities which produces the greatest number of correct results is not derived from actual vertex label frequencies or edge label frequencies in the correctly-labelled test set and differs (in some cases, quite significantly) from them. This suggests that relaxation is, of itself, not an appropriate technique to use here, since moving some way away from "ideal" relaxation actually improves performance.

The relaxation algorithm implemented gives only one output labelling, not a set of reasonably good ones, and this is in itself a reason for preferring an alternative method. Despite previous failures with genetic algorithms and the good general reputation of relaxation as a way of tackling other labelling problems in computer vision, it appears that genetic algorithms are a more promising line of investigation (although there remains doubt about whether they are fast enough for an interactive application).

A further practical disadvantage of all of the new methods suggested in this Chapter is that, using heuristics or probabilities, they require tuning constants. The optimal values of such constants may vary from one set of drawings to another, and determining them is a time-consuming process.

It is apparent that labelling is neither an entirely local problem nor entirely a combinatorial problem. For example, considering the geometry makes it evident that, in the absence of occluding T-junctions, for any three consecutive edges d, e and f bounding a region, if e occludes the corresponding face, at least one of d and f must also occlude the corresponding face, since it is not possible for both end vertices of e to lie in the plane of the face. Current labelling algorithms do not use this path-consistency fact, with the result that errors such as the one in Figure 2.11 cannot be ruled out. Further investigations should consider the influence of geometry and of neighbouring junctions on the choice of labelling method.

Chapter 5

Parallel Lines

5.1 Introduction

Groups of lines in the drawing which are intended to be parallel in 3D are identified, in order that the merits of candidate symmetries and regularities in the drawing can be evaluated, and in order to generate a 3D geometry with edges parallel where the corresponding lines are parallel.

This problem is non-trivial. Consider the drawing in Figure 5.1. It is evident that edges A, B and C should be parallel, and any reasonable process will detect this. It is also evident that edges D, E and F should be parallel; however, depending on the quality of the drawing, D and E may be closer to G than to F, and a naïve algorithm may make D, E and G parallel instead.

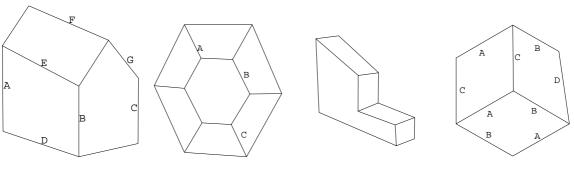


Figure 5.1: House Figure 5.2: Figure 5.3: L- Figure 5.4: In-Hexagonal Block complete Bund-Frustum ling

In Figure 5.2, even if the 2D lines A, B and C are parallel, the corresponding 3D edges may not be. Other similarly-oriented lines (even if closer in angle to A or

C in the drawing) correspond to edges parallel to B.

There is also the less clear-cut problem of determining intention. Did the user who drew Figure 5.3 intend the top and bottom faces of the object to be parallel, or is the difference in angle deliberate?

Section 5.2 outlines some previous work in this area. Section 5.3 describes an attempt to reproduce one recent method for grouping parallel lines. Section 5.4 introduces a new idea for grouping parallel lines based on satisfying expectations. Section 5.5 introduces the relationship between junction labelling, edge convexity/concavity, turns at corners, and face planes; this is used both in bundling and later in this thesis (notably Chapter 10). Section 5.6 gives results of investigation into these ideas. Section 5.7 describes one use of parallel line grouping: an attempt to identify the groups of parallel lines which correspond to the three coordinate axes of a partially axis-aligned object.

5.2 History

Some authorities, notably Sugihara [163], define this problem away by strengthening the definition of general viewpoint to require that all pairs of lines parallel in 2D correspond to edges parallel in 3D. This implicitly disallows freehand drawing errors, and is rejected here.

Line parallelism is detected and used in other systems [38, 59, 90, 91].

Lipson and Shpitalni [90, 91] plot an "Angular Distribution Graph", a histogram of line angles, and detect the peaks by comparing their shape with Gaussian distributions. In all of the examples they give (derived from freehand drawings of normalons and semi-normalons) the peaks are distinct; since they specifically allow freehand drawing errors, it may reasonably be assumed that methods exist which distinguish overlapping hills from single hills with two summits. These are not described.

2D line parallelism was the only regularity identified by Grimstead's system [38]. The algorithm used for this process allocated angles of 2D lines to buckets, and merged nearby buckets until it was evident which lines should be parallel. It appears to be a discrete version of the method used by Lipson and Shpitalni. Again, detail is missing and it is not possible to reproduce Grimstead's work exactly. A straightforward attempt to reproduce this work led to an unreliable method which

introduced problems with some types of drawing—see Section 5.3.

As well as determining which edges may be parallel, there is the problem of determining a level of confidence in the hypothesis that they are. Traditionally in computer graphics [127] this confidence is quantified as a *figure of merit* for parallelism (in the range 0..1):

$$F(A \parallel B) = (\hat{\mathbf{a}}.\hat{\mathbf{b}})^{M_p},$$

where $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ are unit vectors along lines or edges A and B, and M_p is an arbitrary constant. Other equations are possible (e.g. Lipson [90] uses a half-Gaussian curve which also produces a figure in the range 0..1). The solid modelling tradition has been followed here; no experiments have been performed to compare the various equations, as I anticipate that differences are slight. RIBALD uses $M_p = 50$ for both 2D parallelism and 3D parallelism (adjusting M_p could in principle provide a method of tuning the system to allow for differences in the users' sketching ability; this has not been tested). It will be noted that calculation of this figure does not require that bundles of parallel lines have already been identified, so this idea can be used in stages of processing which precede bundling.

Figures of merit for other hypothesis are listed in Appendix D.

5.3 Reproduction of Bucketing

In order to be able to compare the ideas in this thesis with previous work, it was necessary to attempt to reproduce Grimstead's bucketing approach. Some assumptions were necessary. In this investigation, bucketing required lines which are to be grouped as parallel to be no more than 15° apart, and each group to be at least 15° from any other group. These angles are arbitrary, but sufficient to permit at least six groups of parallel lines in any drawing (Figure B.114, page 313, has six lines, none of them parallel to one another), and up to twelve in well-drawn drawings.

Variations on this method were investigated, but none avoided the fundamental problem with the method. For example, consider Figure 5.1. Depending on threshold settings, it is possible that F and G may be within the allowed angle below which buckets are merged, and made parallel; this is clearly nonsense geometrically.

On the basis of a comparison between this implementation and the ideas in the

next section, bucketing has been rejected. Details of this comparison are to be found in [172]. Bucketing was less reliable, and it can be noted that when bucketing fails, it gives no useful information about the sketch, whereas incomplete bundling (see for example Figure 5.4) still gives some useful information.

5.4 Partitioning into Bundles

The preference in this thesis is for methods which correspond to geometric intuition, and it is such a method which is introduced here. As before, lines are grouped together which are nearly parallel in 2D and which are expected to correspond to edges which are parallel in 3D. These groupings are made on the basis of expectation as well as on the fact of 2D parallelism, and take into account the drawing topology.

The algorithm for this is:

• Repeat

- if any unbundled edge *must* be parallel to a bundled edge then put it in the same bundle (see below)
- if any unbundled edge is very close to being parallel to a bundled edge then
 put it in the same bundle providing that this is possible (see below)
- for each face with unbundled edges
 - * if an unbundled edge is *close* to parallel with a bundled edge, and the topology *suggests* that they should be parallel (see below), then put it in the same bundle providing that this is possible
- if nothing has been bundled in this iteration but edges remain unbundled,
 then pick any remaining edge and allocate it to a new bundle

• Until all edges have been bundled

The topology requires that edges must be parallel if they derive from a single line split at an extended trihedral junction (see Figure 5.5), K-junction or non-occluding T-junction.

The topology *suggests* that edges should be parallel if:

- they appear separated by exactly one side on any face which contains exactly four more convex than concave corners (e.g. simple quadrilaterals, L-hexagons, T-octagons) as in Figure 5.6
- they appear separated by exactly one side on any pentagonal face as in Figure 5.7
- they appear on opposite sides of a simple 2n-gon (hexagon, octagon, ...) as in Figure 5.8
- they appear in a candidate feature instance in locations which are parallel in the corresponding generic feature template (see Chapter 6)

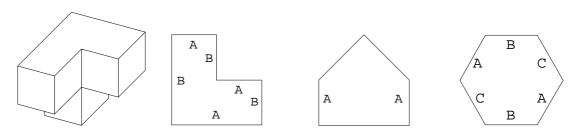


Figure 5.5: Ex- Figure 5.6: L- Figure 5.7: Figure 5.8: 2n-gon tended Trihedral hexagon pentagon

Junction

Two edges *cannot* be parallel if:

- bundling them together would cause two edges meeting at a vertex (other than a K-vertex or extended trihedral vertex) to be in the same bundle,
- bundling them together would cause an edge leaving a face (at a vertex other than a K-vertex or extended trihedral vertex) to be in the same bundle as an edge which is part of that face,
- bundling them together would cause two edges leaving a face on the same side of the face plane but in opposed directions in the drawing (such as A and C in Figure 5.2) to be in the same bundle.

The overall algorithm as described here contains three loops: (a) the worst case number of iterations of the main loop is the same as the number of edges, if each bundle size is 1, (b) each iteration considers each unbundled edge, (c) detecting

whether adding a new edge to a bundle is permitted is proportional to the bundle size, which in the worst case is proportional to the number of edges. Since worst cases (a) and (c) are mutually exclusive, it could be argued that the algorithm is theoretically $O(n^2)$ —(bundle size times number of bundles) is proportional to the number of edges—but this argument is unconvincing. However, a limit of $O(n^2)$ can be justified on other grounds. The test within the third loop is whether or not two edges can be in the same bundle. If the results of this test are stored for future reference, the test need only be made $O(n^2)$ times irrespective of the surrounding control structure.

Some lines may escape bundling as the expectation which would make them parallel is not amongst those listed. A postprocessing stage which considers "singletons" (lines not bundled with any other line) and attempts to bundle them with similarly-oriented lines (the rules determining when bundling is not permitted apply here too) may improve results. This is investigated in Section 5.6.

5.5 Corners and Face Planes

One of the criteria required for assessing whether two edges can be in the same bundle requires knowledge of whether an edge is above or below the plane of a face it leaves. In the trihedral domain, this can be deduced from the labelling. Table 5.1 exhausts the possibilities: the columns consider a loop of sides of a face, where two edges form incoming and outgoing sides at a corner, and the third edge leaves the face (either above or below the plane of the face) at the corner.

label	incoming	outgoing	turn	leaving line	direction	
Lba,Wbca,Yccc	convex	convex	right	convex	below	
Lab,Lac,Lcb,Wcdc,Yabd	convex	convex	left	concave	below	
Lab,Lac,Lcb,Wcdc,Yabd	convex	concave	right	convex	above	
Lab,Lac,Lcb,Wcdc,Yabd	concave	convex	right	convex	above	
Lbd,Lda,Wdcd	concave	concave	left	convex	above	
Lbd,Lda,Wdcd	concave	convex	right	concave	below	
Lbd,Lda,Wdcd	convex	concave	right	concave	below	
Yddd	concave	concave	right	concave	above	

Table 5.1: Corners and Face Planes

At the time of writing, extension of this idea into the tetrahedral domain has not been completed. RIBALD detects many of the more common situations, but does not include an exhaustive list of possibilities. Again, this information is derived from the labelling.

This information is also useful in topological reconstruction—see Chapter 10.

5.6 Results and Recommendations

5.6.1 Correctness

Variant	Close	Very Close
Strict	$\hat{\mathrm{a}}.\hat{\mathrm{b}}>0.99$	$\hat{\mathrm{a}}.\hat{\mathrm{b}}>0.9999$
Normal	$\hat{\mathrm{a}}.\hat{\mathrm{b}}>0.98$	$\hat{\mathrm{a}}.\hat{\mathrm{b}}>0.999$
Lax	$\hat{\mathrm{a}}.\hat{\mathrm{b}}>0.95$	$\hat{\mathbf{a}}.\hat{\mathbf{b}}>0.99$

Table 5.2: Bundling Variants

To investigate the performance of the idea of bundling, RIBALD implements three variants which differ in their definitions of "close" and "very close"—see Table 5.2. Attempted singleton removal was also implemented as an option. The test set comprised all drawings for which any labelling variant (Chapter 4) produced the preferred result, minus Figure B.149, for which there is no correct answer.

Results are summarised in Table 5.3.

Variant	Sing.	< -2	-2	-1	0	0*	1	2	> 2
Strict	N	2	1	3	374	7	40	14	40
Normal	N	3	5	6	403	3	23	14	24
Lax	N	6	10	23	399	10	15	5	12
Strict	Y	5	0	19	361	10	37	14	35
Normal	Y	6	4	27	386	4	21	17	16
Lax	Y	8	12	42	380	12	12	8	6

Table 5.3: Bundling Results

The columns indicate the difference in number of bundles produced from the number present in a human interpretation of the drawing (i.e. the column headed "2" lists the number of times bundling produced two more groups of parallel lines than were expected). The column headed "0" indicates correct results; the column headed "0*" indicates the number of times that the number of groups was correct but the group members wrong.

The results could be considered misleadingly optimistic, as about half of the test drawings were produced as illustrations and lines are parallel where edges are intended to be parallel; these test the cases where lines cannot be parallel for the reasons outlined above, but not the cases where lines must be deduced to be parallel. Even so, it is clear that the "strict" version performs less well than the "normal" or "lax" versions.

It appears that singleton removal does more harm than good—when it makes a difference, it more often bundles together lines which should not be parallel than lines which should be.

All six of the drawings which always appear in the right-hand column are Archimedean solids—Figure B.132 and others like it would also defeat any of the other approaches outlined in Section 5.2. More seriously, all variants identified the two lines marked * in Figure 5.9 as being parallel, which indicates that there are further logical restrictions on which lines can be parallel in addition to those already identified (and also shows the difficulty of avoiding accidental coincidences).

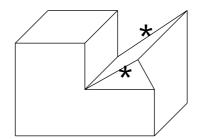


Figure 5.9: Accidental Coincidence

Otherwise, to a large extent, deviations from perfect results can be attributed to the unsolved problem of determining intention. Some lines, intended to be parallel, are too far apart in orientation; the "normal" version of the program fails to bundle these correctly, but the "lax" version produces the intended output. Other lines, not intended to be parallel, are too close in orientation; the "normal" version of the program incorrectly bundles these, but the "strict" version produces the correct output.

Results in subsequent chapters are obtained using the "normal" version of parallel line bundling without singleton removal.

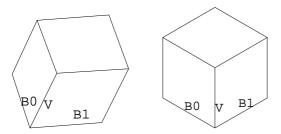
5.6.2 Timings

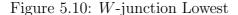
RIBALD takes 0.14 seconds to bundle Figure B.132, the drawing with most lines. It takes 0.03 seconds to bundle Figure B.74, the drawing with the largest bundles of parallel lines; for Figure B.456, with slightly fewer lines but also fewer clues, it takes 0.04 seconds, the longest for any realistic engineering drawing. Since timing is clearly fast enough for an interactive system, no experiments have been performed to determine how close bundling is to $O(n^2)$ in practice.

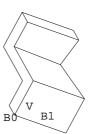
5.7 Special Sets of Parallel Lines

It can be noted that many objects are designed and drawn in such a way that they rest on a horizontal plane. This is particularly true of normalons and semi-normalons such as the ones portrayed in Figures B.486 and B.491. Such objects usually also have a vertical axis, perpendicular to the plane; this axis is often drawn vertically in the sketch, as it is in these two figures.

To make use of this observation, three special bundles are identified: V, which corresponds to "vertical", and B_0 and B_1 , which correspond to the "base" of the object. These are used later to make geometric hypotheses—see Chapter 11. Figures 5.10 and 5.11 illustrate the four cases detected by the algorithm for identifying V, B_0 and B_1 ; the full algorithm is given in [178].







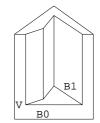


Figure 5.11: L-junction Lowest

Chapter 6

Features

6.1 Introduction

It was noted in Chapter 2, without any definition of "feature", that any identification of features implied by the line drawing will simplify the process of topological reconstruction of the object, and possibly also make this process more robust.

For the purposes of this thesis, a feature is a commonly-occurring localised configuration of lines with a recommended interpretation; it is in effect a form feature. Han [44, 45] distinguishes form features, descriptions of shape with no implied relation to function or manufacturing method, and which may be additive or subtractive, from machining features, which are produced by a specific machining process and thus necessarily subtractive, noting that engineering research has concentrated on the latter (rapid prototyping devices, an apparent exception, do not in general use feature-based models [84]). Han [44] also notes the conceptual advantage of using form features during the design process and subsequently converting the finished design to manufacturing features.

This chapter describes two types of localised configurations which can usefully be identified, corresponding to hole loop features (bosses, pockets and through holes) and slots.

Section 6.2 describes previous relevant work in feature recognition. Section 6.3 describes how feature recognition may be used in a system based on the other ideas in this thesis. Section 6.4 describes slot features. Sections 6.5 and 6.6 describe methods for detecting and classifying hole loop features; this is the major new idea

in this chapter, and the results of testing it are presented in Section 6.7.

6.2 History

There is considerable literature on recognising features from complete CAD models, but much of it is not relevant here. Detecting the presence of a particular feature in a complete CAD model is straightforward—Han [44] lists numerous algorithms for feature detection, classifying the most promising into four general categories: graph pattern matching, convex hull decomposition, cell-based decomposition and hint-based reasoning. Qamhiyah et al [135] use repeated graph pattern matching to extract multiple features, reconstructing the CAD model of the remainder of the object at each stage, until left with a CAD model of the "featureless" object. Gupta et al [41] note that the number of alternative feature-based descriptions of an object is exponential in the number of features identified, and attempt to obtain a set of primary features from which more complex features can be built. Han et al [47] emphasise that the choice between alternative feature-based descriptions cannot be divorced from the problem of manufacturing the object modelled, nor even [46] from the particular manufacturing equipment available, and these considerations are clearly beyond the scope of this thesis.

It should be noted that the problems addressed in this literature are not ones with which this thesis is concerned—they assume a complete solid model of the object. Also, one of the form features considered in this chapter, the boss, is an additive feature and thus not considered by work on machining features.

However, it is clear that the problem which much recent work on machining features is intended to address, that of multiple interpretations [41, 45], is one which could also occur with form features. It follows that form features should only be identified if, by doing so, a specific problem of interpretation can be solved or avoided; form features should not be identified merely because it is possible to do so.

Identification of machining features in two-dimensional drawings is an item of current research. Meeran and Taib [107] list ten systems prior to their own, and identify their limitations (most only recognise rotational parts or extrusions) and deficiencies.

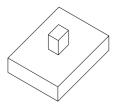
With their own system, Meeran and Taib [107] take as input three orthogonal

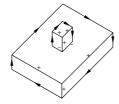
2D views of an object. They attempt to locate three sorts of machining features, which they call type I (slots, and also steps and notches), type II (hole loop features, restricted to holes and pockets), and type III (side pockets). These features must be parallel to the base plane and aligned with one of the remaining two principal axes. Since their algorithm is graph-based, and ignores the object geometry, the object may contain curved faces (their examples include objects with axially-aligned edge blends). The algorithm proceeds by creating a graph of the object, eliminating first outer edges and then leaf nodes (vertices with only one remaining edge). It uses heuristics to match what remains against profiles for their three feature types. It is fast, and moderately reliable, and capable of distinguishing multiple features in the same object. In one example reported, their system detected 20 features (5 type I, 13 type II and 2 type III) from three 3D orthographic views of an object in 0.5 seconds. Some objects defeat their system—this occurs most commonly when alternative interpretations are possible of the original 2D views, but they also report occasional unexplained interpretation of type III features as type I.

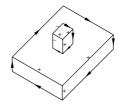
Although not directly relevant to the problems of this thesis, Meeran and Taib's work provides useful insight, firstly into the types of feature which occur sufficiently often in engineering practice for automated recognition to be worthwhile, and secondly in that feature recognition is a local template-matching process which (almost inevitably) involves heuristics.

6.3 Implementation

As noted in Chapter 2, ordering the components of a line drawing interpretation system presents a problem. Clearly, feature information helps the labelling process—feature figures of merit are used both in choice heuristics and in pruning heuristics (see Chapter 4.4.1). Without cofacial configurations, junction labels cannot be propagated across the empty space which separates the outer edges of a face from a hole loop. For example, in Figure 6.1, the edges form two separate subgraphs. It is visually obvious which edges of the boss should be concave and which occluding (Figure 6.2), but a purely topological labelling process would give equal merit to the labellings in Figure 6.3 (a fourth trihedral labelling, in which there are no concave edges, can be discarded as representing two objects rather than one).







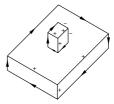


Figure 6.1: Boss

Figure 6.2: Correct

Figure 6.3: Incorrect

With all feature types identified in this chapter (slots, holes/pockets and bosses), candidate features are identified before labelling, and a figure of merit assigned to each. The figure of merit of a candidate feature reflects the likelihood that the drawing shows a skewed view of the hypothesised feature—junction angles will be distorted, but parallel lines should appear approximately parallel. The frequency of occurrence of a feature in engineering objects should also contribute towards the figure of merit.

Labellings which match the line labels required by a candidate feature are given extra merit based on the merit of the candidate feature (see Chapter 4). Candidate feature templates are also used as parallel line suggestions (see Chapter 5).

Later, after both line labelling and bundling of parallel lines, candidate features are discarded if they require line labels which do not match those in the preferred labelling or require impossible groupings of parallel lines.

Thus, for both slots and hole loop features, RIBALD divides the feature identification process into two. Candidate features are identified after identification of the background region (a line which must be occluding because it lies on the background region should not be included in a candidate feature in which the line is expected to be concave or convex) and after line labelling (for hole loop features, this is described in Section 6.5). Candidate features are accepted or rejected, and their implications determined, after line labelling and bundling of parallel lines and before inflation (for hole loop features, this is described in Section 6.6).

6.4 Underslots and Valleys

Slots are common in engineering objects. Identification of such features improves the reliability of topological reconstruction in Chapter 10—in particular, without

special-case knowledge of how to handle underslots, reconstruction from drawings containing them is particularly unreliable—and this is the justification for the inclusion of slots as features to be recognised. In practice, slot features would usually be labelled and bundled correctly even without being treated as a special case, but since the information is available it is also used in labelling heuristics (Chapter 4) and in bundling parallel lines (Chapter 5).

For manufacturing purposes, it matters little whether slots are on the top or the bottom of the object. However, this clearly affects the appearance as seen in a line drawing, so the two are distinguished as underslots and valleys (see Figures 6.4 and 6.5). RIBALD looks for these two types of features by trying to match the region around each T-junction in the drawing with the templates shown in these two figures; if one is found, it is given a figure of merit which is the product of a tuning constant (F_u or F_v) and the figures of merit for parallelism of edge pairs which are parallel in the template.

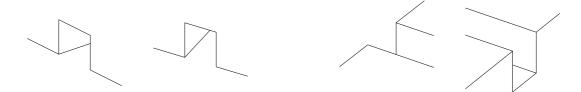


Figure 6.4: Templates for Underslots

Figure 6.5: Templates for Valleys

Since each T-junction is analysed once to determine whether its neighbourhood matches a template, and template matching takes (at most) a fixed time, identifying underslots and valleys takes O(n) time.

6.5 Cofacial Configurations

Two problems to be solved are: "how to detect hole loops?" and "what to do about them?". Even if a hole loop can be identified (which is not necessarily straightforward—for example, Figure B.429 (page 326) would defeat Puppo's algorithm [134], which does not distinguish occluding and non-occluding T-junctions), the question arises whether the hole loop corresponds to a boss, a pocket or a through hole (or perhaps to none of these).

These problems are addressed by identifying (and allocating figures of merit to)



Figure 6.6: Templates for Holes and Pockets

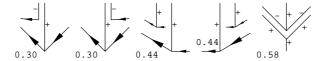
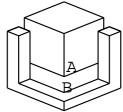


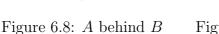
Figure 6.7: Templates for Bosses

candidate cofacial configurations. A candidate cofacial configuration is a configuration of junctions and lines which matches one of the templates in Figures 6.6 and 6.7. These templates centre on an inner junction j_A which is within the angle of two specified lines meeting at an outer junction j_B . Candidate cofacial configurations are sought for any drawing with more than one subgraph, and must meet all of the following conditions:

- ullet The subgraph G_A containing junction j_A and the subgraph G_B containing junction j_B must be different
- G_A must not be known to be behind G_B (i.e. occluded by it at a T-junction anywhere in the drawing)—see Figure 6.8
- ullet An imaginary line between j_A and j_B does not cross any actual line in the drawing—see Figure 6.9
- ullet j_A and j_B must match the junction types of the inner and outer junctions shown in one of the templates
- j_A is within the angle of the appropriate two lines leaving j_B
- No other junction is within the parallelogram bounded by j_B , the two lines leaving j_B , and j_A —see Figure 6.10 (in practice, to allow for roundoff error in geometric tests, the two junctions at the other end of the lines from j_B must also be allowed to be in the box, even though ideally they cannot be)

In practice, it is usually the case that sides of the pocket or boss are parallel to sides in the outer loop of the face. This is necessarily the case where the face and





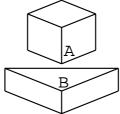


Figure 6.9: Line Crosses Template

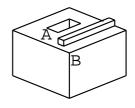


Figure 6.10: Junction in Parallelogram

feature are axis-aligned (see, for example, Figure B.438, a stylised drawing taken from life of a house with a recessed balcony), and is common even when they are not (see, for example, Figures B.458 and B.495). This can be used as a clue to the merit of a candidate feature. Another clue is how often the configuration indicates the feature in practice. Each template identifies which lines should be parallel for a "perfect" pocket or boss, and how often the configuration indicates the feature in the set of test drawings in Appendix B (the number shown in the Figures 6.6 and 6.7). RIBALD uses as the figure of merit for a candidate cofacial configuration the product of the number in the template and the figures of merit for parallelism for the corresponding actual lines in the drawing. The predicted labellings in the templates are the basis of one of the most successful heuristics used in selecting junction labellings in Chapter 4.

The combinatorial contents of these figures could be generalised to just two rules, which in principle could remove the need for templates:

- two parallel lines, one convex and the other concave, from different subgraphs and with nothing between them suggest a boss
- two parallel lines, one convex and the other occluding, from different subgraphs and with nothing between them, suggest a hole or pocket.

However, determining frequencies of occurrence for parts of drawings matching templates is straightforward, whereas deriving a generalised frequency of occurrence measure is not, so RIBALD uses the template-matching approach.

Since each pair of junctions in the drawing is matched against templates, and geometric tests associated with template-matching require that each junction and line in the drawing is tested to ensure that it does not conflict with the template, there are $O(n^2)$ candidate cofacial configurations and identification of them takes $O(n^3)$ time.

6.6 Hole Loops from Cofacial Configurations

Obviously, a drawing with only one subgraph cannot contain a hole loop feature. However, as seen in Chapter 2.5.1, the presence of more subgraphs does not always imply that a hole loop feature is present. The problem to be solved is to identify whether a subsidiary subgraph in the drawing implies an object with a boss, a hole or a pocket, or something else.

The cofacial configurations identified in the previous section can be used as clues in making this decision. The presence of candidate cofacial configurations matching hole/pocket or boss templates suggests that the subsidiary subgraph is a hole/pocket or boss; the absence of such template matches is an indication that it is not. Other clues are: that at least one subgraph is not a hole loop; that if the subsidiary subgraph is contained entirely within a single region of the drawing, a hole loop must be present; and that if lines in the subsidiary subgraph are on the drawing boundary, the subgraph cannot be a pocket and is somewhat less likely to be a boss—the higher the proportion of drawing boundary lines the subgraph contains, the less likely it is to correspond to a boss (in Figure B.429 it is evident which of the five subgraphs is not a boss).

RIBALD currently assumes that hole loops correspond to holes, pockets or bosses, and cannot cope correctly with counterexamples such as Figures B.430 and B.431 (in practice, RIBALD cannot label Figure B.430 and confidently identifies a pocket in Figure B.431); any subsidiary subgraph which is neither a hole or pocket nor a boss (such as Figure B.553) is assumed not to be a hole loop. After edge bundling and before inflation, and after removing candidate cofacial configurations which do not match the preferred labelling, RIBALD identifies three merit figures for each subgraph s, the figures for it being a pocket or hole (P_s) , a boss (B_s) or something else, not a hole loop (O_s) .

Each subgraph is classified as indicating a hole/pocket, a boss, or something else, using the following algorithm (where F_o , F_b and F_c are tuning constants, chosen because they work reasonably well rather than derived from any theory—see

Appendix C):

- For each candidate cofacial configuration (as identified in Section 6.5)
 - if the expected labelling (as determined by the template) for both the inner and outer junctions match those actually produced by line-labelling, accept this cofacial configuration; otherwise delete it
- ullet Count the number of boundary edges n_{B_s} in each subgraph s
- ullet For each subgraph s
 - If n_{B_s} is non-zero, set $o=\frac{n_{B_s}}{n_E}$, $O_s=\frac{1+o}{2}$, $P_s=0$, $B_s=\frac{1-o}{2}$
 - Else if the subgraph is contained entirely within one region of the drawing, set $O_s=0$ and $P_s=B_s=\frac{1}{2}$
 - Else (n_{B_s} is 0 but the subgraph extends to a region boundary of another subgraph) set $O_s=P_s=B_s=\frac{1}{3}$
 - Adjust merit for O_s by adding F_o (see [178])
- For each accepted cofacial configuration
 - Determine outer and inner subgraphs o and i and merit M
 - Adjust merit for O_o by adding M^{F_c}
 - Adjust merit for B_i or P_i according to template by adding M
- ullet For each subgraph s
 - if n_{B_s} is non-zero, set $P_s=0$, divide B_s by $F_b^{(n_{B_s})}$ and renormalise ${\cal O}_s+P_s+B_s$
 - determine whether the subgraph indicates a pocket or hole, a boss, or a non-hole-loop interpretation, according to which of P_s , B_s or O_s is highest
- For each accepted cofacial configuration
 - if the outer subgraph is not a hole loop and the inner subgraph indicates a
 pocket, make a pocket mouth from this configuration by reidentifying the

face containing the inner junction and its associated lines as an inner loop of the face containing the outer junction and its associated lines (see [178] for details)

— if the outer subgraph is not a hole loop and the inner subgraph indicates a boss, make a boss from this configuration by reidentifying the face containing the inner junction and its associated lines as an inner loop of the face containing the outer junction and its associated lines (see [178] for details)

Since there are $O(n^2)$ candidate cofacial configurations, this process takes $O(n^2)$ time.

The problem of distinguishing holes from pockets remains unresolved. Where the bottom of the feature is visible in the drawing, the problem is essentially geometric in that it depends on the depth of the feature—for example, the feature in Figure B.413 is clearly a through hole, but if it were less deep it would be a pocket. Neither the template-matching described here, nor the labelling algorithms described in Chapter 4, refer to geometry in this way. Where the bottom of the feature is not visible, even this clue is absent. RIBALD assumes that hole/pocket features in the end caps of extrusions and frusta are holes, and that all other hole/pocket features are pockets; this assumption, although often wrong, has the merits of simplicity and predictability.

6.7 Results

Identification of underslots and valleys is straightforward—either a drawing contains such a configuration or it does not. As seen above, classification of hole loops as bosses, holes/pockets, or "other" is less rigorous, and has been investigated in more detail. RIBALD classifies a hole loop according to which of B_s (boss), Ps (hole/pocket) or O_s ("other") is numerically greatest. The results are presented in Tables 6.1, 6.2 and 6.3.

In addition to those cases listed in Table 6.3, Figures B.74, B.487, B.141, B.547, B.223, B.90, B.108 and B.109 also contain multiple subgraphs; there is no indication in any of these that either holes, pockets or bosses are present, and in all cases all subgraphs are correctly classified as "other".

The failures with Figures B.420 and B.460 occur when nothing in the drawing matches any of the templates (in Figure B.420, the central hole loop does not match any template because lines from the other holes are within the parallelograms of each potential template). "Switches" such as Figure B.431, being simultaneously pockets and bosses, are beyond the capabilities of the method described in this chapter. The configuration from which RIBALD produces a pocket in Figure B.89 is clearly not a pocket according to the criteria given here—this must be an implementation error.

RIBALD takes 0.44 seconds to identify the candidate cofacial configurations in Figure B.538, but for more typical drawings timings are satisfactory. The secondworst case is Figure B.420, with six subgraphs; RIBALD takes 0.22 seconds to identify the cofacial configurations in this case. In general, the time taken is a function both of the number of lines in the drawing and the number of subgraphs—the other time-consuming cases are Figure B.456 (0.08 seconds), Figure B.429 (0.06 seconds) and Figures B.460, B.454, B.455 and B.419 (all 0.05 seconds). In no case does classifying subgraphs as holes/pockets or bosses take measurable time.

As noted in Chapter 4, trying to match the expected configurations is a good heuristic for choosing between labellings.

Drawing	Feature	O_s	P_s	B_s	Correct?
B.421	Hole	0.000508	0.989576	0.009915	
B.438	Pock	0.000020	0.999583	0.000397	$\sqrt{}$
B.536	Hole	0.000020	0.999583	0.000397	$\sqrt{}$
B.538	Hole	0.000000	1.000000	0.000000	
B.540	Hole	0.000020	0.999583	0.000397	
B.513	Hole	0.001343	0.972472	0.026185	
B.469	Hole	0.000005	0.999892	0.000102	
B.472 (1)	Hole	0.023528	0.517679	0.458793	
B.458	Hole	0.023827	0.511543	0.464630	
B.460 (1)	Hole	0.000804	0.983525	0.015672	
B.454 (1)	Hole	0.022500	0.538750	0.438750	
B.455 (1)	Hole	0.022500	0.538750	0.438750	
B.456	Hole	0.000005	0.999893	0.000101	
B.495	Hole	0.002801	0.942570	0.054629	
B.498	Hole	0.001064	0.978178	0.020758	
B.425	Hole	0.022500	0.538749	0.438751	
B.426 (1)	Hole	0.001099	0.977476	0.021425	
B.426 (2)	Hole	0.000021	0.999572	0.000407	
B.415	Hole	0.001064	0.978183	0.020753	
B.416	Hole	0.001064	0.978183	0.020753	
B.433	Hole	0.000020	0.999582	0.000397	
B.434	Hole	0.000020	0.999582	0.000397	
B.435	Hole	0.010417	0.786451	0.203132	
B.436 (1)	Hole	0.000020	0.999582	0.000397	
B.436 (2)	Hole	0.000020	0.999582	0.000397	
B.413	Hole	0.000020	0.999583	0.000397	
B.414	Hole	0.000030	0.999377	0.000593	
B.419	Hole	0.013103	0.986897	0.000000	$\sqrt{}$
B.420 (1)	Hole	0.001184	0.975730	0.023087	
B.420 (2)	Hole	0.000433	0.991125	0.008443	$\sqrt{}$
B.420 (3)	Hole	0.000430	0.991185	0.008385	
B.420 (4)	Hole	0.001182	0.975759	0.023059	√
B.420 (5)	Hole	0.025000	0.487500	0.487500	×
B.439	Hole	0.000508	0.989576	0.009915	
B.440	Hole	0.000185	0.996210	0.003606	V
B.441 (1)	Hole	0.024525	0.497233	0.478242	
B.441 (2)	Hole	0.024525	0.497233	0.478242	V

Table 6.1: Detection of Holes/Pockets

Drawing	Feature	O_s	P_s	B_s	Correct?
B.422	Boss	0.000080	0.001568	0.998352	
B.423	Boss	0.000488	0.009516	0.989996	
B.437	Boss	0.125602	0.000000	0.874398	
B.460 (2)	Boss	0.025000	0.487500	0.487500	×
B.424 (1)	Boss	0.378699	0.000000	0.621301	
B.424 (2)	Boss	0.003148	0.061388	0.935464	
B.427 (1)	Boss	0.122194	0.000000	0.877806	
B.427 (2)	Boss	0.337473	0.000000	0.662527	
B.428 (1)	Boss	0.118849	0.000000	0.881150	
B.428 (2)	Boss	0.333651	0.000000	0.666349	
B.428 (3)	Boss	0.333651	0.000000	0.666349	
B.429 (1)	Boss	0.116620	0.000000	0.883380	
B.429 (2)	Boss	0.331102	0.000000	0.668898	$\sqrt{}$
B.429 (3)	Boss	0.331102	0.000000	0.668898	
B.429 (4)	Boss	0.331102	0.000000	0.668898	

Table 6.2: Detection of Bosses

Drawing	Feature	O_s	P_s	B_s	Correct?
B.472 (2)	n/a	0.542969	0.000000	0.457031	
B.451	n/a	0.772318	0.000000	0.227682	
B.454 (2)	n/a	0.737500	0.000000	0.262500	
B.455(2)	n/a	0.737500	0.000000	0.262500	
B.553	n/a	0.715625	0.000000	0.284375	
B.431	n/a	0.003711	0.689830	0.306460	×
B.442 (1)	n/a	0.634375	0.000000	0.365625	
B.442 (2)	n/a	0.695312	0.000000	0.304688	
B.89 (1)	n/a	0.686607	0.000000	0.313393	
B.89(2)	n/a	0.686607	0.000000	0.313393	
B.89 (3)	n/a	0.026764	0.954025	0.019211	×
B.405	n/a	0.925078	0.037461	0.037461	

Table 6.3: Other Drawings with Multiple Subgraphs

Chapter 7

Inflation

7.1 Introduction

It is desirable to *inflate* the visible part of the object—to add *approximate* depth coordinates for each vertex appearing in the line drawing—before reconstructing the back of the object. In later stages of processing, depth information will be used in estimating the merit of various hypotheses made about the drawing.

The 2D coordinates x_v, y_v of each vertex v are known (these coordinates are those in the original drawing and may or may not be accurate), as are the vertex pairs joined by each edge, and the loops of edges forming each face. Additionally, which pairs of edges are presumed to be parallel in 3D may be known. Inflation uses any of this information which may be relevant, translates it into *compliance functions* which can be expressed as equations, and finds the best solution to the resulting system of equations. The outputs are depth (z-)coordinates for each visible vertex and for each point at which a partially-occluded edge disappears from view.

The methods which have been tried can be classified into two basic approaches. Firstly, information may be limited to that which can be translated into a linear system of equations, and the optimum solution found by linear algebra. Alternatively, non-linear equations may be included, with the solution being found using an iterative optimisation process. There are also two possible targets. Here, an approximate geometry is preferred, being the "best" fit (usually, as here, a least squares fit) to all compliance functions. A fully-correct geometry in which, for example, all faces are exactly planar may require adjustment of the x_v, y_v coordinates to enforce this

and may require analysis to detect and eliminate incompatible compliance functions from the solution; neither of these is desirable at this stage.

The requirements for the output of an inflation component, in descending order of desirability, are:

- the depth ordering of adjacent pairs of visible vertices must be correct;
- depth ordering must not be sensitive to inaccuracies in the drawing;
- depth information must be calculated in a fraction of a second for drawings of typical engineering components;
- depth information should be based on as little prior processing of the drawing as possible (depth information is to be used to test hypotheses, so it should not presuppose these hypotheses if this can be avoided)
- depth information should be as good an interpretation of the drawing as is possible while achieving the other objectives.

This chapter does not describe "beautification", in which an existing 3D model is improved by adjustment of face equations or vertex coordinates. All optimisation methods which rely on a preprocessing stage to identify approximate depth coordinates (thereby ensuring that downhill methods start in the right valley) are included in the latter category, described in Chapter 11, as are all methods which iteratively detect and remove incompatible compliance functions and all methods which change the x and y coordinates of junctions visible in the drawing. Many past systems contain only one stage which determines z-coordinates and therefore do not make this distinction. Here, discussion is split between Chapter 11 (which emphasises optimisation-based methods) and the description below (which emphasises analytical methods).

Section 7.2 summarises prior work in general terms.

Section 7.3 considers compliance functions in more detail. With the exception of Junction Label Pairs (JLP, Section 7.3.17), all of the compliance functions listed here have been used successfully in other systems. The JLP approach, which is new, is preferred because it can be used directly in a system of linear equations in which the unknowns are the values sought, and because the results are intuitively plausible.

Section 7.4 considers a single recent depth estimation component in more detail, identifying weaknesses which make it inappropriate for initial inflation.

Section 7.5 describes a component which is better-suited to the specific requirements of preliminary inflation. Use of JLP as the primary compliance function in any inflation system is new (it also appears that a z-coordinate linear least-squares approach using corner orthogonality (Section 7.3.8) and line parallelism (Section 7.3.5) is new).

Section 7.6 demonstrates that JLP gives acceptable results and compares its effectiveness with corner orthogonality, which shares its merits of simplicity and intuitive plausibility. The effects of some secondary compliance functions on the quality of output are also analysed.

7.2 History

The simplest linear method, described below, is to use a system of equations linear in just one set of variables z_v , where z_v is the depth coordinate of vertex v.

A more complex method, due to Grimstead [38], is to include coefficients in each face equation $P_f x_v + Q_f y_v + z_v + C_f = 0$ for any combination of vertex v and face f where the vertex lies on the face; the output variables are P_f , Q_f and C_f for each face and z_v for each vertex.

Optimisation aims to minimise an objective function (a weighted sum of compliance functions) describing the drawn object. Optimisation methods can be further subdivided into those which adjust all vertex depths simultaneously [16, 15, 83, 101] (usually using a black-box optimisation algorithm) and those which adjust the vertex depths one by one [91, 92]. A further refinement, originated by Leclerc and Fischler [83], is to introduce into the objective function a parameter λ which increases from 0 to 1 as the optimisation process progresses; the overall objective function becomes $F = F_A + (1 - \lambda)F_B + \lambda F_C$, where F_A are those parts of the objective function which must always be satisfied, F_B are those parts of the objective function used to inflate the flat line drawing into 3D, drawing the solution towards the global minimum, and F_C are those parts of the objective function used to fine-tune the solution once it is securely close to the global minimum. Although successful in practice, this refinement blurs the distinction between the problems of inflation

 (F_B) and beautification (F_C) .

Barrow and Tenenbaum [2] use an iterative optimisation scheme which slides the z coordinates of vertices in and out along the z-axis in order to maximise or minimise a single compliance function. They prefer one of three global measures of regularity:

- sum of squares of angles between faces
- sum of squares of cosines of angles between faces
- sum of squares of $(2\pi \text{sum of angles at a vertex})$

They report that all three measures produce similar results.

Lipson and Shpitalni [91] provide a list of compliance functions which they have used: face planarity; line parallelism; line verticality; isometry; corner orthogonality; skewed facial orthogonality; skewed facial symmetry; line orthogonality; minimum standard deviation of angles; face perpendicularity; "prismatic face"; line collinearity; planarity of skewed chains. These are described in the following section. The compliance functions are weighted according to their degrees of accuracy in free-hand drawings. Their system produces depth values by performing a least-squares fit against these. They examined several optimisation methods and prefer cyclic application of Brent's method [6] to each vertex in turn. They stress the need for reasonable preliminary estimates in order to minimise optimisation time.

7.3 Compliance Functions

This section describes individual compliance functions, including those listed above. Mathematical detail is included only for those which are self-contained or which could form part of a linear system.

7.3.1 Approach: Distance Propagation along Three Axes

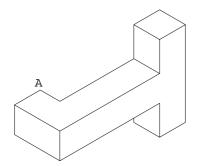
Lamb and Bandopadhay [77] point out that if three bundles of lines (see Chapter 5) can be identified as corresponding to three perpendicular axes of the object, the relative spatial locations of vertices in the object can often be determined simply by

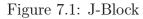
propagating distances along those axes. They note that occluding T-junctions and non-axially-aligned edges form barriers to distance propagation.

This sound and intuitively correct method works for all single-subgraph normalons and for many single-subgraph semi-normalons. However, it has not been investigated further in this thesis as (a) it is of limited applicability, and the cases for which it works are those handled well by more general methods, (b) it relies on the assumption of isometry (see Section 7.3.7) and (c) by switching to object-axis-relative rather than viewer-relative coordinates, it goes against the ideal of maintaining the original user input data throughout the interpretation process.

7.3.2 Approach: Direct Use of Mirror Symmetry

Vetter and Poggio [180] note that if a polyhedral object is known to have an axis of bilateral symmetry, the entire object can be recognised from one 2D drawing, given the general viewpoint assumption, and providing that at least four pairs of bilaterally symmetric points can be determined. This is not true of reconstruction—hidden atoms bisected by the mirror plane cannot be deduced from mirror symmetry alone. For example, although it is obvious that there should be an edge descending behind the object from vertex A in Figure 7.1, neither the original drawing nor its reflected equivalent contains this edge. Other methods must be used to deduce its presence.





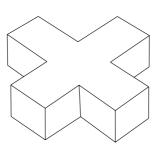


Figure 7.2: X Block

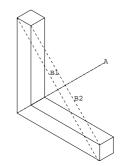


Figure 7.3: L Block

My observations on implementing this approach found that it is not an improvement on the more general methods described here and in Chapter 11. Errors in the generated reflected geometry magnify inaccuracies in the original drawing. It is possible to include the z-coordinates predicted by this approach in a linear system, but in view of the generally poor performance of mirror planes in predicting geometry, this idea is not regarded as promising and has not been investigated in detail.

7.3.3 Approach: Planar Constraints

Sturm and Maybank [156] obtain depth information from 2D line drawings in perspective projection by enforcing constraints between faces. Three types of constraints are enforced: coplanarity, parallelism and perpendicularity. They show that this choice of constraints leads to a system of equations which can be solved analytically by matrix methods (they use singular value decomposition). Their method is not fully automatic—their system does not attempt to deduce from the drawing which faces in a drawing are intended to be parallel, so this information must be entered manually.

7.3.4 Compliance Function: Facial Planarity

Let each face f lie in the plane $P_f x + Q_f y + R_f z + C_f = 0$, and each vertex v have coordinates (x_v, y_v, z_v) . Since, by the general projection hypothesis, no face is parallel to the projection direction, $R_f \neq 0$, so it is possible to set $R_f = 1$ to normalise the equation: $P_f x_v + Q_f y_v + z_v + C_f = 0$. This equation forms the basis of Grimstead's linear system method [38] (see Section 7.4 below).

Alternatively, for any four vertices A, B, C and D on a face, an equation can be generated in z_A , z_B , z_C and z_D to make A coplanar with the other three. Providing BC and BD are non-collinear, BA can be expressed as a linear combination of the two, i.e. so (A - B) = m(C - B) + n(D - B), where m and n can be calculated from the known x and y coordinates of the vertices; rearranging this gives

$$z_A + (m+n-1)z_B - mz_C - nz_D = 0$$

which could be used in any linear system in which the unknowns include depth coordinates.

This does not extend uniquely to non-quadrilateral faces, for which more than one equation must be generated. As Lipson and Shpitalni [91] point out, enforcing coplanarity of any four points on a face does not enforce global face planarity.

An alternative approach is to enforce facial planarity after inflation by taking face equations as the input data and placing vertices at the intersections of the appropriate faces. Face equations could be obtained directly (using Grimstead's linear system method) or by finding the best fit to vertices lying on the face (following use of the z-coordinate linear system method).

Evidently, any use of facial planarity requires knowledge of which vertices lie on which faces. For natural line drawings of trihedral polyhedra, this is straightforward since all T-junctions in the drawing are occluding. For non-trihedral polyhedra, a labelling is required in order to distinguish occluding from non-occluding T-junctions. If the drawing is also permitted to contain hole loops, further prior processing is required in order to ensure that cofacial loops are identified (all loops belonging to a face must be coplanar).

This compliance function will not inflate a drawing into 3D by itself—clearly, $P_f = Q_f = C_f = z_v = 0$ for all (f, v) solves one linear system and $z_A = z_B = z_C = z_D$ for all (A, B, C, D) solves the other—so, if used at all, it must be combined with an inflationary compliance function.

7.3.5 Compliance Function: Parallel Lines

If it is believed that lines AB and CD should be parallel in 3D, it is straightforward to generate equations to encourage this. The lengths m of line AB and n of line CD can be calculated from the x and y coordinates, giving the equation

$$nz_A - nz_B - mz_C + mz_D = 0$$

which is linear in z-coordinates.

This function requires either knowledge of which pairs of lines in 2D correspond to edges which are parallel in 3D, or a weighting which should be applied to the equation reflecting confidence in the assumption of parallelism.

Again, this function will not inflate a drawing by itself. It is however useful as a secondary component of an inflation system, both for tidying the output and for ensuring that the system includes equations in occluded line coordinates (see, for example, Figure 7.11 on page 141).

7.3.6 Compliance Function: Vertical Lines

Lipson and Shpitalni [91] suggest that a line which is vertical in the drawing should correspond to an edge which is vertical in 3D space. This suggestion is rejected, with the cube in Figure B.10 (page 308) being given as a counter-example which emphasises the difference between "vertical in 3D space" (i.e. parallel with the y-axis) and "perpendicular to a base plane" (by the assumption of general viewpoint, the base plane is not the x-z plane).

7.3.7 Compliance Function: Isometry

Lipson and Shpitalni [91] observe that lines which are the same length in the drawing should correspond to edges which are the same length in 3D space. This is not useful in initial depth estimation, where it is the qualitative issue of which vertices are nearer than their neighbours, rather than the quantitative issue of by how much, which is important. The idea will be reconsidered in final geometric fitting (Chapter 11).

7.3.8 Compliance Function: Corner Orthogonality

A cubic corner [125] is a trihedral vertex of a solid object at which the three faces are aligned with the three coordinate axes. The criteria for determining whether a W-junction or Y-junction can be an accurate drawing of a cubic corner were established by Perkins [125]: for a W-junction, the two smaller angles must both be acute; for a Y-junction, all three angles must be obtuse. The proof of this result involved equations for the ratio of depth change along a line to the 2D length of any line VA at a cubic corner V which is a W-junction or Y-junction and which is linked by edges to vertices A, B and C:

$$\frac{|z_A - z_V|}{m} = \sqrt{\frac{-\cos\beta\cos\gamma}{\cos\alpha}},$$

where m is the 2D length of line VA, and α , β and γ are the 2D angles BVC, AVC and AVB.

Rearranging and simplifying provides an equation which could be used in a linear system:

$$|z_A - z_V| = m\sqrt{(\tan\beta\tan\gamma) - 1}.$$

In order to use this method, there must be a separate mechanism for determining whether A is in front of or behind V. The method fails for junctions which do not meet the Perkins criteria, such as may be found in oblique projections. Experimental results in Section 7.6 show that the quality of output can be poor if the object drawn is not a normalon.

This approach has also been used successfully for normalons by fixing a single vertex and propagating depth knowledge along edges [17].

7.3.9 Compliance Function: Skewed Facial Orthogonality

This is Kanade's method [63] for calculating the normal (P, Q, R) of a face given two axes on the face which are believed to be perpendicular in the 3D world, applied to those corners of faces which could be right-angles. The result is general for any two lines at angles α and β to the horizontal (for example, in Figure 7.3), α is the angle between line A and the horizontal and β is the angle between line B and the horizontal, chosen chosen such that the angle between α and β is obtuse) providing that the lines are in the plane of the face and perpendicular in 3D.

Kanade [63] notes that the vectors **a** and **b**, corresponding to the actual 3D directions of lines α and β respectively, must obey $\mathbf{a}.(\frac{P}{R},\frac{Q}{R},1) = \mathbf{b}.(\frac{P}{R},\frac{Q}{R},1) = 0$, and the belief that they are perpendicular translates to $\mathbf{a}.\mathbf{b} = 0$. From this, he obtains the result:

$$(\cos\alpha\cos\beta + \sin\alpha\sin\beta) + \left(\frac{P}{R}\cos\alpha + \frac{Q}{R}\sin\alpha\right)\left(\frac{P}{R}\cos\beta + \frac{Q}{R}\sin\beta\right) = 0$$

and it follows that

$$\frac{P}{R} = \rho \cos \frac{\alpha + \beta}{2}, \frac{Q}{R} = \rho \sin \frac{\alpha + \beta}{2}, \rho = \frac{\sqrt{-\cos(\alpha - \beta)}}{\cos \frac{\alpha - \beta}{2}}$$

(Grimstead [38] uses a similar expression).

The derivation assumes a correct orthographic projection, as would be the case for a photographic image of a real object. However, inaccurate or non-orthogonal projections, such as are often found in line drawings, can cause problems.

Skewed facial orthogonality, and deskewing methods in general, produce two possible face normals for each face. Choice between these must be based on other reasoning.

This method fits easily into an extended linear system incorporating P_j and Q_j , but does not fit naturally into the minimal linear system in z_i . Skewed facial orthogonality thus has no advantages over corner orthogonality, and is both more complex and less flexible, so is not recommended.

7.3.10 Compliance Function: Skewed Facial Symmetry

This method uses Kanade's original idea [63], which was to apply the equations given in the previous section to a face believed to show a skewed version of mirror symmetry, such as that in Figure 7.3, where lines B1 and B2 are perpendicular to A. In addition to the problems noted above, this would require that potential symmetry is identified before initial inflation, which contravenes one of the stated purposes of inflation (depth information will be used to assess candidate symmetries).

7.3.11 Compliance Function: Line Orthogonality

Lipson and Shpitalni [91] suggest that consecutive lines in the same face, other than those which are evidently collinear, should be made perpendicular in 3D space. The assumption itself is questionable, its implementation using the skewed facial orthogonality equations would suffer from all of the disadvantages noted in the previous two sections, and a more restricted version would reduce to the corner orthogonality equations described above.

As "minimum sum of dot products" at a vertex, this compliance function has been used with success to provide initial inflation in a λ -style optimisation, but that approach has already been rejected in Section 7.1.

7.3.12 Compliance Function: MSDA

It was noted by Marill [101] that the natural interpretations of convex polyhedra tended to be those with the minimum standard deviations of angles (MSDA) at corners on faces. This method is relatively successful for drawings which meet his assumptions, but the assumption of convexity is too limiting for the method to be of general use.

Since MSDA is a property of the object as a whole, not a local property, it cannot be incorporated in a linear system approach. It is not ideal even for the

optimisation approach which adjusts a single vertex at a time, since the MSDA for the entire object must be recalculated after each adjustment.

7.3.13 Compliance Function: Face Perpendicularity

Lipson and Shpitalni [91] suggest that all dihedral angles should initially be made 90°. This could be regarded as an improvement on cubic corners in that, as it uses edges rather than vertices, it is unaffected by the presence of non-trihedral vertices. This compliance function would evidently be a useful way of providing initial inflation in a λ -style optimisation, but that approach has already been rejected here.

7.3.14 Compliance Function: Prismatic Face

Under the title "prismatic face", Lipson and Shpitalni [91] include the various factors which contribute to a right extrusion, and in particular planar end caps and rectangular side faces. The geometric implications of these are not qualitatively different from face planarity and line parallelism. Weightings for these compliance functions could be increased for potential extrusions, but this contradicts the requirements by prejudging that the object drawn is indeed an extrusion rather than a frustum.

7.3.15 Compliance Function: Line Collinearity

Lipson and Shpitalni [91] suggest that lines which are collinear in the drawing should correspond to edges which are collinear in 3D space. This is intuitively plausible (and a stronger definition of general viewpoint would make it necessary), but it is unclear whether inflation should enforce this. For example, in Figure 7.14 on page 137, line collinearity is a consequence of the object's mirror symmetry, and the presence or otherwise of such symmetry is one of the things depth information will be used to assess; it could be argued that in such cases including line collinearity equations is premature. It is clear that line collinearity equations could potentially improve inflation of Figure B.38, but to no practical benefit: neither identification of mirror symmetry nor classification of the object as a normalon extrusion will be affected. A good case could be made for inclusion of line collinearity equations

where the lines are in different subgraphs, as in Figure B.553, as a means of relating depth coordinates of the two subgraphs; there has not been time to investigate the merits of this idea. Line collinearity will be reconsidered in Chapter 11.

7.3.16 Compliance Function: Planarity of Skewed Chains

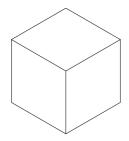
As will be seen in Chapter 8, adjacent faces sharing an edge with mirror symmetry can be chained. This imposes additional constraints on the geometry of the object. Since these constraints are non-linear in depth coordinates, require identification of symmetry before they can be generated, and depend on the topology of the object as a whole, they are more suited to final geometric fitting and are discussed in Chapter 11.

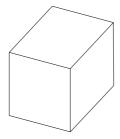
7.3.17 Compliance Function: Junction Label Pairs

Although geometrically sound, existing compliance functions take little account of human perception. The junction label pair (JLP) function, introduced here, is an attempt to remedy this. While it is impossible to calculate the depth of junctions in a single line drawing, humans can and do interpret line drawings and can, in general, reach a consensus about the depth implications—there is, for example, little ambiguity about what Figures 7.2–7.3 on page 117 represent. This consensus forms the basis of the new compliance function.

In Figure 7.4, a drawing of a cube in isometric projection, all lines are either Lba to Wbca or Wbca to Yccc. It is clear that the Yccc junction is nearer than the Wbca junctions, which in turn are nearer than the Lba junctions. In either case, the ratio of change of depth to 2D line length is $1/\sqrt{2}$. Depth information for the visible part of a cube, or any other axis-aligned drawing in exact isometric projection which uses only these three junction types, can be recovered precisely from this knowledge.

The logic given above can be extended for the JLPs in Figures B.1–B.9. Further JLPs can be found in drawings of objects which can be built from a small number of cubes, such as Figures B.43, B.42, B.87, B.66 and B.62, and of the simplest trihedral objects with hole loops, Figures B.422 and B.413. Handling of JLPs including extended trihedral junction labels is straightforward, from Figures B.156, B.158, B.159 and B.161.





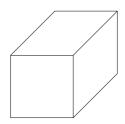


Figure 7.4: Standard Isometric

Figure 7.5: Slanted Isometric

Figure 7.6: Diagonal Isometric

It is not clear whether or not lines terminating in occluding T-junctions should be included in this analysis. It is certainly the case that in normalons all lines leaving Lba junctions approach the viewer, even those terminating at T-junctions; this could be used as an argument in favour of including the junction label pairs Lba-Tbaa and Lba-Tbab. Against this, it can be argued that since nothing is known about what lies at the occluded end of an edge when the line terminates at a T-junction, no use should be made of "knowledge" about this occluded vertex. The argument generalises to other frequently-occurring combinations involving T-junctions. This has been investigated, and the results are discussed in Section 7.6 below.

Not all drawings are in perfect isometric projection, or indeed in any mathematically-correct projection. It is not intuitively obvious whether JLP is more or less sensitive to different projections or drawing inaccuracies than (for example) corner orthogonality, particularly when used in combination with other compliance functions; a comparison is given in Section 7.6 below.

The JLP approach can be extended to non-normalons. Although some JLPs which cannot appear in drawings of normalons can and do appear in drawings of non-normalons, this can be handled in many cases simply by ignoring the unknown JLPs (one JLP per vertex is sufficient). A more robust solution is to generate a low-weight equation making the depth coordinates equal for the two vertices of an unrecognised JLP.

Extension to K-type tetrahedral vertices is straightforward, but it is at this point that the law of diminishing returns sets in—only those from Figures B.260, B.268, B.276 and B.282 have been included in RIBALD. Similar logic can also be used for the occluding M-L pairs in Figures B.117 and B.182. Since these values are of less

universal validity, RIBALD includes an arbitrary weighting factor W (range 0..1) in the equations in order to give priority to the more trustworthy values. The equations thus become

$$W \times (z_{i1} - z_{i2}) = W \times L_i \times K$$

 $(L_i \text{ is the length of edge } i; K \text{ is the depth/length ratio for the JLP}).$

Line Type	Nearer Junc.	Further Junc.	K	W	Figure
Convex	Yccc	Wbca	0.7071	1.0000	B.1
Convex	Yccc	Wcdc	0.7071	1.0000	B.2
Convex	Lac	Wbca	0.7071	1.0000	В.3
Convex	Lcb	Wbca	0.7071	1.0000	B.4
Convex	Yccc	Wdcd	0.7071	1.0000	B.6
Convex*	Wbca	Tbaa	0.7071	1.0000	B.43
Convex*	Wbca	Tbab	0.7071	1.0000	B.43
Convex	Lac	Wcdc	0.7071	1.0000	B.42
Convex	Lcb	Wcdc	0.7071	1.0000	B.42
Convex	Yccc	Tbdc	0.7071	0.75	B.158
Convex	Yccc	Tdac	0.7071	0.75	B.161
Convex	Yccc	Keced	0.7071	0.50	B.260
Convex	Kcccd	Wbca	0.7071	0.50	B.260
Convex	Yccc	Keced	0.7071	0.50	B.268
Convex	Keede	Wbca	0.7071	0.50	B.268
Convex	Yccc	Keced	0.7071	0.50	B.276
Convex	Yccc	Keced	0.7071	0.50	B.282
Convex	Mbcca	Wbca	0.7071	0.50	I
Convex	Yccc	Kabcd	0.7071	0.50	I
Convex	Yccc	Kabdc	0.7071	0.50	I
Convex	Yccc	Mccdc	0.7071	0.50	I
Convex	Yccc	Mcdcc	0.7071	0.50	I
Convex	Yccc	Mbcda	0.7071	0.50	I
Convex	Yccc	Mbdca	0.7071	0.50	I
Convex	Xcccc	Wbca	0.4030	0.28	V
Convex	Xcccc	Mbcca	0.3928	0.28	V
Convex	Yccc	Mbcca	0.1414	0.10	V
Convex	Wede	Wbca	0.0345	0.02	V
Convex	(any)	(same)	0.0	0.4000	
Convex	(any)	(other)	0.0	0.0100	

Table 7.1: Constants and Weights for Depth Estimation

The complete set of junction label pairs as implemented in RIBALD and used to obtain the results shown below are tabulated in Tables 7.1–7.3. As well as entries for

Line Type	Nearer Junc.	Further Junc.	K	W	Figure
Concave	Wede	Yabd	0.7071	1.0000	B.2
Concave	Wdcd	Yabd	0.7071	1.0000	B.6
Concave	Wede	Lbd	0.7071	1.0000	B.7
Concave	Wede	Lda	0.7071	1.0000	B.8
Concave	Wede	Yddd	0.7071	1.0000	B.9
Concave	Wdcd	Yddd	0.7071	1.0000	B.87
Concave	Wdcd	Lbd	0.7071	1.0000	B.422
Concave	Wdcd	Lda	0.7071	1.0000	B.422
Concave	Wede	Tbda	0.7071	0.75	B.156
Concave	Tbdc	Yabd	0.7071	0.75	B.158
Concave	Wede	Tdab	0.7071	0.75	B.159
Concave	Tdac	Yabd	0.7071	0.75	B.161
Concave	Keded	Yabd	0.7071	0.50	B.276
Concave	Kdcdc	Yabd	0.7071	0.50	B.282
Concave	Kcccd	Yabd	0.7071	0.50	I
Concave	Keede	Yabd	0.7071	0.50	I
Concave	Wede	Kabcd	0.7071	0.50	I
Concave	Wede	Kabcd	0.7071	0.50	I
Concave	Mbcda	Mdcdd	0.7071	0.50	I
Concave	Mbcda	Mddcd	0.7071	0.50	I
Concave	Mbdca	Mdcdd	0.7071	0.50	I
Concave	Mbdca	Mddcd	0.7071	0.50	I
Concave*	Tbad	Wede	0.0304	0.02	V
Concave	(any)	(same)	0.0	0.4000	
Concave	(any)	(other)	0.0	0.0100	

Table 7.2: Constants and Weights for Depth Estimation

JLPs deduced from geometric reasoning, some were derived experimentally, either from pairs which invariably produced the same depth ordering (marked I in the tables), or from pairs whose implications were unclear¹ but which occurred sufficiently often to make doing something explicit worthwhile (marked V in the tables). For these latter, the depth ratio is $K = C/\sqrt{2}$ and the confidence W = C/2, where $C = \frac{c_A - c_B}{c_A + c_B + c_E}$ and c_A is the frequency with which vertex A is clearly closer than

 $^{^1}$ Two examples illustrate the uncertainty. It seems in Figure B.310 that the Yccc junction is in front of the Mbcca junction, but that a change of viewpoint could alter this. This impression of uncertainty is reinforced by examining Figures B.495 and B.497, where the same pair of junction labels occur but with implications which disagree with one another. Similarly, in Figure B.58 it appears that the Lba junction is in front of the Yabd junction, and the Wbca junction in front of the Wcdc junction (although all of these junction types appear in trihedral objects, these pairs of junction types do not occur in normalons, making any deduction from first principles difficult). That there is uncertainty about these can be seen by comparing Figures B.470 and B.476.

Line Type	Nearer Junc.	Further Junc.	K	W	Figure
Occluding	Wbca	Lba	0.7071	1.0000	B.1
Occluding	Wbca	Yabd	0.7071	1.0000	B.2
Occluding	Wbca	Lac	0.7071	1.0000	В.3
Occluding	Wbca	Lcb	0.7071	1.0000	B.4
Occluding*	Tbab	Lba	0.7071	1.0000	В.3
Occluding*	Tbaa	Lba	0.7071	1.0000	B.4
Occluding	Lab	Lba	0.7071	1.0000	B.5
Occluding	Wbca	Lbd	0.7071	1.0000	B.7
Occluding	Wbca	Lda	0.7071	1.0000	B.8
Occluding	Lab	Lac	0.7071	1.0000	B.43
Occluding	Lab	Lcb	0.7071	1.0000	B.43
Occluding	Lab	Yabd	0.7071	1.0000	B.66
Occluding*	Tbab	Yabd	0.7071	1.0000	B.413
Occluding*	Tbaa	Yabd	0.7071	1.0000	B.413
Occluding	Wbca	Tbda	0.7071	0.75	B.156
Occluding	Tbda	Lba	0.7071	0.75	B.156
Occluding	Wbca	Tbdc	0.7071	0.75	B.158
Occluding	Wbca	Tdab	0.7071	0.75	B.159
Occluding	Tdab	Lba	0.7071	0.75	B.159
Occluding	Wbca	Tdac	0.7071	0.75	B.161
Occluding	Mbcca	Lba	1.4142	0.50	B.117
Occluding	Mbcda	Lba	0.7071	0.50	B.182
Occluding	Mbdca	Lba	0.7071	0.50	B.182
Occluding	Kabcd	Lba	0.7071	0.50	I
Occluding	Kabdc	Lba	0.7071	0.50	I
Occluding	Wbca	Kabcd	0.7071	0.50	I
Occluding	Wbca	Kabdc	0.7071	0.50	I
Occluding	Mbcca	Wbca	0.1872	0.13	V
Occluding	Lba	Yabd	0.0848	0.06	V
Occluding	(any)	(same)	0.0	0.4000	
Occluding	(any)	(other)	0.0	0.0100	

Table 7.3: Constants and Weights for Depth Estimation

vertex B, c_B is the frequency with which vertex B is clearly closer than vertex A, and c_E is the frequency of them being about the same. (Frequencies were measured from the subset of test drawings which can be labelled correctly.)

This compliance function fits easily into either linear system approach and could also be useful either as a part of an optimisation approach or as a preprocessing stage to avoid the "local minimum trap". Entries in the JLP tables can be used directly to predict (with very few exceptions) the depth ordering of neighbouring vertices, so this combination could therefore be used as a preprocessing stage for the corner orthogonality method described above.

JLP requires junction and line labels.

7.4 Grimstead's Linear System Approach

Using equations of the form $P_f x_v + Q_f y_v + z_v + C_f = 0$, Grimstead generated the frontal geometry of the object using a system of linear equations with the unknowns P_f , Q_f , C_f and z_v .

Face planarity equations were generated for each vertex-face pair. Line parallelism equations were generated for each pair of 2D parallel lines identified. Equations for P and Q were generated for each skewed symmetry artefact detected.

The system was then solved to give equations for each face using a weighted least-squares algorithm. The solution process was iterative, with the equation with the largest residual error being dropped at each subsequent iteration. Weightings of equations with large residuals were reduced on the next iteration; the original weightings of each equation were arbitrary. Eventually, a self-consistent set of equations remained. The performance was assessed using right-angle fit and minimum standard deviation of angles in order to determine whether the process was converging towards a 3D solid or towards the redundant (flat) solution. In the former case, the output of the linear system gave P, Q and C values for each face, and vertex coordinates were obtained by intersecting faces.

The approach as a whole suffered from several disadvantages, of which a number are relevant here:

• It is an iterative process, $O(e^4)$ or worse. This is not insurmountable—the

output of the first iteration could be used for preliminary depth coordinates;

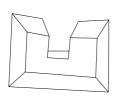
- It changes vertex x- and y-coordinates, an idea which has already been rejected. This too is not insurmountable—the linear system gives vertex z-coordinates as part of its output, and these could be used directly instead of recalculating vertex coordinates from face equations;
- Grimstead's recommended weightings are arbitrary, being the values which were empirically found to work best for the test drawings used.
- The variables being optimised are of different kinds, thus creating doubt as to whether the fit really is "best".
- The skewed symmetry equations for P and Q were decoupled—a single geometric constraint is represented by multiple constraint equations (iterative least-squares fitting might drop one and retain the other when rejecting inconsistent equations, and iterative weighting adjustment might make one more important than the other).

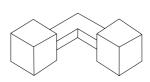
Also, a hidden weighting is also given to skewed symmetry estimates of 'front-on', rather than 'side-on', faces, through the representation of the face normal as [P, Q, 1] rather than [P, Q, R]. This is justifiable—it is easier for the user to draw "front-on" faces, so they will be more accurate.

7.5 Depth from Labelling

RIBALD uses a linear system where the only unknowns are the vertex z-coordinates. The number of unknowns is given by the number of junctions in the drawing. In the simplest form of this approach, the number of equations is one more than the number of lines in the drawing, these equations being generated using the JLP compliance function. The additional equation fixes an arbitrary vertex at an arbitrary depth in order to locate the object in space. Where a JLP is not explicitly tabulated, the depth difference is assumed to be zero and the weighting very low—this guarantees that each unknown will be represented in the system. A black-box linear system solver is invoked to find the least-squares fit to the resulting overconstrained system of equations.

For a simplistic method, this approach works surprisingly well in practice for most junction types. It is even possible to obtain a correct depth ordering for Sugihara's Box, Figure B.146. The method fails entirely for drawings which are not graph-complete, such as Figures 7.7 and 7.8—it is not possible to find a path between every pair of edges without making use of occluding T-junctions, and the resulting linear system cannot be solved.





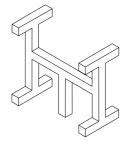


Figure 7.7: No Visible Vertices on Back Line

Figure 7.8: Cannot Reach Back From Front

Figure 7.9: Too Many Parallel Line Pairs

Various refinements have been investigated. Equations from line parallelism are easily incorporated into the linear system (the equations are linear in z-coordinates) and in many cases improves the frontal geometry significantly, so are recommended. It is found in practice that the computational overhead is acceptable for most drawings of engineering objects but becomes unacceptable if all possible parallel line pairs of drawings such as Figure 7.9 are included in the linear system. Recall that the number of equations should be O(e) for large e: a threshold is applied; if the number of lines n in a parallel bundle is below the threshold, all n(n-1) possible line pairs generate equations; if n is above the threshold, (n-1) equations are generated to make the longest line in the bundle parallel to all of the others. Setting the threshold to 20 is a satisfactory compromise.

RIBALD sets the weightings for parallel line pair equations proportional to the product of the lengths of the two lines and to the figure of merit for parallelism. The relative weightings of parallel line pairs and JLPs are such that if the two longest lines are exactly parallel then the weighting for the equation making them parallel is the same as the weighting for the equation giving the direction of an *Lba–Wbca* line.

If line parallelism is not included, some other mechanism must be used to ensure that the edges resulting from interrupted lines at K-type vertices and extended

trihedral T-junctions are collinear in 3D.

Face planarity is a more difficult issue. Enforcing face planarity as a postprocessing stage can be rejected—it sometimes improves well-drawn sketches, but can also make poorly-drawn sketches significantly worse, and conflicts with the preference for leaving the x and y coordinates for each vertex unchanged from the original drawing for as long as possible.

For drawings including hole loops, such as Figure 7.10 (page 137), some mechanism is needed to make cofacial loops coplanar. This is best achieved by incorporating the four-vertex version of the face planarity function into the linear system.

Use of four-vertex coplanarity for the general case of four or more vertices lying on a face has been examined, with results shown in Section 7.6 below.

Various solutions are possible to the problems posed by Figures 7.7 and 7.8, but none are ideal. An equation placing the point at which the line vanishes from view a fixed arbitrary depth behind the corresponding point on the line which occludes it is linear in z-coordinates, and could be incorporated. These are sometimes needed for drawings where the subgraph count is greater than 1, but if included unnecessarily the quality of the output deteriorates. As implemented in RIBALD, such equations are, by default, not included, but if the linear system cannot be solved and the subgraph count is greater than 1, such equations are added to the system, and the linear system solver invoked a second time.

It can be seen in Figures 7.10, 7.11 and 7.14 that lines terminating at occluding T-junctions are often parallel to other lines in the drawing. If this is so, use of entries in the JLP tables referring to occluding T-junctions is not required.

Since corner orthogonality can be used in linear systems, and is preferable for isometric projections of normalons, RIBALD includes an auto-selection process which uses corner orthogonality equations in place of the default JLP for objects with exactly three bundles of parallel lines and where all W-junctions and Y-junctions meet the Perkins criteria. Alternatively, either compliance function can be invoked specifically. Section 7.6 analyses the output produced.

Although there is a theoretical justification for choosing the value $K=1/\sqrt{2}$ in the JLP table, it is not clear that this is the value which produces best results. It is even possible that, in place of a fixed table of JLPs, the table could be produced by analysis of finished geometrical objects (for example, using an iterative process

where the finished geometric object output by one iteration is used to generate the JLP table for the next). This idea has not been investigated, and there are potential objections. The suggested analysis process might or might not converge, and might converge to a local minimum (although this is not necessarily a problem as the local minimum might be one which works well for an individual user—this could even be one way of tuning the system to suit an individual user).

7.6 Results and Conclusions

The derivation of the JLP approach assumed a drawing in isometric projection. Its performance in non-isometric projections is compared here with corner orthogonality, which does not make this assumption. Furthermore, the derivation of both compliance functions assumed a drawing in a correct orthogonal projection. The performance of both in incorrect projections is also investigated.

Choice between JLP and corner orthogonality, and choice concerning which secondary compliance functions should be used in combination with them, is investigated with reference to six test drawings selected partly to be a representative sample of test drawings and partly to investigate specific points.

7.6.1 JLP vs Corner Orthogonality

A preliminary investigation [172] was sufficient to reject skewed facial symmetry. With parallelism of lines incorporated, RIBALD produced dihedral angles for the three cubes shown in Figures 7.4–7.6 (page 125) of 90.0°, 90.0°, 90.0° for Figure 7.4, 87.3°, 80.8°, 80.4° for Figure 7.5, and 84.3°, 84.3°, 71.3° for Figure 7.6. By way of comparison, skewed symmetry would produce angles of 90° for Figures 7.4 and 7.5, both of which are correct projections, but fails entirely for Figure 7.6². Distorting the projection by moving the right-hand lines 10° closer to the horizontal gives inter-facial angles of 89.4°, 84.5°, 83.9° for Figure 7.4 (skewed symmetry would give 99.1°, 89.4°, 81.7°) and 87.9°, 82.0°, 74.5° for Figure 7.5 (skewed symmetry would give 101.5°, 71.4°, 69.5°). Although better in some circumstances, skewed facial symmetry is inappropriate for initial inflation because there are valid drawings for which

²although $\cos 90^{\circ}$ is 0, $\cos \frac{\pi}{2}$ will never be exactly 0 as π cannot be represented exactly as a real number; it will be positive as often as not, and when it is, $\sqrt{-\cos \pi}$ will fail

α	β	JLP worst	JLP mean	JLP best	CO worst	CO mean	CO best
30	30	89.999	90.000	90.000	89.999	90.000	90.000
29	32	88.851	89.223	89.409	89.245	89.558	89.882
28	34	87.713	88.429	88.787	88.503	89.096	89.698
27	36	86.587	87.620	88.136	87.772	88.614	89.453
26	38	85.473	86.794	87.455	87.052	88.115	89.153
25	40	84.369	85.951	86.743	86.340	87.600	88.801
24	42	83.276	85.092	86.000	85.636	87.068	88.401
23	44	82.195	84.216	85.227	84.938	86.523	87.954
22	46	81.124	83.323	84.423	84.246	85.963	87.463
21	48	80.063	82.413	83.588	83.558	85.389	86.930
20	50	79.014	81.486	82.722	82.873	84.803	86.357
19	52	77.974	80.541	81.825	82.191	84.203	85.746
18	54	76.945	79.580	80.898	81.510	83.592	85.097
17	56	75.925	78.602	79.941	80.831	82.969	84.574
16	58	74.916	77.608	78.954	80.153	82.334	84.131
15	60	73.917	76.598	77.938	79.474	81.688	83.684
14	62	72.927	75.572	76.894	78.795	81.032	83.235
13	64	71.946	74.530	75.823	78.115	80.365	82.783
12	66	70.975	73.475	74.725	77.433	79.689	82.328
11	68	70.013	72.405	73.601	76.749	79.003	81.870
10	70	69.060	71.322	72.453	76.063	78.308	81.409
9	72	68.116	70.227	71.283	75.375	77.605	80.945
8	74	67.181	69.121	70.091	74.683	76.895	80.477
7	76	66.255	68.004	68.879	73.988	76.176	80.006
6	78	65.337	66.879	67.650	73.289	75.452	79.531
5	80	64.427	65.745	66.404	72.587	74.721	79.053

Table 7.4: Sensitivity of Inflation to Projection

it fails entirely, and because it is roughly twice as sensitive as JLP to typical freehand drawing errors.

Comparison of depth ratios derived from JLP with those derived from corner orthogonality is less consistent. The tests in Tables 7.4–7.6 were run on variants of Figure 7.10 in which all lines were drawn parallel with one of three axes, one of the axes being vertical, and the other two axes were at angles of α and β with the horizontal. Distortions in Table 7.4 approximate correct non-isometric projections; distortions in Tables 7.5 and 7.6 represent two typical ways of deviating from isometric projection. The RIBALD options used were to generate depth equations from all vertex pairs, including the two Y-T pairs, and from parallel lines bundled

α	β	JLP worst	JLP mean	JLP best	CO worst	CO mean	CO best
30	30	89.999	90.000	90.000	89.999	90.000	90.000
29	31	89.420	89.613	89.995	89.618	89.801	89.999
28	32	88.834	89.223	89.978	89.236	89.601	89.995
27	33	88.242	88.828	89.948	88.839	89.396	89.973
26	34	87.644	88.429	89.908	88.432	89.187	89.936
25	35	87.040	88.026	89.855	88.014	88.974	89.886
24	36	86.430	87.620	89.792	87.585	88.758	89.822
23	37	85.813	87.209	89.716	87.144	88.539	89.744
22	38	85.191	86.794	89.629	86.691	88.317	89.650
21	39	84.562	86.375	89.531	86.225	88.093	89.541
20	40	83.927	85.951	89.420	85.746	87.866	89.422
19	41	83.286	85.524	89.299	85.255	87.638	89.288
18	42	82.638	85.092	89.166	84.749	87.408	89.138
17	43	81.984	84.656	89.022	84.229	87.178	88.971
16	44	81.324	84.216	88.866	83.695	86.948	88.785
15	45	80.658	83.772	88.699	83.146	86.719	88.886
14	46	79.985	83.323	88.521	82.582	86.493	89.088
13	47	79.305	82.870	88.332	82.004	86.272	89.359
12	48	78.619	82.413	88.132	81.412	86.056	89.711
11	49	77.927	81.951	87.921	80.807	85.798	89.839
10	50	77.228	81.486	87.698	80.192	85.417	89.270
9	51	76.523	81.016	87.465	79.569	85.006	88.553
8	52	75.812	80.541	87.222	78.945	84.560	87.647
7	53	75.094	80.063	86.967	78.329	84.072	86.496
6	54	74.370	79.580	86.702	77.736	83.532	85.173
5	55	73.640	79.093	86.427	77.193	82.926	85.214

Table 7.5: Sensitivity of Inflation to Projection

together. The JLP columns show the worst, mean and best perpendicular dihedral angles obtained using a fixed depth ratio of $1/\sqrt{2}$, and the CO columns show the equivalent results obtained using variable depth ratios calculated using corner orthogonality. In Table 7.4, corner orthogonality is clearly preferable to JLP; in the other two tables, it is somewhat preferable on average.

7.6.2 Illustrative Results

Further analysis is needed to establish whether JLP or corner orthogonality is to be preferred more generally, and which (if any) secondary compliance functions should

α	β	JLP worst	JLP mean	JLP best	CO worst	CO mean	CO best
30	30	89.999	90.000	90.000	89.999	90.000	90.000
30	31	89.420	89.613	89.995	89.533	89.722	89.996
30	32	88.834	89.223	89.977	89.060	89.447	89.999
30	33	88.242	88.828	89.948	88.581	89.168	89.984
30	34	87.644	88.429	89.907	88.097	88.889	89.957
30	35	87.040	88.026	89.855	87.610	88.608	89.918
30	36	86.430	87.620	89.791	87.120	88.327	89.867
30	37	85.813	87.209	89.716	86.628	88.045	89.801
30	38	85.191	86.794	89.629	86.136	87.762	89.720
30	39	84.562	86.375	89.530	85.644	87.479	89.769
30	40	83.927	85.951	89.420	85.154	87.196	89.858
30	41	83.286	85.524	89.299	84.667	86.912	89.971
30	42	82.638	85.092	89.166	84.185	86.592	89.891
30	43	81.985	84.656	89.022	83.709	86.254	89.726
30	44	81.324	84.216	88.866	83.242	85.908	89.533
30	45	80.658	83.772	88.699	82.786	85.554	89.311
30	46	79.985	83.323	88.521	82.344	85.191	89.057
30	47	79.305	82.870	88.332	81.919	84.819	88.768
30	48	78.619	82.413	88.132	81.516	84.439	88.440
30	49	77.927	81.951	87.920	81.058	84.051	88.069
30	50	77.228	81.486	87.698	80.424	83.654	87.648
30	51	76.523	81.016	87.465	79.756	83.249	87.170
30	52	75.812	80.541	87.221	79.049	82.836	86.621
30	53	75.094	80.063	86.967	78.293	82.415	85.988
30	54	74.370	79.580	86.702	77.476	81.990	85.402
30	55	73.640	79.093	86.426	76.576	81.563	86.025

Table 7.6: Sensitivity of Inflation to Projection

be used in combination with them. This is illustrated here with reference to six test drawings (see Figures 7.10–7.15).

The results are tabulated below. In the tables, column \bot indicates the method used for creating perpendicularity, column ll indicates whether or not line parallelism was used, column 4vp indicates whether or not four-vertex planarity was used, and column $T\bot$ indicates whether or not lines terminating at T-junctions produced perpendicularity equations. The remaining columns vary depending on what is examined.

Table 7.7 shows the worst, mean and best perpendicular dihedral angles after inflation of Figure 7.10, and the mean and worst deviations from planarity. Dihedral

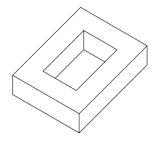


Figure 7.10: O Block

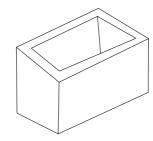


Figure 7.11: O Block

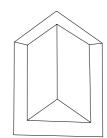


Figure 7.12: Bracket

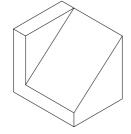


Figure 7.13: Wedge

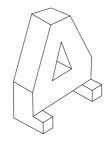


Figure 7.14: A Block

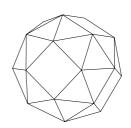


Figure 7.15: Snub Cube

angles for each edge were estimated by calculating a best-fit face equation for the faces meeting the edge. Deviations from planarity were estimated by calculating a best-fit face equation for each face and the distance of the vertex coordinates from this plane (the absolute values, in pixels, are arbitrary but the relative values significant). Several points can be noted. Firstly, because of the presence of a hole loop, some face planarity equations were present in the system regardless of the options selected in order to make the inner and outer loops of the O-face coplanar. Secondly, none of the methods give dihedral angles of 90°—although the drawing looks reasonable, it is not a perfect projection. Thirdly, although best dihedral angle results are obtained using just corner orthogonality, the differences are slight—all variants of the method give reasonable results. Fourthly, line parallelism equations are as effective as face planarity equations in enforcing face planarity—one or other should be included, but using both is superfluous.

Table 7.8 shows the dihedral angles after inflation of Figure 7.11, and the deviations from planarity, using various options. Although the quality of output is significantly worse than that obtained from Figure 7.10, differences between variants are slight. The variants fail about equally at the impossible task of producing coplanar faces without adjusting any x- or y-coordinates. There is no justification

	met	hods		dih	edral ang	gles	deviations from planarity		
\perp	ll	4vp	ТТ	worst	mean	best	mean	worst	
JLP	N	N	N	80.657	84.575	88.999	5.621	36.3523	
CO	N	N	N	83.390	86.029	89.619	9.968	27.4684	
JLP	Y	N	N	80.657	83.772	88.699	0.000	0.0021	
CO	Y	N	N	82.728	85.543	89.225	2.832	5.6158	
JLP	N	Y	N	80.657	84.575	88.999	5.621	36.3523	
CO	N	Y	N	83.036	85.887	88.881	6.602	27.2313	
JLP	Y	Y	N	80.657	83.772	88.699	0.000	0.0021	
CO	Y	Y	N	82.924	85.433	88.606	1.660	5.3446	
JLP	N	N	Y	80.657	83.772	88.699	0.000	0.0000	
CO	N	N	Y	81.867	85.511	89.619	5.691	13.9337	
JLP	Y	N	Y	80.657	83.772	88.699	0.000	0.0000	
CO	Y	N	Y	82.591	85.501	89.186	2.745	5.6111	
JLP	Ν	Y	Y	80.657	83.772	88.699	0.000	0.0000	
CO	N	Y	Y	82.427	85.371	88.881	2.473	7.7982	
JLP	Y	Y	Y	80.657	83.772	88.699	0.000	0.0000	
CO	Y	Y	Y	82.788	85.392	88.567	1.631	5.3402	

Table 7.7: Results for the well-drawn O-Block

here for going beyond the basic linear system.

Figure 7.12 appears in several textbooks, notably Shirai's [148], where it illustrates the difference between freehand line drawings and mathematically-correct drawings. The figure is not mathematically-correct, since the two edges which divide the top face from the front face must be collinear, and the corresponding lines are not. Ideally, after inflation, all dihedral angles should be either 90° or 45° (although it could be argued that making the convex hull of the top face an equilateral triangle would also be a reasonable interpretation). The final six columns of Table 7.9 show the worst and best right-angles, the worst and best 45° angles, and the mean and worst deviations from planarity. It is clear from the columns "worst 90" and "best 90" that JLP is preferable to corner orthogonality, both in terms of dihedral angles and face planarity, confirming the initial impression that corner orthogonality does not produce good results for non-normalons. Otherwise, there is little to choose between the variants. Analysis of dihedral angles suggests that it is preferable to use the simplest version of the method, since this is the one in which the two dihedral angles which should be 45° are most nearly equal. Face planarity equations

	met	hods		dih	edral ang	gles	deviation	ons from planarity
	ll	4vp	T⊥	worst	mean	best	mean	worst
JLP	N	N	N	56.228	77.487	89.957	11.481	69.5183
CO	N	N	N	56.887	77.739	89.615	11.445	69.4618
JLP	Y	N	N	55.704	76.606	89.159	10.542	70.2891
CO	Y	N	N	56.190	76.891	89.403	10.509	70.5823
JLP	N	Y	N	55.438	76.619	88.661	10.069	69.1658
CO	N	Y	N	55.940	76.929	88.901	9.719	69.5663
JLP	Y	Y	N	55.267	76.114	88.520	9.775	69.6663
CO	Y	Y	N	55.695	76.399	88.753	9.819	70.1685
JLP	N	N	Y	54.799	76.782	89.957	11.366	68.3666
CO	N	N	Y	55.453	77.033	89.615	11.329	68.3044
JLP	Y	N	Y	54.263	75.884	89.159	10.426	69.1273
CO	Y	N	Y	54.736	76.169	89.403	10.392	69.4072
JLP	N	Y	Y	53.995	75.904	88.661	9.952	67.9961
CO	N	Y	Y	54.487	76.214	88.901	9.601	68.3863
JLP	Y	Y	Y	53.821	75.387	88.520	9.658	68.4952
CO	Y	Y	Y	54.235	75.672	88.753	9.701	68.9837

Table 7.8: Dihedral Angles for the badly-drawn O-Block

have some effect in achieving their objective.

The wedge in Figure 7.13 was included in order to examine results for objects with non-axis-aligned faces and no helpful mirror symmetry. If the idea of line isometry is accepted, the "sloping roof" should make an angle with the L-shaped front face of $\frac{4}{3}$, 53.13°. The rightmost columns in Table 7.10 show the worst and best perpendicular dihedral angles and the angle between the sloping roof and the L-shaped front face, and the mean and worst deviations from planarity. Surprisingly, corner orthogonality is consistently preferable to JLP in achieving the correct dihedral angle, although the difference is not always large. Four-vertex planarity is almost essential for a good interpretation. The vertex which is most seriously misplaced if face planarity equations are not included is the one where the concave line meets the L-shaped front face. This gives few clues concerning which feature present in Figure 7.13 makes use of face planarity equations necessary.

In Figure 7.14, the dihedral angle between the square top and the sloping side should be 60° if line isometry is accepted. The dihedral angle between the sloping side and the vertical side should therefore be 30°. If line parallelism is accepted, the

me	etho	ds		dihedra	l angles		deviations from planarity		
	ll	4vp	worst 90	best 90	worst 45	best 45	mean	worst	
JLP	N	N	84.032	87.945	49.316	46.025	17.440	40.6833	
CO	Ν	N	73.132	88.562	58.292	49.494	23.341	156.3602	
JLP	Y	N	84.643	88.204	52.905	42.458	12.802	43.5912	
CO	Y	N	74.347	89.809	55.720	50.002	12.633	55.8313	
JLP	N	Y	84.166	89.993	52.134	42.793	6.105	34.0688	
CO	Ν	Y	77.057	89.297	59.107	43.848	9.004	39.2854	
JLP	Y	Y	84.338	88.819	53.263	41.456	6.642	35.9234	
CO	Y	Y	77.312	89.046	59.443	43.247	8.575	39.9671	

Table 7.9: Dihedral Angles for the Angle Bracket

me	etho	ds	dihe	edral angle	es	deviations from planarity		
	ll	4vp	worst 90	best 90	roof	mean	worst	
JLP	N	N	67.623	87.121	76.570	22.097	53.5437	
CO	N	N	73.418	85.196	72.811	22.715	71.8669	
JLP	Y	N	77.060	87.256	71.505	15.635	43.9723	
CO	Y	N	78.283	86.798	66.865	14.253	51.5327	
JLP	N	Y	70.907	87.903	60.294	6.828	17.5546	
CO	N	Y	79.500	88.506	55.368	3.129	10.9773	
JLP	Y	Y	78.094	88.539	59.345	3.695	5.7795	
CO	Y	Y	82.980	87.886	54.814	2.492	10.5396	

Table 7.10: Dihedral Angles for the Wedge

internal dihedral angle in the triangular through hole should be 60° . The rightmost columns of Table 7.11 list, respectively, these three angles, the worst and best perpendicular dihedral angles, and the mean and worst deviations from planarity. Since this drawing was included because it illustrates a common type of problem junction (an occluding T-junction completing a triangular partial face) for which only line parallelism generates a reasonable equation, it is to be expected that line parallelism is almost essential for good dihedral angles, and this is so. Four-vertex planarity does more harm than good to dihedral angle values, but its incorporation can still be justified for its significant effect in making faces planar. It can be noted that here the improvement of JLP over corner orthogonality on deviations from planarity is almost as large as that achieved using the face planarity function.

It can be noted that in neither this drawing nor any of the preceding ones does

	met	hods			di	hedral ar	ngles		dev. fr	om plan.
上	ll	4vp	$T\bot$	$T(60^{\circ})$	$F(30^{\circ})$	I(60°)	Worst ⊥	Best \perp	mean	worst
JLP	N	N	N	59.248	36.660	86.411	60.674	89.505	7.234	47.373
CO	N	N	N	61.523	37.440	86.411	59.849	89.795	9.135	81.887
JLP	Y	N	N	60.310	35.722	77.466	84.125	89.644	6.066	44.208
CO	Y	N	N	62.089	37.408	78.139	78.761	89.432	7.475	82.506
JLP	N	Y	N	65.221	27.754	86.019	56.057	89.402	3.613	18.771
CO	N	Y	N	65.307	28.083	82.644	51.349	89.534	5.390	51.428
JLP	Y	Y	N	65.723	27.197	78.024	79.780	89.854	2.904	15.759
CO	Y	Y	N	65.844	27.574	80.403	70.211	89.965	5.485	52.259
JLP	N	N	Y	59.248	36.660	84.815	59.184	89.505	6.327	47.373
CO	N	N	Y	61.523	37.440	84.815	58.326	89.795	9.135	81.887
JLP	Y	N	Y	60.310	35.722	77.470	84.125	89.644	6.066	44.208
CO	Y	N	Y	62.089	37.409	78.144	78.760	89.433	7.475	82.506
JLP	N	Y	Y	65.221	27.754	84.431	54.532	89.402	2.709	13.886
CO	N	Y	Y	65.307	28.083	80.961	49.822	89.534	5.390	51.428
JLP	Y	Y	Y	65.723	27.197	78.027	79.781	89.853	2.902	15.759
СО	Y	Y	Y	65.844	27.574	80.409	70.210	89.966	5.485	52.259

Table 7.11: Dihedral Angles for the A-Block

presence or absence of occluding T-junction entries in the JLP tables make any significant difference.

Figure 7.15 is a snub cube, an Archimedean regular solid in which all dihedral angles should be either 26.8° or 37.0° The rightmost columns in Table 7.12 show the smallest, mean and largest dihedral angles and the mean and worst deviations from planarity. Corner orthogonality cannot be used here as there are no trihedral vertices.

None of the variants tested produce good frontal geometry for this drawing, and

methods			dil	nedral an	gles	deviations from planarity		
上	ll	4vp	small	mean	large	mean	worst	
JLP	N	N	0.028	17.737	71.918	2.342	69.5815	
JLP	Y	N	2.257	17.636	67.669	1.501	28.5971	
JLP	Ν	Y	1.203	21.189	54.495	0.359	4.9473	
JLP	Y	Y	3.978	19.343	62.795	0.424	5.7120	

Table 7.12: Dihedral Angles for the Snub Cube

did not necessarily achieve correct depth ordering of adjacent vertices. In view of the global regularity of the object, a global approach such as Marill's MSDA would be more appropriate here than any approach based on accumulation of local data.

7.6.3 Overall Results

Table 7.13 lists the number of drawings for which all neighbouring vertices were ordered correctly in depth by the variants listed above. In producing this table, all 481 drawings which can be labelled correctly using any of the labelling algorithms described in Chapter 4 were used as test data. Since different methods may be preferred for normalons and non-normalons, results are also presented separately for those drawings where bundling identified three bundles and those where it identified four or more bundles.

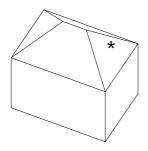
me	etho	ds	overall		3-bu	ndle	4+b	undle
	ll	4vp		×		×		×
JLP	Y	Y	295	186	128	7	167	179
JLP	Y	N	307	174	131	4	176	170
CO	Y	Y	257	224	103	32	154	192
CO	Y	N	256	225	100	35	156	190
JLP	Ν	Y	144	337	55	80	89	257
JLP	N	N	138	343	62	73	76	270
CO	N	Y	155	326	51	84	104	242
CO	N	N	144	337	54	81	90	256

Table 7.13: Correct and Incorrect Neighbour Ordering

The results confirm that use of parallel line information in inflation is almost essential.

The apparent advantage of JLP over corner orthogonality comes from the inability of corner orthogonality to handle correctly drawings in the projection used for the drawings in Appendix B.5.3.

It appears from the tabulated results that it is slightly preferable to omit fourvertex planarity equations. This could be an artefact of the assessment criterion chosen, that of correct depth ordering of adjacent vertices. Since JLP is specifically designed to produce correct depth ordering of adjacent vertices, it is plausible that a combination with other compliance functions will make it less effective at meeting this specific criterion even if it improves results in a more general but less easily-quantified way. In drawings with more than one subgraph, such as Figure B.553, four-vertex planarity equations are the best way of ensuring that the depth coordinates of the two subgraphs are related.



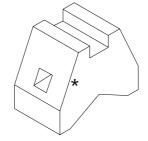


Figure 7.16: House

Figure 7.17: Setting Piece [128]

It can be noted that in many of the cases for which the best variant (JLP, using parallel lines but not 4-vertex planarity) produced "incorrect" results, depth ordering produced by the algorithm is tolerable but is not the one I expected. Figures 7.16 and 7.17 show the misdirected edges in two such cases.

The Archimedean solids defeat RIBALD's inflation approach—even the best variant produced incorrect depth ordering for all but one of them.

7.6.4 Timings

Only limiting cases have been investigated.

With Figure B.132, the drawing with most lines, depth estimation using JLPs only takes approximately 0.10 seconds. Adding parallel line pair equations increases the time to 0.26 seconds. Adding face planarity equations instead increases the time to 0.18 seconds. Adding both increases the time to 0.34 seconds.

Figure B.74 has the largest number of parallel lines. Depth estimation using the Perkins equations only takes approximately 0.05 seconds. Adding parallel line pair equations increases the time to 0.19 seconds. Adding face planarity equations increases the time to 0.07 seconds. Adding both increases the time to 0.22 seconds.

It can be concluded that the ideas in this chapter lead to an implementation which runs acceptably quickly.

7.6.5 Conclusions

This Chapter emphasises the distinction between initial inflation, where minimal interference with the original x- and y-coordinates is important, and beautification, where the objective is perfect geometrical output. In concentrating on the specific requirements of inflation, it introduces one new compliance function (JLP), and an overall approach which also appears to be new, that of finding a least-squares fit to a linear system in which the only unknowns are z-coordinates, and in which inflation is achieved using a single primary compliance function (JLP or corner orthogonality) derived from considerations of orthogonality at vertices, supported by secondary compliance functions (line parallelism and/or four-vertex planarity).

Experimental results justify the overall approach. For drawings of typical engineering objects, correct depth ordering is often achieved and reasonable geometry obtained in acceptable time. While there are some special cases (e.g. Platonic and Archimedean solids) where the approach fails to achieve its objectives, these special cases do not correspond to common engineering objects and are easily identified.

Selection between JLP and corner orthogonality is less clear-cut. It was expected that corner orthogonality would produce better results for normalons and JLP for non-normalons; this is often true, but there are exceptions such as Figure 7.13 (page 137). Use of JLP can nevertheless be justified in that it is more robust and versatile—corner orthogonality can only be used for trihedral vertices which meet the Perkins criteria.

Of the options investigated, use of line parallelism is strongly recommended—there are drawings for which good output can only be achieved if line parallelism is used. Four-vertex planarity equations are required for drawings with hole loops; their use in other circumstances can result in minor benefits but is not essential. The presence or absence of *T*-junction entries in the JLP table is irrelevant.

In obtaining results in subsequent chapters, corner orthogonality is used for drawings with exactly three bundles of lines and where the 2D angles between bundles are all less than 75° , and JLP (without T-junction entries in the table) is used for all other drawings; line parallelism is used for all drawings; four-vertex planarity equations are used only for those drawings with more than one subgraph; and equations to place occluded lines behind occluding lines at T-junctions are omitted by

default but if the overall set of equations cannot be solved and there is more than one subgraph these equations are then included.

Chapter 8

Local Symmetry Detection

8.1 Introduction

In defining symmetry, two concepts are necessary: an *atom*, which may be anything which is indistinguishable except by location from any other atom of the same kind¹, and an *operation*, which changes the locations (and possibly orientations) of atoms. Following Kettle [66], *symmetry* is characterised by the fact that it is possible to carry out operations which, whilst interchanging the locations of some or all of the atoms, give arrangements of atoms which are indistinguishable from the initial arrangement. Rotation, reflection and inversion are thus *symmetry operations*. Rotation axes and mirror planes are termed *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 here termed *regularities*. Identification of a symmetry element or regularity in an object produces one or more *constraints*, which limit the possible locations of the atoms.

Lockwood [94] describes the aesthetic appeal of symmetry. Martin and Dutta [104] give various technical reasons why the possession of symmetry may be beneficial in designing shapes, and consider various approaches for making almost-symmetric shapes symmetric. It is therefore plausible that objects are intended to be symmetrical in some way. Symmetry and regularities implied by a drawing are used in two ways in later stages: in assisting the process of generating the topology of hidden parts, and in producing constraints to be met by the 3D geometry.

¹As noted already, the *atoms* of a solid object are its vertices, edges and faces.

While it is not possible to determine the symmetry of an entire object from a drawing which only shows part of it, clues to the structure of the object can be gained by considering the symmetry of each region. Each region is a skewed view of all or part of the corresponding 3D face—line lengths may not be preserved, and junction angles may be distorted, but parallel lines remain parallel within sketching tolerance. In the case of clues which suggest mirror (reflection) symmetry, these clues can be combined to deduce the presence or absence of reflection symmetry in the object—such combination is termed *chaining*.

Section 8.2 considers the history of symmetry detection, and explains how it is related it to the problem of graph isomorphism. Section 8.3 defines terminology and introduces ideas used later in the chapter, Section 8.4 describes figures of merit for the geometrical aspect of symmetry identification, Section 8.5 makes recommendations for implementing local symmetry detection, Section 8.6 makes recommendations for combining local mirror symmetry elements to detect object mirror symmetry, and Section 8.7 presents results obtained using these recommendations; this is the new work in this chapter.

8.2 History

As well as identifying many of the reasons why symmetry is beneficial, Martin and Dutta [104] classify the measures which must be taken to enforce it, distinguishing topological and geometric problems (see Figures 8.1 and 8.2). They also analyse some of the consequences of enforcing symmetry and some of the problems which may be encountered when an attempt is made to enforce a doubtful symmetry operation. Since many of the algorithms they consider derive from graph isomorphism and thus take no account either of geometry or of convexity/concavity of corners or edges, it is these problems which dominate their discussion.

Following on from this work, Mills et al [111] have solved the problem of identifying and enforcing global symmetry where a complete set of vertex coordinates is known but both topological and geometric errors may exist; the algorithm does not require prior knowledge of the symmetry element being sought, and does not require input tolerances (required tolerances are identified by the algorithm), but does require that the input data contains at least one data point relating to each vertex—it



Figure 8.1: Rectify Topology

Figure 8.2: Rectify Geometry

does not allow hidden vertices. Langbein et al [78, 79] have indicated methods for detecting partial global symmetries and regularities in similar data; rectification is an item of continuing research.

Neither approach is applicable to this thesis, where the assumptions are different (not all of the topology is known, but what is known is correct; 2D geometric information is reasonably accurate but the third, depth, coordinate is provisional). The approach of Parry-Barwick and Bowyer [121], which uses set theoretical models, can also be rejected on these grounds.

A polyhedron can be represented as a graph, with the vertices and edges of the polyhedron corresponding to the vertices and edges of the graph (Sugihara [161] calls the latter *nodes* and *arcs* in order to make the distinction clear). It is obvious that the graph can be embedded in the surface of a solid object of the same genus as the original polyhedron; it is also well-known that for the special case of genus zero polyhedra, the graph can be embedded in a plane.

Symmetry detection in polyhedra can therefore be translated to a graph isomorphism problem, where only the special case of graphs which can be embedded in a suitable surface need be considered. The general graph isomorphism problem is believed to be NP-hard (although no proof of this exists as yet), and no polynomial algorithm is known [28, 70]. However, polynomial-order algorithms are known for triply-connected planar graphs (corresponding to trihedral polyhedra with no through holes or hole loops): Weinberg's [189], which is $O(n^2)$, Hopcroft and Tarjan's [54], which is $O(n \log n)$, and Hopcroft and Wong's [55], which is O(n). The subgraph isomorphism problem, which is more directly relevant to this thesis as it corresponds to the situation where only part of an object is visible, is known to be NP-complete [31, 169].

Jiang and Bunke's algorithm [60] for rotational symmetry detection is based on graph automorphism. It creates and compares cyclic directed paths which traverse each half-edge once. It is fast— $O(n^2)$ with a small constant—and simple to implement. It allows for geometric as well as topological symmetry, but since it makes use of a geometric rotation matrix calculated from the first three vertices in the potential automorphism, it is severely intolerant of errors in subsequent vertex locations. Also, although the graph-theoretical aspects of the algorithm have been proved correct for objects with non-trihedral as well as trihedral vertices, in view of the way the rotation matrix is calculated, the algorithm will fail if the first three vertices in the automorphism are collinear, as could happen if the second vertex is extended trihedral or K-type tetrahedral. Jiang et al address this problem in a later paper [61], which also extends the idea to mirror symmetry detection. However, the algorithm remains restricted to connected graphs (i.e. polyhedra with no hole loops) and, most seriously of all for this thesis, it is restricted to entire polyhedra, not partial polyhedra, so cannot be used to identify partial automorphisms.

Sugihara [161] has extended the approach of Hopcroft and Tarjan [54] and has produced an $O(n \log n)$ algorithm for congruity of polyhedra which allows for non-trihedral vertices and through holes such as those in the objects in Appendix B.3, but not disconnected hole loops, as the requirement for a connected graph remains. Although the published algorithm does not allow for coordinate errors, Sugihara indicates a modification which would allow for these without affecting the order of the algorithm.

In attempting to extend this to partial isomorphisms, Sugihara observes that the NP-completeness arises when it is not known which parts of the two graphs are present and which are "hidden" (corresponding to parts of the object which are visible in the drawing but whose symmetrical equivalents are not visible). Sugihara proves that, given only the knowledge that one arc of the graph (one edge of the partial polyhedron) can be mapped to a visible arc of the corresponding isomorphic graph, a partial isomorphism can be found in $O(n^2)$ time. Finding the seed edge (where it and its symmetrical equivalent must be visible in both the original drawing) remains problematic—if all edges in the drawing are tried, the time obviously increases to $O(n^3)$. Furthermore, since the proof assumes use of standard graph theory techniques, the partial isomorphism, once found, will ignore non-graph-theory

considerations such as edge concavity/convexity.

Myers [114] has used genetic algorithms to solve practical partial isomorphism problems, but these are too slow for use in an interactive system.

The algorithm I have devised (outlined in [178]), while theoretically of higher order than Sugihara's idea, takes more account of the topological properties of polyhedra and thus may be closer to meeting the preference in this thesis for intuitively-correct methods. Geometry is not used in detecting the automorphism, but instead in assessing its merit once it has been found.

8.3 Compatibility, Pairing, Propagation

For two arrangements of atoms to be indistinguishable, the atoms occupying the corresponding locations in the two arrangements must be *compatible*. Compatibility must be defined for each type of atom, as follows.

For rotational symmetry, two vertices are compatible if, in the final object, they have the same underlying vertex type². For mirror symmetry, two vertices are compatible if, in the final object, one underlying vertex type is the mirror image of the other (note that all trihedral, and many tetrahedral, underlying vertex types are their own mirror images). Where the underlying vertex type of one or both vertices has not been determined unambiguously, the vertices are compatible if any subset of the possible underlying vertex types indicates compatibility.

Two corners of a face are compatible if they are both convex turns or both concave turns.

Edge compatibility takes account of direction—i.e. edge AB may only be compatible with edge CD if vertices A and C are compatible and vertices B and D are compatible, and may only be compatible with edge DC if vertices A and D are compatible and vertices B and C are compatible. Additionally, two edges are only compatible if, in the final object, they are both convex or both concave (occluding edges in the drawing will become convex).

Two faces are compatible in a particular orientation if they have the same number of edges, their edges when paired are compatible, and their corners when paired are compatible—see Figures 8.3 and 8.4.

²Possible underlying vertex types were determined in Chapter 4.

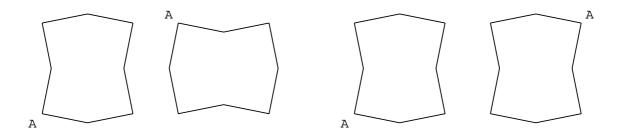


Figure 8.3: Incompatible

Figure 8.4: Compatible

Compatibility is a topological phenomenon, not necessarily related to geometric symmetry (for example, the object in Figure B.18, page 308, has the topological, but not the geometric, symmetry of a cube).

A pairing is an attempt (not necessarily a successful one) to identify corresponding atoms before and after a symmetry operation. Pairings are produced by seeding the pairing with an initial atom compatibility, and then propagating the consequences through the rest of the drawing. The pairing may be complete (each before atom is paired with exactly one after atom and vice versa), successful but incomplete (some atoms are paired, some are not, but no before atom is paired with two after atoms or vice versa) or unsuccessful (a before atom has been paired with two or more after atoms or vice versa).

An *impure* form of compatibility, which takes no account of edge vexity, is also useful in some circumstances—for example, reconstruction of the object in Figure 8.5 from a pairing seeded with an impure mirror plane (partial mirror symmetry marked in dotted lines) would produce the correct topology for the entire object (albeit with confusion as to whether the hidden edges should be convex or concave). The similarity to graph theory is evident—a complete impure pairing is the same as graph isomorphism in standard graph theory.

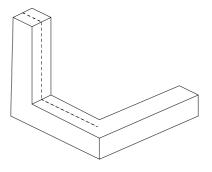


Figure 8.5: Z Block

RIBALD only uses pure atom compatibilities as seeds but impure compatibilities elsewhere do not stop propagation.

The algorithm I have devised for this is given in [178]. It is not optimal—from Sugihara's argument, a $O(n^2)$ algorithm should exist, whereas the algorithm in RIBALD is $O(n^4)$. Neither is it complete—there has not been time for geometric reasoning at non-trihedral vertices to be incorporated. Timings in Section 8.7 show that it is fast enough for the purposes of local symmetry detection; however, it is also used to implement ideas in Chapter 10, and for this a faster algorithm would be preferred.

8.4 Figures of Merit

The figure of merit for a candidate symmetry seeded by a face is an estimate of the likelihood that the region on which the symmetry is centred is a skewed view of the hypothesised symmetrical 3D face. Other factors which could contribute towards the figure of merit are: the frequency of occurrence of that type of artefact in freehand sketches; how well the artefact fits in with other knowledge about the sketch; and the complexity of the artefact.

The figure of merit for parallelism of two lines A and B has already been defined. Other primitive figures of merit are required, and are described in Appendix D.

Figures of merit for symmetries where the seed is an edge or a vertex are based on similar considerations.

In some cases, such as partially-occluded faces, it is not possible to determine whether or not the completed face has a particular symmetry or regularity. RIBALD assumes in such cases that the artefact is present, but assigns it a low figure of merit. An alternative approach might consider artefacts to have three states—completely present and thus likely; partially present and thus possible; and demonstrably absent on the basis conflicting evidence. This has not been pursued, as it is not clear that this is a sufficiently-detailed classification. For example, it has already been seen that an artefact present locally, but contradicted by evidence elsewhere in the drawing, can still be of use in local reconstruction of hidden parts; such an artefact should neither be rejected immediately nor accepted unequivocally.

Unlike compatibility, which is purely topological, figures of merit reflect the

likelihood of geometric symmetry.

8.5 Data Identified

Both rotational and mirror symmetry can be localised to a seed (inversion cannot, and is not detected), and this seed can be a face centre, an edge midpoint or a vertex.

8.5.1 Rotational Symmetry about a Vertex

Not all vertices need be analysed. Vertices can only be candidates for C_3 symmetry if their possible underlying vertex types include Yccc, Yddd, Xccccc, Xdddddd or Zcdcdcd. Those underlying vertex types identified in Chapter 4 which could lead to other rotational symmetries are shown in Table 8.1³. In addition, their faces must be compatible in the orientations which pair the vertex with itself (Lba junctions are provisionally assumed to be candidates for all symmetry operations). Occluding T-junctions cannot be seeds for symmetries—there will usually be a hidden vertex terminating the occluded line; since nothing is known about this vertex, nothing can be deduced about its symmetry implications.

Symmetry	Vertex Types
C_2	Xcccc, Xcdcd, Xdddd, Xcccccc, Xdddddd
C_3	Yccc, Yddd, Xcccccc, Xdddddd, Zcdcdcd
C_4	Xcccc, Xdddd
C_5	Xccccc, Xddddd
C_6	Xcccccc, Xdddddd

Table 8.1: Symmetrical Vertex Types

The figure of merit for rotational symmetry at a vertex meeting these criteria is obtained by multiplying the ratio of the number of faces visible at the vertex to the number which the symmetry would require by the figure of merit for the ratio of the 3D lengths of the longest and shortest edges meeting at the vertex.

Vertex rotational symmetry detection takes O(n) time, since in the worst case it must consider (a) each vertex, and (b) each edge arriving at that vertex.

 $^{^3}$ Note that vertices of type Zcdcd cannot have C_2 symmetry as they are chiral.

It is not clear that identification of vertex rotational symmetry is helpful in reconstructing the complete topology, so RIBALD includes it as an option.

In testing whether faces are compatible for vertex rotational symmetry, RIBALD assumes that partial faces are incompatible with any other face. Attempting to match the visible part of the partial face would in principle be better.

8.5.2 Rotational Symmetry about an Edge Midpoint

Where both faces bounded by an edge are visible, the faces are compatible, and the vertices at either end of the edge are also compatible, a candidate C_2 rotational symmetry operation can be identified located at the mid-point of the edge.

It is not clear that identification of edge midpoint rotational symmetry is helpful—useful occurrences of this type of symmetry are rare. RIBALD includes it as an option. No attempt is made to identify midpoint rotational symmetry for occluding lines.

8.5.3 Rotational Symmetry about a Face Centre

Candidate rotational symmetry operations $C_2, C_3, C_4, C_5, C_6, ...$ for each face are identified (RIBALD stops arbitrarily at C_6).

Firstly, a particular symmetry operation C_n is only a candidate if the face has n, 2n, 3n, ... vertices. Since the number of vertices of a partially-occluded face is unknown, these faces can generate candidate rotational symmetries provided that the visible part contains nothing to contradict the hypothesis, but these symmetries are given a low figure of merit (an object can be rotationally symmetric even if the axis of symmetry does not pass through at least one wholly-visible face or vertex—Figure B.455 is one example).

The candidate operation is rejected if the face in its original orientation is not compatible with itself in its rotated orientation—for example, C_4 is only a candidate symmetry operation for an octagonal face if the corners are either all convex or alternately convex and concave.

Similarly, the candidate operation is also rejected if edges on equivalent sides are not either all concave or all convex.

Even-numbered candidate symmetry operations $C_2, C_4, C_6, ...$ should also be rejected if opposing edges are not parallel to within the tolerance allowed for sketching inaccuracies. For a face of 5j corners to have C_5 symmetry, each set of five corners around the face $V_i, V_{j+i}, V_{2j+i}, V_{3j+i}, V_{4j+i}$ where $0 \le i < j$ must be a skewed regular pentagon: the undrawn line joining corners V_{j+i} to V_{4j+i} must be parallel to the base of the pentagon (the line joining V_{2j+i} to V_{3j+i}), and so on for each edge taken as base in turn. Similar rules are used for C_6 symmetry.

The base figure of merit F of any rotational symmetry hypothesis meeting the convexity and connectivity criteria is 1. This is multiplied by a factor representing how well the geometry meets the hypothesis. This could be based on the 2D geometry of the sketch, but if, as recommended, the frontal geometry is estimated before detection of local symmetry, additional information becomes available, and this is used instead. For each pair, cyclically, of vertices (V_0, V_j) , (V_j, V_{2j}) , etc., around the object, F is multiplied by the figure of merit for the ratio of the 3D lengths of lines from the centre to the vertex. This is simpler and quicker than an alternative tried, the figure of merit for parallelism between the vector from the face centre to vertex i and the vector from the face centre to vertex 0 rotated by $(360i/n)^{\circ}$. In the cases tested, the ratio of edge lengths also gave a subjectively better estimate of the merit of the symmetry.

For even-sided faces, F is also multiplied by the figure of merit for parallelism between opposite sides of the region. The resulting bias towards odd-sided symmetry hypotheses (where F is not reduced by this factor) is in principle undesirable (it is expected that objects of interest are more likely to have even rotational symmetry than odd) but in practice appears to cause no harm.

Using a factor based on the figure of merit for parallelism between opposite spokes has also been tried for even-numbered symmetries; this causes no problems but gains no benefits, and adds further bias to odd-sided symmetry hypotheses, so has not been retained.

Face rotational symmetry detection is O(n), since in the worst case it must consider (a) each face, and (b) each edge on that face.

8.5.4 Mirror Symmetry through a Vertex

Explicit detection of vertex mirror symmetry has not been implemented. A vertex implicitly has mirror symmetry if a face mirror chain passes through the vertex (see Section 8.5.6). RIBALD does not handle non-trihedral vertices correctly here: RIBALD will not chain a face mirror symmetry terminating at a vertex to another face mirror symmetry—this is correct for trihedral vertices, but for some tetrahedral vertex types, chaining would be valid.

8.5.5 Mirror Symmetry about an Edge

In view of the doubtful utility of identifying line C_2 symmetry, identification of line mirror symmetry was not a priority and RIBALD does not detect it.

This omission could cause problems at a later stage if the following circumstances apply: two faces F_a and F_b each have mirror symmetry terminating in vertices V_a and V_b respectively, and an edge connects V_a and V_b (Figure B.105 is one example of this). The decision as to whether or not this constitutes a mirror chain (see Section 8.5.6) should be decided by whether or not there is a plane of mirror symmetry along this edge. The omission could also make a difference in interpretation of drawings where there is mirror symmetry about a line but not about either of the regions which the line leaves.

8.5.6 Mirror Symmetry across a Face

Face-based mirror planes can be subdivided into vertex-to-vertex planes and edge-to-edge planes in faces with an even number of sides, and vertex-to-edge planes in faces with an odd number of sides. Partial faces are not considered—mirror planes across partial faces are deduced from other reasoning (see Section 8.6).

A face is a candidate mirror symmetry seed if all vertices, other than those (if any) which are in the symmetry plane, can be paired sequentially across the mirror axis (A in Figure 7.3, page 117), and lines joining these pairs (B1 and B2 in Figure 7.3) are approximately parallel with one another and centred on the mirror axis. The angle of the axis, and the average angle of the lines joining sequential pairs, are recorded—they are the input angles (α, β) required by skewed symmetry

for estimating face normals [63], an option which, although not part of the current version of RIBALD, has been used with success elsewhere [38].

The figure of merit for face mirror symmetry is the product of the figures of merit for the ratios of distances either side of the line of matched vertices and the figures of merit for parallelism between the lines joining these pairs of vertices and a line at the estimated angle β .

Face rotational symmetry detection is O(n), since in the worst case it must consider (a) each face, and (b) each edge on that face.

8.6 Mirror Chains

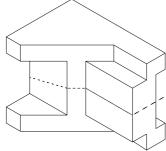
Candidate mirror planes can be chained: where a face mirror plane terminates at an edge mid-point, it can be chained with another face mirror plane terminating at the same edge as in Figure 8.5. Where a face mirror plane terminates at a vertex, it can be chained with a mirror plane along the edge leaving the face at that vertex, and vice versa, as in Figure 7.3 (see Section 8.5.4). Note that this is specific to trihedral vertices—RIBALD does not include a full extension to non-trihedral vertices.

The base figure of merit F_c for a chain is calculated from the *reinforced* merit for all mirror planes in the chain. This is reduced where expectations concerning the mirror chain are contradicted when the mirror pairing is propagated across the entire sketch:

- The fewer unpaired vertices this leaves, the more convincing is the evidence for the mirror chain. In addition, the longer a mirror chain is, the more convincing a successful propagation is as evidence of the mirror chain. F_c is multiplied by a factor $(V_p/V)^{2/C}$, where V_p is the number of paired vertices, V is the total number of vertices in the sketch, and C is the number of faces in the mirror chain.
- If the mirror chain ought to cross a fully-visible face, but that mirror plane has not been identified, the chain becomes doubtful. If an expected continuation mirror plane is missing, F_c is multiplied by a factor 1/2.
- Incompatible line pairings (convex vs. concave) make a mirror plane suspect. F_c is multiplied by a factor $1/(e^L)$ where L is the number of incompatible line

pairs (e (=2.718....) is chosen arbitrarily; it is "a number greater than 2", as this problem is considered more convincing evidence of the incorrectness of a mirror chain than the previous one).

It was intended that if a mirror chain terminates at an edge which also bounds a partial face, RIBALD should attempt to find the "best" mirror plane across this partial face to continue the chain by considering all possible mirror planes—see Figure 8.6. The criteria which would contribute to the merit of each potential mirror plane are listed in [178].



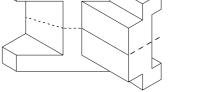


Figure 8.6: Extending Mirror Chain

Figure 8.7: Problem Extending Mirror Chain

At the time of writing, this idea had been removed from RIBALD as the implementation introduced a serious problem: the mirror chain in Figure 8.7 was completed by hypothesising that the partial face was a seven-sided face with an edge-to-vertex mirror plane. This is clearly not "best", and it seems likely that the idea can be retrieved with a more careful evaluation of figures of merit.

Results 8.7

In testing these ideas, the principle requirement is to show that the more plausible it is that a symmetry exists, the higher the figure of merit produced.

Sample results suggest that this is the case for mirror planes. In the three drawings of Grimstead's bracket, Figures B.91, B.92 and B.93, the merit figure for the obvious mirror plane is 0.7335, 0.7072 and 0.6355 respectively; by contrast, the merit figures for the second-choice mirror plane, vertex-to-vertex diagonally across the front L-shaped face, are 0.0422, 0.0488 and 0.0565 respectively.

The merit figures for C_5 rotations in the decagonal faces of Figure B.132, which is not well-drawn, range from 0.0050 to 0.4011; the merit figures for C_6 rotations in the hexagonal faces range from 0 to 0.0186. By contrast, the merit figures for C_5 rotations in the pentagonal faces of the better-drawn Figure B.120 range from 0.1088 to 0.9940, and those for C_6 rotations in the hexagonal faces from 0.1245 to 0.5902. This suggests that even-number rotations are indeed penalised unduly, both by comparison with odd-number rotations and with mirror chains, but that within a symmetry operation, good rotations are correctly preferred to bad ones.

Results for Figure B.173 show that there are problems outstanding with vertex rotational symmetry. RIBALD identifies that C_2 , C_3 and C_6 symmetries are possible at the central vertex, giving each a figure of merit of 0.4760. However, it also identifies that all five rotational symmetries are possible at the boundary vertices, assigning them merit figures from 0.4510–0.6489 (C_2 and C_3), 0.3383–0.4867 (C_4), 0.2706–0.3894 (C_5) and 0.2255–0.3245(C_6). By comparison, the merit figures for the C_3 face-centre rotations are in the range 0.4018–0.4997. It is clear that merit figures based on symmetries which make assumptions about hidden faces are still too high and should be reduced further.

RIBALD takes no longer than 0.01 second (the shortest time interval it can measure) to identify local symmetry elements (rotations and mirror planes) in any of the test drawings.

The longest time taken by RIBALD to form and assess the chains of mirror symmetry for any test drawing is 2.13 seconds for Figure B.132. Apart from other Archimedean solids, the time taken is reasonable—0.22 seconds for Figure B.74, 0.16 seconds for Figure B.554, 0.15 seconds for Figure B.454 and 0.11 seconds for Figure B.429 being amongst the most time-consuming. Almost all of this time is taken by propagating the pairings across the rest of the object in order to produce a figure of merit for the chain—pairing propagation is $O(n^4)$, and must be done for each mirror chain, making the process $O(n^5)$, whereas forming the chains is $O(n^2)$ with a small constant.

Chapter 9

Classification

9.1 Introduction

Although, ideally, this thesis is concerned with general methods of interpreting drawings, some special classes of object are both so common, and so much easier to interpret than the general case, as to make it worthwhile trying to identify them. For example, 24% of the drawings in Appendix B which can be labelled correctly are (or could be) extrusions, and 26% are normalons. Accordingly, I attempt to classify the object portrayed in the drawing into one of several such classes. A successful classification makes topological reconstruction (Chapter 10) more robust (and faster), and is especially useful in final geometric fitting (Chapter 11), where the general-case methods found so far are not yet robust enough nor fast enough to meet fully the aims of this thesis.

Classification hypotheses are based on an agglomeration of the local artefacts identified in Chapter 8 and the groups of parallel lines identified in Chapter 5. Since an object may meet the requirements of more than one of the classes, a figure of merit is assigned for each classification.

Section 9.2 outlines previous work in object classification from line drawings. Section 9.3 describes recommended classes and the criteria which a drawing must meet in order for the object to be included in each class, Section 9.4 describes how classes may be combined, and Section 9.5 summarises the results of implementing these ideas; these are the new work in this chapter.

9.2 History

Orthogonality of the faces of the sketched solid has been used as an optimisation criterion in other systems (and is even assumed by some). Lipson and Shpitalni [92] suggest that if a histogram of line angles is plotted and there are only three peaks, these probably correspond to the three perpendicular axes and the object is axis-aligned.

The Regeo project [15] also detect normalons as a special case; since their implementation interprets drawings with hidden lines shown, and does not allow for freehand drawing errors, identification of normalons is trivial.

Mills's algorithm [111] for global symmetry identification in point sets could be used for identifying regular solids and mirror-symmetric objects in hidden-line drawings, but has not been incorporated as part of any larger system. Since this method assumes that the entire object is known, it is not appropriate for interpretation of natural line drawings.

9.3 Classes

The special shape classes which may usefully be identified are discussed here. In addition to these, RIBALD implements a "none of the above" class, for which the figure of merit is the product of the figures of merit for the object not meeting each particular class.

9.3.1 Normalons

Normalons—objects where all face normals are aligned with one of the three coordinate axes—are reasonably common in engineering practice (one survey [143] suggests that 30% of parts are normalons). Identifying these simplifies topological reconstruction (Chapter 10) and makes the process of fitting face normals trivial (Chapter 11).

For the object to be a normalon, all edges must be aligned with one of the coordinate axes, so if the bundling process (Chapter 5) has identified more than three groups of parallel lines the object cannot be classified as a normalon. Also, in a normalon, there are four more convex turns than concave turns in the outer loop

of any fully-visible face (so, for example, the object in Figure 5.2, page 91, cannot be axis-aligned). For this purpose, collinear lines at extended trihedral vertices such as those in Figure B.156 on page 315 (and also at K-type vertices) are neither right turns nor left turns; although one of the faces in the figure has five vertices, the object portrayed is a normalon.

If the drawing meets these criteria, the base figure of merit for the object being a normalon is 1.0.

In principle, every junction in a drawing of a normalon must meet the Perkins criteria (Chapter 7). This is true of accurate projections but not necessarily of freehand line drawings—Perkins also observed that, empirically, drawings appear axis-aligned to the human eye if one set of faces is drawn rectangularly and the third axis as a diagonal—for example, Figure 9.1 meets the empirical but not the mathematical criteria for a normalon, and should be classified as such. Perkins went on to report [126] that while the human eye will reject assumptions which contradict these criteria if other, valid assumptions of rectangularity or symmetry are found elsewhere, it will sometimes impose rectangularity in defiance of the mathematical criteria if doing so creates some order in a non-rectangular and asymmetric drawing.

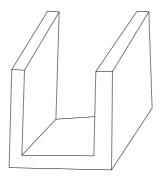


Figure 9.1: Technically Invalid Axis-Aligned Drawing

Since the object may still be a normalon even if the drawing fails to meet the Perkins criteria, the figure of merit for axis-alignment is reduced for each line which breaks these criteria (by multiplying by the figure of merit for parallelism between the line and the nearest mathematically-correct line), but the hypothesis is not rejected entirely.

The figure of merit is also decreased if the assumption of parallelism is uncertain it is multiplied by the parallelism figure of merit between each line and the mean orientation of lines in that bundle.

Identification of normalons as described here takes O(n) time. No alternative methods of identifying normalons or assessing figures of merit have been tested (alternatives involving generating face normals were considered but rejected as they involve extra processing and provide no obvious benefit).

9.3.2 Semi-Normalons

The requirements for semi-normalons relax the requirements for a normalon—most, but not all, face normals must be aligned with one of the three coordinate axes. Semi-normalons are common in engineering practice—one survey [143] showed that although only 30% of the parts covered could be described using axially-aligned blocks and cylinders, 85% could be described using axially-aligned blocks, wedges and cylinders (it is assumed that the axially-aligned cylinders are mostly drilled holes). Adding axially-aligned wedges to a normalon will in general give a semi-normalon, as in Figure B.91.

Semi-normalons are identified as follows:

- if there are more than N bundles of parallel lines, exit—the object is not a normalon (RIBALD uses N=6)
- list the sets of three bundles which meet at junctions of three lines
- count the number of junction bundle set occurrences (at junctions of three lines)
 or potential occurrences (at junctions of two lines) of each set
- if no set occurs more than any other, exit—the object is not a normalon
- the object might be a normalon—estimate the figure of merit

If the object is a semi-normalon, the three bundles which occur together more often than any other three are obviously the three mutually-orthogonal special bundles V, B_0 and B_1 (vertical and two in the base plane) described in Chapter 5.7; if V, B_0 or B_1 was originally some other bundle, it is updated to reflect this new knowledge about the object.

Initially, the figure of merit is calculated as for normalons. It is multiplied by the proportion of junctions which contributed to the count of the best set of three bundles.

Identification of semi-normalons, as described here, takes O(n) time.

9.3.3 Semi-Normalons with Mirror Symmetry

Classifying an object as a semi-normalon is of only moderate utility—it can help in reconstructing the topology, but it does not of itself give any clues about the geometry of non-axially-aligned hidden faces. In practice, many semi-normalons have a single predominant mirror symmetry which reflects axis-aligned edges to axis-aligned edges—55% of the test drawings in Appendix B which can be labelled meet this requirement. Since this combination provides enough information to deduce the face normals of many hidden faces, it is detected as a special case. The figure of merit is the product of that for semi-normalons (as above) and the highest figure of merit of any mirror chain (see Chapter 8).

If the drawing meets the requirements for this class, the figure of merit for this class, once calculated, is then subtracted from the figure of merit for the previous class (that of semi-normalon without mirror symmetry).

It would be simple to extend this idea further to a combination of semi-normalon with either a C_2 or C_4 rotation, which would also provide enough information to deduce the face normals of many hidden faces, but such objects appear to be less common in engineering practice. RIBALD does not identify semi-normalons with rotational symmetry as a class.

9.3.4 Artefact-Axis-Aligned Solids

Similar to the concept of a normalon is one where the mirror planes and rotation axes define three mutually-orthogonal axes—see for example Figure 9.2. Such objects seem to be relatively rare, and RIBALD does not identify such objects as a special class.

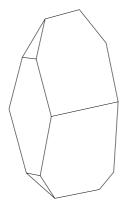


Figure 9.2: Tapered Wedge

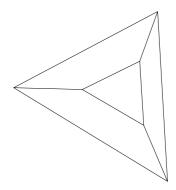


Figure 9.3: A Frustum

9.3.5 Extrusions

Right extrusions are a particularly common class of engineering objects (one catalogue [12] consists entirely of extrusions), and the topology and geometry are easily deduced—a right extrusion has two identical, parallel end cap faces joined by rectangles. More general extrusions are possible, but much less common.

The requirements for an extrusion are:

- no more than one fully-visible face (the front end cap) is other than a convex quadrilateral
- all vertices can be labelled as trihedral
- each vertex has at least one interpretation with no more than one concave edge
- all edges leaving the front end cap (the side edges) are in the same bundle
- all vertices are either on the front end cap or on side edges (occluding *T*-junctions need not be—Figures B.44 and B.413 show extrusions)—this may seem obvious, but cannot be omitted (for example, Figure B.108 meets all of the other requirements)
- each partial face may, when reconstructed, be a convex quadrilateral (i.e. no more than four visible vertices and no concave corners)

Note that, exhaustive as they appear, there is still a problem with these requirements, in that Figure B.223 meets them and is identified as an extrusion.

If these requirements are met, the figure of merit is the product of

- the figures of merit for each side line being parallel in 2D to the average orientation of this bundle
- the figures of merit for each line on the front end cap being parallel in 2D to the corresponding line on the back end cap (where this line is visible)

For drawings containing only quadrilaterals, RIBALD assumes that each region in turn is the front end cap of an extrusion, and picks the interpretation which produces the best figure of merit.

Identification of extrusions, as described here, takes O(n) time—although identification of quadrilateral extrusions appears to require $O(n^2)$ time, these are all topologically equivalent to a cube and are not the limiting case for large n.

9.3.6 Extrusions with Side Holes

The class of extrusions, as described above, includes objects with through holes in the direction of extrusion, such as Figure B.413, but not objects with side-to-side holes, such as Figure B.498 or Figure B.513. Although there are several figures in the test set with side-to-side holes, this is an artefact of the way the test data was generated—most of the through holes were cylindrical in the originals.

RIBALD does not identify extrusions with side holes as an additional class, partly because of the low frequency of such objects, and partly because of the difficulty of distinguishing drawings such as Figure B.415 from those such as Figure B.438—in the latter case, it is clear to a human that a pocket, not a through hole, is intended. This recommendation may need to be reviewed in future work if cylindrical holes are to be permitted. A more general class, that of extrusion with a single additional feature (boss, pocket, side-to-side hole, or slot), may be more useful—there has not been time to investigate this idea.

9.3.7 Frusta

Right frusta such as Figure 9.3 can be identified by similar criteria to those used for extrusions, although in this case, the lines joining the visible front face and the back face should, if extended, meet at a single point rather than being parallel. The figure of merit for the hypothesis that the object is a frustum depends on how close

to a single point these lines come, and is reduced if the object is likely to be an extrusion (this is in order to prevent the system misclassifying an extrusion as a frustum where the nearly-parallel lines meet a long way away).

For drawings containing only quadrilaterals, RIBALD assumes that each region in turn is the front end cap of a frustum, and picks the interpretation which produces the best figure of merit.

The figure of merit for an object being a frustum is the product of

- $1 F_x$, where F_x is the figure of merit for the object being an extrusion
- the figures of merit for each side line being parallel in 2D to a line joining its start junction to the apex of the extended frustum
- the figures of merit for each line on the front end cap being parallel in 2D to the corresponding line on the back end cap

Identification of frusta, as described here, takes O(n) time—although identification of quadrilateral frusta appears to require $O(n^2)$ time, these are all topologically equivalent to a cube and are not the limiting case for large n.

9.3.8 Platonic and Archimedean Solids

Although the Platonic and Archimedean solids such as Figures B.116 and B.119 are useful test cases for the ideas in Chapters 10 and 11 for handling general-case rotations, it is preferable in practice to treat them as a special class of object, particularly since neither bundling (Chapter 5) nor inflation (Chapter 7) handles them well. This class is identified as follows:

- If any face is only partially visible, exit—the object is not regular
- If any edge is concave, exit—the object is not regular
- If any vertex has no all-convex interpretation, exit—the object is not regular
- Initial set of possible regular objects = { all Platonic and Archimedean solids }
- For each vertex

- determine the number of corners of each face touching this vertex
- eliminate from the set of possible regular objects any for which the required
 numbers of faces is not a superset of the numbers at this vertex
- If the set of possible regular objects is not empty, the object might be regular assess the merit

RIBALD does not check the requirement for alternating faces such as would be required for a drawing to be interpreted as (for example) Figure B.127; errors resulting from this omission are noted in Section 9.5.

The base figure of merit for any drawing which meets the requirements is 1.0. This is multiplied by the mean merit for each face being a regular polygon: for odd-sided faces, this is the product of the figure of merit for parallelism of an edge and the undrawn line linking the two vertices either side of the opposite vertex; for even-sided faces, it is the product of the figures of merit for parallelism of opposite edges.

Identification of Platonic and Archimedean solids, as described here, takes O(n) time.

9.3.9 Rotationally Symmetric Solids

If the entire drawing is consistent with C_n symmetry, n > 4, the object could be classified as rotationally symmetric. However, the set of test drawings includes no examples in this class which could not equally well be classed as extrusions (Figure B.50 etc.) or Platonic or Archimedean solids. Thus, RIBALD does not identify rotationally-symmetric objects as a special class.

9.4 Combining Classes

If an object meets the requirements for more than one of the above classes, a judgement must be made about the best way of dealing with it. Some combinations of classes are possible, while other combinations are impossible, as listed in Table 9.1 ($\sqrt{}$ indicates compatible classes and \times indicates incompatible classes).

Where combinations of classes are impossible, an adjudication must be made. RIBALD makes this adjudication on the basis of figures of merit—although these

Class	N	S	M	X	F	R
N Normalon		×	×		×	
S Semi-normalon	×		×	×	×	×
M Semi-normalon + mirror	×	×			×	×
X Extrusion		×			×	
F Frustum	×	×	×	×		×
R Platonic or Archimedean		X	×		X	•

Table 9.1: Compatible and Incompatible Classes

figures are produced by different methods, in practice, as the results in Section 9.5 show, a numerical comparison is usually sufficient.

Where two classes are incompatible (for example, extrusion and frustum), the class with the higher figure of merit is chosen.

Where three classes are all mutually exclusive (for example, semi-axis-aligned, frustum and Platonic/Archimedean solid), again the class with the highest figure of merit is chosen.

Where two of three classes may be combined and the third is incompatible with either of the other two (for example, axis-aligned extrusions and frusta), the reinforced merit of the two compatible classes is compared with the figure of merit for the third in order to determine the best class combination.

More complex cases are adjudicated in similar fashion, with the merit of compatible classes being reinforced and the highest class or group of compatible classes being chosen.

Ideally, where two incompatible classes have similar figures of merit, a backtracking mechanism should be available so that if the preferred class resulted in an interpretation which was not the user's intention, a request for an alternative interpretation would produce one based on the second class. However, this idea has not been investigated in practice.

9.5 Results

Since the figures of merit for various classes are calculated by widely different methods, it is possible that some method of scaling them might be needed to make them

comparable. The investigation in this section concentrates on those drawings which meet the requirements for mutually-exclusive classes, in order to determine those classes for which the figure of merit is an under- or over-estimate relative to the others. Cases where the choice is solely between a semi-normalon with and without a mirror plane are not analysed, as this depends purely on the figure of merit for the mirror plane. Note, however, that RIBALD classifies Figure B.403 as semi-axis-aligned with mirror plane on the basis of the chain of mirrors across the front faces, indicating that there is further work to be done before this choice can be considered robust.

Results are presented in the following tables, in which columns F_i are the figures of merit for classes i in Table 9.1, e.g. F_X is the figure of merit for class X, interpretation as an extrusion. F_0 is the figure of merit for the object not belonging to any class. In each table, column C shows the class (or combination of classes) to which RIBALD allocates the drawing, and column I shows the class intended when the drawing was produced.

Drawing	F_X	F_F	F_0	С	I
B.537	0.97932	0.03635	0.01992	X	X
B.51	0.63871	0.42642	0.20723	X	X
B.52	0.17833	0.19304	0.66305	_	X

Table 9.2: Extrusion or Frustum?

Table 9.2 lists those drawings where the choice is simply between classification as an extrusion and as a frustum. Nothing has been placed in an incorrect class, but the failure to identify Figure B.52 as an extrusion suggests that extrusion figures of merit are unduly low—it is not obviously badly-drawn.

Extrusions can be normalons, and frusta cannot be, so identification that a potential extrusion or frustum might be a normalon is usually enough to ensure that the object is classified as an axis-aligned extrusion. Table 9.3 lists these drawings. Again, it appears that extrusions are undervalued, and this time two misclassifications result from this. Although neither Figure B.30 nor Figure B.46 is perfectly-drawn, in neither case should such drawing errors as are present result in the object being classed as a frustum rather than an axis-aligned extrusion. Note that although

Drawing	F_N	F_X	F_F	F_0	С	I
B.542	0.99891	0.99342	0.01298	0.00001	NX	NX
B.25	0.96886	0.90813	0.05916	0.00269	NX	NX
B.23	0.96082	0.87552	0.21925	0.00381	NX	NX
B.24	0.17036	0.00006	0.04886	0.78906	_	NX
B.30	0.37194	0.22544	0.84754	0.07417	F	NX
B.38	0.87618	0.69781	0.45423	0.02042	NX	NX
B.40	0.20526	0.51692	0.28414	0.27484	NX	NX
B.37	0.67656	0.52051	0.38900	0.09476	NX	NX
B.34	0.94123	0.72275	0.38932	0.00995	NX	NX
B.35	0.51903	0.31301	0.62965	0.12237	NX	NX
B.45	0.66804	0.67332	0.36897	0.06843	NX	NX
B.46	0.35253	0.11019	0.80957	0.10971	F	NX
B.237	0.27982	0.32148	0.28825	0.34780	NX	
B.246	0.27982	0.32148	0.28825	0.34780	NX	_

Table 9.3: Normalon Extrusion or Frustum?

Figure B.237 is incorrectly classed as an axis-aligned extrusion (the combination figure of merit is 0.51134) this is not necessarily a classification problem, as it results from a bundling error.

Drawing	F_N	F_X	F_F	F_R	F_0	С	I
B.17	0.88664	0.79063	0.35158	0.91180	0.00136	NXR	NX
B.11	0.95579	0.94015	0.08909	0.95821	0.00010	NXR	NXR
B.12	0.68180	0.52362	0.35878	0.70083	0.02908	NXR	NXR
B.14	0.68676	0.58544	0.65554	0.86110	0.00621	NXR	NXR
B.18	0.71296	0.52702	0.57456	0.76925	0.01333	NXR	NX
B.19	0.57161	0.91168	0.16378	0.81664	0.00580	NXR	NX
B.21	0.00022	0.98551	0.02847	0.98781	0.00580	XR	X
B.22	0.00000	0.00000	0.01812	0.09243	0.89113	_	_

Table 9.4: Regular Normalon Extrusion or Frustum?

Extrusion normalons can also be regular (the only regular extrusion is the cube). Table 9.4 considers drawings meeting these requirements. Clearly, Figures B.17, B.18 and B.19 do not show cubes; RIBALD should (but as yet does not) include a means of identifying that although a combination of classes is permitted it is not preferred. Furthermore, in five of the first six drawings, the single class with the highest figure of merit is that of regular solid, indicating that a further geometric

factor in addition to line parallelism is required in assessing the figures of merit of regular solids with quadrilateral faces.

Figure B.21 illustrates a further deficiency in RIBALD's current mechanism, in that in view of high figures of merit for the object being a regular solid and an extrusion, the object is classified as a regular extrusion; however, since the only regular extrusion is a cube, which is also a normalon, the fact that the object is clearly not a normalon should prevent this combination classification. Again, it can be noted that the figure of merit for regular solids with quadrilateral faces is overestimated.

Drawing	F_S	F_M	F_X	F_F	F_0	С	I
B.61	0.00000	0.81488	0.03408	0.74296	0.04596	MX	MX
B.57	0.00000	0.73261	0.06474	0.72377	0.06908	MX	MX
B.528	0.08886	0.66114	1.00000	0.00000	0.00000	MX	MX
B.529	0.29637	0.34000	0.99956	0.00017	0.00020	MX	MX
B.470	0.00000	0.71737	0.92103	0.14904	0.01899	MX	MX
B.450	0.07914	0.73905	0.99780	0.00077	0.00053	MX	MX
B.452	0.01280	0.78720	0.95804	0.03667	0.00849	MX	MX
B.137	0.00000	0.57516	0.00000	0.90155	0.04183	F	F
B.138	0.08541	0.31459	0.00005	0.97838	0.01355	F	F
B.31	0.00000	0.77893	0.00010	0.45161	0.12122	MX	NX
B.54	0.17183	0.32817	1.00000	0.00000	0.00000	MX	MX
B.55	0.17214	0.32786	0.36988	0.51562	0.16984	MX	MX
B.56	0.17143	0.32857	0.36988	0.51562	0.16980	MX	MX
B.58	0.05578	0.27756	0.93798	0.10939	0.03768	MX	MX
B.414	0.08964	0.57702	0.15848	0.27026	0.23646	MX	NX
B.49	0.00000	0.77068	0.97759	0.04338	0.00492	MX	MX
B.439	0.25140	0.30415	1.00000	0.00000	0.00000	MX	MX
B.440	0.27882	0.27673	1.00000	0.00000	0.00000	MX	MX
B.50	0.00000	0.54487	0.76266	0.32581	0.07283	MX	MX
B.36	0.42647	0.44020	0.00133	0.07853	0.29546	MX	NX
B.111	0.14661	0.35339	0.11215	0.00000	0.48992	_	_
B.60	0.00000	0.58882	0.25894	0.73269	0.08145	F	MX

Table 9.5: Semi-Normalon Extrusion or Frustum?

Extrusions can also be semi-normalons, while again frusta cannot. Obviously, a semi-normalon extrusion must have a mirror plane, so in those cases where the four possible individual classes are extrusion, frustum, and semi-normalon with or without mirror plane, the only combination class is (extrusion and semi-normalon

with mirror plane), and this is likely to be preferred for this reason. Table 9.5 shows drawings meeting these requirements. Only one is misclassified—Figure B.60 was intended to be a semi-normalon extrusion, not a frustum, and such drawing errors as are present should not be enough to alter its classification. In the misclassified drawing, as well as several others, it again appears that it is the extrusion figure of merit which is too low. The misclassification of Figure B.414 as a semi-normalon rather than a normalon is a bundling error, not a classification problem, and classification of the poorly-drawn Figures B.31 and B.36 as semi-normalons with mirror planes is about the best that can be expected.

Drawing	F_S	F_M	F_X	F_F	F_R	F_0	С	I
B.127	0.08407	0.31593	0.00000	0.00000	0.99998	0.00001	R	R
B.13	0.00000	0.60573	0.04395	0.03824	0.19711	0.29107	MX	MXR
B.308	0.27138	0.39528	0.00000	0.00000	0.76236	0.10470	R	MX
B.474	0.00000	0.83436	0.00000	0.00000	0.25019	0.12420	MX	MX
B.139	0.00000	0.49997	0.00000	0.25765	0.14740	0.31648	MX	F
B.312	0.12354	0.30504	0.00000	0.00000	0.75000	0.15228	R	MX
B.315	0.00000	0.65436	0.00000	0.00000	1.00000	0.00000	R	MX
B.267	0.00000	0.78211	0.02290	0.99948	0.67430	0.00004	F	MX
B.275	0.00000	0.78211	0.02290	0.99948	0.67430	0.00004	F	MX
B.179	0.03175	0.96825	0.00000	0.00000	1.00000	0.00000	R	R
B.115	0.00794	0.24206	0.00000	0.00000	1.00000	0.00000	R	R
B.20	0.00000	0.78210	0.33090	0.37939	0.41450	0.05298	MX	MX
B.167	0.00794	0.24206	0.00000	0.00000	1.00000	0.00000	R	MX
B.112	0.00000	0.69954	0.03137	0.56149	0.45669	0.06934	MX	
B.185	0.00000	0.78211	1.00000	0.00000	0.66667	0.00000	MX	MX

Table 9.6: Semi-Normalon Extrusion, Regular Extrusion or Frustum?

Regular solids complicate the issue—an extrusion can be either regular or a seminormalon, but not both. The fact that it is not a normalon should be a clue to it not being regular either, but RIBALD's omission of this inference appears to cause no harm (and actively helps in the case of Figure B.13). Table 9.6 shows drawings meeting these requirements. Figure B.267 is misclassified as a frustum because of an extremely low extrusion figure of merit. Figure B.112 illustrates RIBALD's determination to class everything as something if at all possible—it is in fact a well-drawn representation of a rectangular bar cut by an unaligned plane so as to leave an object with no symmetry, but RIBALD prefers any of three more symmetrical

interpretations to the intended asymmetrical interpretation.

Figure B.315 and the topologically-equivalent Figure B.308 are both identified incorrectly as being the regular solid shown in Figure B.127. Detecting face ordering would fix this problem (in Figures B.315 and B.308, two triangular faces at a vertex are adjacent; in Figure B.127 they are not), but it is noticeable that again the merit figures for regular interpretations of rectangular but obviously non-square faces are too high. Similarly, Figure B.312 is identified incorrectly as the regular solid in Figure B.128.

Figure B.167 is interpreted as an octahedron rather than a square pyramid (a semi-axis-aligned object with mirror plane); this, although legitimate, is undesirable, as Figure B.179 shows another way of drawing an octahedron but there is no alternative unambiguous way of drawing a square pyramid.

Figure B.139 is incorrectly classified as a semi-normalon with mirror plane rather than as a frustum, the intended interpretation. This will have no effect on topological reconstruction (except to slow it down), and may even improve final geometric fitting by ensuring that geometric constraints based on planes of mirror symmetry are enforced.

It can be concluded that classification is effective and in general correct, but that further work is needed to make the figures of merit for the various classes commensurate before selection of the preferred class or combination of classes can be considered robust. Three clear improvements on the current implementation in RIBALD can be noted. Firstly, the figures of merit for extrusions are undervalued. Secondly, the figures of merit for regular solids with quadrilateral faces should be reduced whenever it appears that the faces are not square. Thirdly, combination class figures of merit should take account of negative as well as positive inferences (e.g. an object cannot be a regular extrusion if it is not also a normalon).

Timing for object classification is satisfactory (RIBALD takes no longer than 0.01 second to classify any test drawing) and has not been investigated in any detail.

Chapter 10

Reconstruction of Hidden Topology

10.1 Introduction

Previous stages of processing make inferences concerning the visible part of a drawing: junctions and lines are labelled, a preliminary estimate is made of the visible 3D geometry, inferences are made about local symmetries and regularities, and the possible classes to which the object as a whole may be allocated: each such hypothesis has a figure of merit. The next (and most central) stage is to construct a topology which includes the hidden part of the object—as noted in Chapter 2, this topology must include a complete and consistent vertex-edge framework but need not include all face loops as adding these is straightforward. Section 10.2 describes previous work in this area.

Seeking the most plausible topology can be viewed as a search through the tree of possible topologies. Section 10.3 presents an analysis which shows that considering all possible topologies is impractical even for trihedral objects. It is therefore necessary to define: a control mechanism for the search (Section 10.4); the nature of a branch (Section 10.5); the means by which branches are generated (Section 10.6); and the means by which selection between branches is made (Section 10.7). Section 10.8 describes how the special classes identified in Chapter 9 allow all or part of the general-case mechanism to be bypassed. All of this work is new, except for those portions of Grimstead's work [38] which have been incorporated. Section 10.9

presents some results.

10.2 History

The problem of reconstructing topology from a natural line drawing was first addressed by Roberts [139]. The purpose of his system was to identify occurrences in drawings of three primitive objects, a cuboid, a wedge (an extruded right-angled triangle) and an axis-aligned L-block, and their spatial relationship with respect to one another (e.g. A is behind B, or A is supported by B). As Roberts's system allowed for lines to be omitted when faces of two primitives were contiguous and coplanar, it was capable of determining CSG representations from line drawings of polyhedra built from a small number of such primitives.

Wang and Grinstein [183] also produce a CSG representation of an object depicted in a drawing. This was originally restricted to normalons, with the single CSG primitive being a cuboid. Interpretation proceeds by the simple but effective means of adding cuboids until all visible edges are accounted for. Wang [182] later proved this approach to be correct in theory and demonstrated that it was effective in practice—it will always produce a valid CSG model from a valid drawing of a trihedral normalon. Wang does not describe safeguards against the error illustrated in Figure 10.21 in Section 10.9.2, where RIBALD produces a valid but incorrect interpretation. Although hole loops are allowed, he (like RIBALD's general case reconstruction) interprets all negative hole loop features as shallow pockets, not through holes, and does not describe how false labellings such as those in Figure 6.3 (page 103) are to be avoided. Oddly, although Wang incorrectly states that normalons must be trihedral, one of his illustrative examples, Figure B.432 (page 326), is extended trihedral¹.

Wang's extension [182] of this work to general polyhedra adds a second CSG primitive, a tetrahedron. This is less convincing. Firstly, it is not clear that a tetrahedron is a useful primitive—axis-aligned wedges as used by Roberts [139] are far more useful in practice [143]. Secondly, he allows non-trihedral vertices for general polyhedra and follows Malik's approach for labelling (see Chapter 4). It

¹Wang appears to use the Huffman-Clowes catalogue [14, 56] for labelling normalons, and does not indicate how this drawing was labelled—none of the variants tested in Chapter 4 labels this drawing correctly.

appears that Wang's ideas for general polyhedra were not implemented—no test results are presented.

Wang [182] also outlined how his ideas could be extended to allow simple curved primitives: axially-aligned cylinders, axially-aligned cones, and spheres. Again, it appears that these ideas have not been tested.

The perception that Wang's work has solved the problem of constructing CSG representations from natural line drawings, except for a few details, seems to have pre-empted further research into this area, and more recent work has concentrated on constructing B-rep models.

Grimstead's system [38] assumed that every hidden face in the object met an occluding edge. The resulting topological reconstruction of the hidden part of the object thus produced the simplest possible object, not necessarily the most plausible. Where three or more points on a hidden face existed in the 3D conversion of the visible drawing, the equation of the hidden face could be obtained directly. Where only two points existed, the equation was deduced on the basis of topological information. Edges were extended through T-junctions, and new edges created behind the visible object from visible L-junctions. Groups of three such edges were then tested for consistency—if the three edges were allocated to three faces, an additional hidden vertex was created where the three faces intersected, and the three edges were removed from the set of edges requiring completion.

Grimstead's algorithm for recovery of hidden parts was successful in the test cases he used. However, Grimstead acknowledges that knowledge of the algorithm makes it simple to find drawings for which the algorithm does not work correctly—Figures 10.4 and 10.5 on page 187 illustrate one such case.

Additionally, the assumption that all hidden faces have at least two visible vertices is not always plausible—it is simple to construct objects where a more complex but symmetrical reconstruction is psychologically preferable to the minimal reconstruction given by Grimstead's system. However, the minimal approach may enable the system to produce an interpretation (albeit an undesired one) of a drawing where a more complex system may fail altogether (for example, the dodecahedron in Figure B.116).

10.3 Number of Possible Completions

Superficially, the following outline algorithm appears simple and straightforward provided only that the number of acceptable topological completions can be limited in some way:

- Find each acceptable topological completion
- Assess each topological completion using a merit function
- Pick the completion with the highest merit value

Defining a limit to the number of acceptable topological completions is not difficult, as will be seen later in this Chapter. The number of possible topological completions is factorial in the number of additional atoms to be added, but this does not in itself rule the idea out. The question which must be answered is whether the number becomes unacceptably large for typical drawings. This section demonstrates that it does.

For simplicity of analysis, I consider drawings of trihedral objects, and reconstruction only of the vertex-edge framework, and ignore geometry. Completion of the framework can be treated as a single-player game, in which the player starts with a partial framework and must make a sequence of moves, each of which makes the framework "more complete", until the player wins (by producing a valid framework) or loses (by reaching a dead-end position which is incomplete but from which no further progress can be made). In trihedral frameworks, there are two indications of incompleteness, L-vertices (with two edges rather than three) and T-vertices (with one edge rather than three)²; reducing either the number of either constitutes progress, as does converting a T-vertex to an L-vertex (L-vertices are nearer complete than are T-vertices).

The legal moves available to the player, as illustrated in Figure 10.1, are:

A Add an L-vertex and two edges, making two existing L-vertices trihedral. This reduces by 1 the number of L-vertices, and adds 1 to the number of vertices and 2 to the number of edges in the framework. Note that this cannot be the move which completes the framework, as it leaves an L-vertex.

 $^{^{2}}$ The T-vertex is the true vertex at which the occluded edge at a T-junction terminates.

- **B** Add a trihedral vertex and three edges, making three existing L-vertices trihedral. This reduces by 3 the number of L-vertices, and adds 1 to the number of vertices and 3 to the number of edges in the framework. (This is a combination of the next move type with the preceding one.)
- C Add an edge joining two L-vertices. This reduces by 2 the number of L-vertices, and adds 0 to the number of vertices and 1 to the number of edges in the framework.
- **D** Extend the partial edge arriving at a T-vertex to an existing L-vertex, and remove the T-vertex from the framework. This reduces by the number of L-vertices by 1 and the number of T-vertices by 1, and adds nothing to the framework.
- E Turn a T-vertex into an L-vertex by adding an edge connecting it to an existing L-vertex (which becomes trihedral). This removes 1 T-junction from the framework, leaves the L-vertex count unchanged, and adds one edge to the framework. Note that this move cannot complete the framework, as it leaves an L-vertex.
- ${f F}$ Merge the partial edges arriving at two T-junctions. This removes two T-vertices and one edge from the framework, and leaves the L-vertex count unchanged. It adds nothing new to the framework.
- G Replace two T-vertices by a single L-vertex by extending the partial edges arriving at the two T-vertices until they join. This removes two T-vertices from the framework, and adds 1 to the L-vertex count (since two T-vertices are removed, movement is still towards completion). One vertex has been removed from the framework.

Adding an edge and a T-vertex to a single existing L-vertex is not allowed—it is a move away from completion, as T-vertices are less complete than L-vertices.

From the above, a partially-completed framework with exactly one L-vertex and no T-vertices loses the game, as does a partially-completed framework with exactly one T-vertex and no L-vertices: from neither position can progress be made.

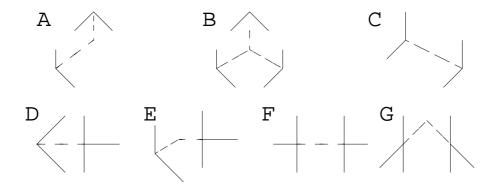


Figure 10.1: Allowed Moves

10.3.1 Cube

Illustrating the framework completion game as played with the cube in Figure B.11 demonstrates, firstly, a result important to the endgame theory of this game, and secondly, that analysing "toy" problems leads to misleadingly optimistic results.

The framework in Figure B.11 has seven vertices, nine edges and three faces. There are no T-junctions. Three of the vertices are L-vertices. In terms of the framework game, the starting position can be notated "3L,0T". Since a single edge cannot join three vertices, there must be at least one hidden vertex.

The options for the completed framework are therefore:

- Add one trihedral vertex and three edges, reducing the number of *L*-vertices to 0 and winning the game. The resulting framework has eight vertices and twelve edges, and thus six faces.
- Add one L-vertex and two edges, reducing the number of L-vertices to 2, giving a partial framework with eight vertices (2 L-vertices) and eleven edges. Then:
 - Adding a trihedral vertex and three edges is impossible as there are only two L-vertices.
 - Adding an L-vertex and two edges loses the game as it leaves the partial framework with one L-vertex, an irrecoverable position
 - Adding a single edge to join the two remaining L-vertices is thus the only winning move. This produces the same topology (eight vertices, twelve edges and six faces) as before.

• Adding a single edge to join two of the original L-vertices loses the game, as it leaves the partial framework with one L-vertex.

Thus the simplest possible interpretation of Figure B.11 is as an object with eight vertices, twelve edges and six faces, and this is provably the only interpretation using the game rules outlined above—all routes lead to the same destination. This result applies to any other partial framework with three *L*-vertices and no *T*-junctions.

Since adding a triconnected vertex and three edges can be decomposed into other move types, it is ignored in the remainder of this section.

10.3.2 Other Endgame Results

Section 10.3.3 will list the number of ways of producing a complete framework from various starting positions. This was calculated recursively by listing each possible move in the position and summing the number of ways of producing a complete framework from the resulting positions after making each such move. In order to ensure that this method terminates, various other endgame positions were analysed, and the results are listed in this section.

Obviously, a "2L,0T" position has just one completion, obtained by adding a single edge to link the two L-vertices.

There are two possible approaches to completing a "4L,0T" position. The four L-vertices can be paired in three distinct ways. Either each pair is linked by adding an edge, or each pair is linked by adding a new L-vertex and two edges, and the two new L-vertices are linked by adding an edge. As a result, there are six distinct completions available from a "4L,0T" position, three of which have the same number of vertices as the original and three of which have two more vertices than the original.

A "1L,1T" position has just one completion, reached by extending the edge through the T-vertex to join it to the L-vertex.

A "2L,1T" position also has just one completion—although there are three possible routes, they arrive at the same destination. Either an L-vertex and two edges (linked to the existing L-vertices) are added first, and the edge through the T-vertex then extended to this new L-vertex, or the edges are added to join the T-vertex to first one and then the other of the L-vertices.

A "3L,1T" position has six completions, although there are many more routes to them. Three can be reached by extending the edge through the T-vertex to one of the three L-vertices, and adding a single edge to join the other two L-vertices; these have one vertex fewer than the starting framework. The other three can be reached by converting the T-vertex to an L-vertex by adding an edge to link it to one of the three original L-vertices, and adding a final trihedral vertex and edges linking it to the three current L-vertices; these completions have one vertex more than the starting framework.

A "0L,2T" position has a single completion, reached by merging the edges through the two T-vertices.

A "1L,2T" position has a single completion which can be reached by three routes, one of which is to merge the two T-vertices into a single L-vertex and then add an edge linking the two current L-vertices. The completion has one vertex fewer than the starting framework.

A "2L,2T" position has six possible completions, reached by a variety of routes. Two can be reached by treating the position as two "1L,1T" positions; these have two vertices fewer than the start framework. Another two can be reached by converting one of the T-vertices to an L-vertex by adding an edge joining it to one of the original L-vertices; there are two ways of doing this, and each resulting "2L,1T" position has a single completion; the completion has the same number of vertices as the start framework. A fifth completion can be reached by treating the position as separate "2L,0T" and "0L,2T" positions; this has two vertices fewer than the start framework. The sixth completion can be reached by adding an L-junction and edges connecting it to the original two L-junctions; the resulting "1L,2T" position has a single completion; the completion has the same number of vertices as the start position.

A "0L,3T" position has a single completion—whichever *T*-vertices are merged first, the final framework is obtained by merging all three *T*-vertices into a single trihedral vertex.

A "1L,3T" position has six completions, with many routes to them. Three can be reached by merging the L-vertex with one of the three T-vertices and merging the edges through the remaining two T-vertices; these have three vertices fewer than the start framework. The other three can be reached by merging two of the

T-vertices into a single L-vertex and then solving the resulting "2L,1T" position; these completions have one vertex fewer than the start framework.

A "0L,4T" position has six completions, with many routes to them. Three can be reached by merging a pair of T-vertices to form an L-vertex, and solving the resulting "1L,2T" position; these completions have two vertices fewer than the start framework. The other three can be reached by pairing the T-vertices and merging the edges through each pair; these completions have four vertices fewer than the start framework.

10.3.3 More Incomplete Vertices

The number of alternative routes to producing a complete framework from different starting positions has been calculated, assuming that the results in the previous section are used for endgame positions, but ignoring converging routes otherwise. These are shown in Table 10.1. The columns show the initial number of T-vertices, and the rows the initial number of L-vertices. ∞ indicates that the number is larger than 2^{31} . Since different routes towards completion may converge, the numbers could be reduced significantly by modifying the approach so that results of positions which had already been analysed were cached.

	OT	1T	2T	3T	4T	5T
0L	n/a	n/a	1	1	6	70
1L	n/a	1	1	6	70	1140
2L	1	1	6	70	1140	25410
3L	1	6	70	1140	25410	743400
4L	6	70	1140	25410	743400	27677160
5L	70	1140	25410	743400	27677160	1278925200
6L	1140	25410	743400	27677160	1278925200	∞
7L	25410	743400	27677160	1278925200	∞	∞
8L	743400	27677160	1278925200	∞	∞	∞
9L	27677160	1278925200	∞	∞	∞	∞
10L	1278925200	∞	∞	∞	∞	∞

Table 10.1: Interpretations for Different Levels of Incompleteness

From the figures in Table 10.1, the number of interpretations increases factorially, so for large levels of incompleteness this simplistic approach is clearly inappropriate.

However, for many drawings which have been used as test cases, the number of possible completions is not large—Figure B.34 is a "4L,1T" problem, Figure B.91 is a "5L,1T" problem, and Figure B.66 is a "6L,1T" problem; for the last of these, analysing all possible completions could be onerous.

Figure B.71 is comparatively simple compared with most real engineering objects. It has 9 L-vertices and 2 T-junctions. Even this has more than 2^{31} interpretations—probably around 7×10^{10} .

Figure B.488 appears to be typical of engineering objects. It has 10 L-vertices and 4 T-junctions, and an estimated 4×10^{16} possible completions. Testing all of these is not a practical possibility.

Figure B.119 is about as complex a drawing as can normally be expected. It has 10 L-vertices and no T-junctions. Testing all 10^9 possible interpretations is undesirable but not completely absurd. However, none of the completions generated using the listed moves is the correct one—this drawing is one which requires moves which can add two hidden vertices at a time.

Figure B.74 has 25 L-vertices and 7 T-junctions. The total number of possible topological completions is probably greater than 10^{50} , and testing all of these is clearly not possible. The drawing is rather more complex than would generally be expected of freehand line drawings, but is not so unlikely that a system which cannot cope with it is acceptable.

It can therefore be concluded that generating all completions possible given a set of moves and picking the best one according to a figure of merit gives misleadingly good results for "toy" drawings such as those often used to test algorithms and ideas. It is out of the question for real engineering objects.

10.4 Control Mechanism

The idea of generating all reasonable topological completions and then picking the best as determined by assessing a figure of merit for each was shown in Section 10.3 to be impracticable even in the trihedral domain: generating all completions is factorial in the number of incomplete vertices and for realistic objects the number of possible completions is huge. For the general reconstruction problem, in which non-trihedral vertices are also allowed, there are clearly many more possible completions

(in the domain analysed using the line-labelling catalogue described in Chapter 4, in which hidden vertices or vertices deriving from junctions where one or more line is occluding may be tetrahedral or extended trihedral, even the cube has over 26 thousand possible interpretations). It is therefore necessary to search through the space of possible completions selectively. Several possibilities for the control mechanism of such a search are examined in this section.

10.4.1 Recognition of Known Objects

Several systems exist for choosing an object from a database of known objects given an input drawing ([146] is a recent example). This conflicts with the requirement in this thesis for reconstruction rather than recognition, but since it is possible that there will be common object topologies which defeat any method, for pragmatic reasons it may be preferable to recognise these particular topologies as special cases, extract their completed topological form from a database and adjust the geometry to match the drawing.

Reconstruction of parts of objects from common fragments, such as the features identified in Chapter 6, is discussed in Section 10.5.

10.4.2 Reconstruction Based on Classification

Given that most objects meet the requirements for special classes, as described in Chapter 9, it is possible to attempt to use different methods for reconstructing the topology depending on the object classification. For example, the topology of an extrusion can be completed simply from the visible end-cap.

There are two obvious problems with this approach. Firstly, several combinations of classes may simultaneously be valid. It is not always the case that special-case reconstruction completes the object—for example, in the case of semi-normalons with a mirror plane, hidden edges crossing the mirror plane will not be added. In attempting to use all the clues provided by different classes to the hidden topology, as the number of special classes increases, there may be a combinatorial explosion and using special-case methods for each valid combination will then be impractical.

Secondly, some drawings resist classification entirely—there must therefore be a general-case method to interpret these.

Ideally, the general-case method should handle as wide a variety of drawings as possible, minimising the need for special-case methods. However, special case methods may be faster and more robust, and may thus be a practical necessity in a realistic system.

10.4.3 Greedy Method with Fixed List of Choices

Straightforward greedy methods have been used successfully in simple systems such as Grimstead's [38]. Consider a process with a list of methods for making a sequence of moves towards completing the object. The list is ordered, such that if a move of type M_1 is possible it is made; thus a move of type M_n is made if and only if no moves of types M_1 to M_{n-1} are possible in the current state of the partially-completed object. After each move, the partially-completed object should be "more complete" (or at least no less complete) by some measurable criterion.

Various deterministic lists of moves ordered in this way have been investigated. For at least the moves considered, no ordering was found which works for *all* test cases, and it seems likely that human ingenuity in devising counter-examples will prove sufficient to defeat any which are proposed.

To illustrate this, consider the T Block (Figure 10.2) and the J Block (Figure 10.4). Two move types are used. T-junction completion extends the partial line through the T-junction to a hidden vertex, which is connected by hidden edges to the next visible edge around the partial face, and to the next L-junction around the background. Vertex completion by intersection connects two or three existing L-junctions with missing edges along differing axes by adding a hidden vertex and connecting it to the L-junctions using hidden edges.

If T-junction completion takes priority, the T-block is completed successfully: after T-junction completion, there are only three remaining L-junctions, and these can be linked by adding a single hidden vertex. However, if vertex completion by intersection takes priority, one of the vertices required for T-junction completion can be used mistakenly (see Figure 10.3).

In the case of the J-block (Figure 10.4), if vertex completion by intersection takes priority, the object will be completed successfully. It is only if T-junction completion takes priority that it fails: the background vertex to which the new hidden vertex

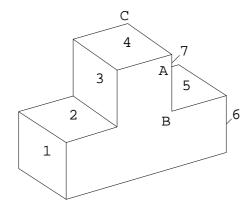


Figure 10.2: T Block

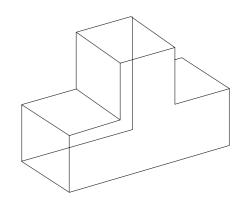


Figure 10.3: T Block Error

should be connected does not yet exist (see Figure 10.5).

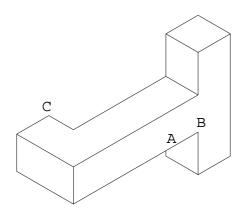


Figure 10.4: J Block

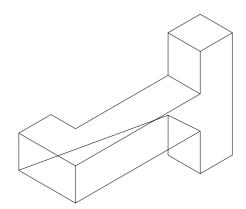


Figure 10.5: J Block Error

10.4.4 Greedy Method with Merit-Based Choices

Choice of move, if not determined by a predefined order of preference, can be generalised as merit-based, and is related to the idea of "pandemonium" [145], where demons looking for specific clues each shout their suggestions with varying degrees of enthusiasm, and the loudest suggestion is adopted. More formally, while the object remains incomplete, various possible moves towards completion are suggested on the basis of clues such as those identified in Chapters 5–9. These moves are assessed on the basis of figures of merit, the highest-rated is chosen, and the clues are re-assessed in case they are invalidated by the newly-made move. As with fixed-list methods, each move is designed to leave the partially-completed object nearer to, or at worst no further from, completion by some measurable criterion. Moves are described in more detail in Section 10.5 below.

The number of iterations required by a greedy approach is limited by the number of missing atoms which have to be added to the drawing to produce the completed object; it is assumed that this is no worse than proportional to the problem size, the number of edges in the drawing (the back is no more complex than the front, and may be simpler). Thus, if move generation, arbitration and execution take $O(n^i)$ time, topological completion using a greedy approach will take $O(n^{i+1})$ time.

10.4.5 Backtracking

With the greedy methods described above, it is possible that the process may reach a state where it can be identified that it is no longer possible to reconstruct a valid object—for example, if the object contains only two incomplete junctions and there is already an edge linking them, no number of further additions can create a valid object, and something must be removed. Immediate dead-ends such as this can be detected explicitly, but identifying unavoidable future dead-ends is impractical. In order to guarantee that the process always produces a valid object from a valid drawing, a backtracking mechanism is required. A backtracking mechanism allows a chosen move to be rejected not only immediately, if after making the move the object is invalid (see Section 10.7 for some examples of this), but also if after following all branches of the resulting tree of moves, none of these results in a valid object. The former has been found useful, but the latter is only sporadically useful—it is more common for the system to produce an incorrect topological completion (one which is valid, but not the one expected by the user). This would not cause backtracking to be invoked, as the system would be unaware that anything was wrong.

Backtracking slows topological reconstruction, sometimes unacceptably so—in principle, the entire search space may be traversed, and as shown in Section 10.3, this is of factorial order.

10.4.6 Recommendations

Since the system must be able to attempt a reconstruction of arbitrary drawings which fall into no predefined class, it must include something resembling the merit-based greedy method described in Section 10.4.4. Although backtracking is easily-implemented and its inclusion is recommendable, an interactive system must only

use it as a last resort for unusually hard drawings.

In implementing the merit-based greedy method, to avoid the potential for a tree growing indefinitely, hypothesis formation should be abandoned if, while the object remains incomplete, there are already more hidden atoms than visible ones. If this occurs, RIBALD deems the completion to be in violation of the assumption that the object is drawn from its most informative viewpoint, the current topological completion is rejected, and RIBALD backtracks to the last state at which there was a plausible alternative.

The ideas of incorporating recognition of specific known objects and reconstruction based on classification for specific special cases of object is considered further in Section 10.8.

10.5 Move Types

My early experimentation [173], restricted to trihedral objects, distinguished between vertex-based moves as described in Section 10.3, aimed at completing the vertex-edge framework, and face-based moves, aimed at adding a complete face at a time to the partial object. The conclusion reached there was that, although vertex-based moves are more generally reliable, there are some classes of drawing which cannot be reconstructed using these moves, so a single mechanism should be used which is capable of handling all useful types of move. This recommendation applies equally well to the more general non-trihedral domain, and is adopted here (despite the problems it creates for move arbitration, described in Section 10.7). Several useful move types have been identified, and are described in this section. A figure of merit is associated with each move; the numerical value depends on how the move is generated (see Section 10.6), and use of these figures or merit is described in Section 10.7.

After acceptance and execution of a move, the topology is reassessed. For example, at any potentially-incomplete vertex with the same number of edges and faces, if each edge is adjacent to two existing faces, the vertex is now complete and its underlying type is known unambiguously.

10.5.1 Creation of a New Vertex

As the aim is to reconstruct the vertex-edge framework of the object, the most obvious move towards completion is to add a new vertex and sufficient edges to link it to two or three existing incomplete vertices, as described in Section 10.3. Adding a new vertex and one edge to link it to one existing incomplete vertex is not a move towards completion and is not used.

Assuming that the number of incomplete vertices is proportional to the problem size, there will be $O(n^2)$ such moves available where two edges are to be added and $O(n^3)$ such moves available where three edges are to be added, so there is a good case to be made for not using the latter except in special circumstances such as when the framework can be completed in a single move.

Addition of a vertex and two edges can be subdivided into three variants, depending on whether (and how many) occluding T-junctions are involved. In the basic case, a new vertex is created, and two edges are added to link it to two incomplete vertices (not necessarily L-vertices, in the general case). Alternatively, one of the edges may be one terminating at an occluding T-junction; this is extended to the location of the new vertex, which replaces the T-junction, and a single new edge is added to link the new vertex to an existing incomplete vertex. In the case where both edges terminate at occluding T-junctions, the new vertex replaces both T-junctions, both existing edges are extended, and no new edge is added. (Vertex-plus-3-edge moves would require four variants—this is a further practical incentive to avoid using them.)

Whenever a new hidden vertex is hypothesised, a prediction is made concerning its location (as with z-coordinates of visible vertices, this is a provisional estimate; if the hypothesis is accepted, the provisional vertex location may be used in assessing the merit of subsequent hypotheses; it will also be used as a starting-point for the geometric fitting methods described in Chapter 11). Since it is found that some hypotheses are better than others at predicting vertex locations accurately, this prediction has its own figure of merit. For trustworthy hypotheses, this may be equal to the figure of merit of the hypothesis as a whole; for hypotheses such as mirror chains, which are good at predicting topology but poor at geometry, it may be considerably lower. Whenever equivalent hypotheses are merged (see Section 10.7),

the predicted locations are also merged taking account of this latter figure of merit. When a hypothesis is accepted, any new vertices created are placed in the location corresponding to the prediction.

As will be seen in Section 10.6, it is sometimes the case that a vertex-plus-2-edge move is produced by a hypothesis which requires this new topology in order to complete a face; this being so, it is simplest to create the face too at this point. RIBALD only implements this idea for quadrilateral faces.

10.5.2 Creation of a New Edge

Creation of a single new edge linking two incomplete vertices is also obviously a useful move towards completing the vertex/edge framework. There will be $O(n^2)$ such moves available.

As with moves creating a new vertex, there are three variants of the move creating a new edge, depending on the involvement of occluding T-junctions. In the basic case, a single new edge is added, linking two incomplete vertices. Where an existing edge terminates at an occluding T-junction, the T-junction is removed and the edge is replaced by one linking the originating vertex with an incomplete vertex. Where two existing edges terminate at occluding T-junctions, both T-junctions and both edges are removed and replaced by a single edge linking both originating vertices.

Again, as with moves creating a new edge, moves creating a single edge can be produced by a hypothesis which requires this edge in order to complete a face, and it is simplest to create the face too at this point. RIBALD only implements this idea for quadrilateral faces.

10.5.3 Creation of a New Face

As noted in Chapter 2.14, adding face loops to a complete vertex/edge framework is straightforward, reliable and quick, so in making a case for deferring all face creation until the vertex/edge framework is complete it can be pointed out that creating these using the low-order polynomial algorithm described there is preferable to creating them using the much higher-order general topological reconstruction algorithm. However, there will be situations where it is obvious that a face should be created from a particular loop of vertices or edges, and it is possible that creating

this face as soon as it is obviously required will reduce the potential for mistakes later in the search for the best topology.

In order to investigate which method works better in practice, RIBALD implements moves which create a single new face from a loop of four, five or six vertices, adding any new edges which may be required. Since, as implemented, these candidate moves are only generated from applying hypothesised symmetry operations (of which there are O(n)) to existing face loops (of which there are also O(n)), there are $O(n^2)$ such moves available. As noted in Section 10.3, the frameworks of some partial objects cannot be completed by the new vertex and new edge moves already described, but (as described in [173]) can be completed by adding new face loops which include new vertices.

In order to be able to complete such objects, RIBALD implements two types of move for adding a face plus edges and vertices.

Firstly, a parallelogram face can be created given one edge linking two incomplete vertices and a single hypothesised vertex location; two new vertices and three new edges will be added, and a face created from the resulting loop of edges. Although there are an infinite number of possible vertex locations, all candidate moves creating a parallelogram face starting from the same edge will be merged (as described in Section 10.7), so there are O(n) such moves. The base figure of merit for such a move is given by tuning constant T_q .

Secondly, a face matching an existing face can be created given at least three existing vertices on the face to be created which match the corresponding vertices on the template face. The remaining vertices, and any edges required, are created, and a face created from the resulting loop of edges. Since, as implemented, these candidate moves are only generated from applying hypothesised symmetry operations (of which there are O(n)) to existing face loops (of which there are also O(n)), there are $O(n^2)$ such moves available.

10.5.4 Reconstruction from a Symmetry Element

A "macro-move" which reconstructs as much topology as possible from a single symmetry element will obviously improve performance in terms of speed, and will also improve performance in terms of robustness (provided that the symmetry element

identified is genuine).

If such a macro-move is accepted, the symmetry element is propagated across the entire object (as described in Chapter 8.3), and new atoms (vertices, edges and faces) are created wherever an existing atom has no equivalent already in the framework.

The current version of RIBALD does not complete partial faces which form part of the mirror chain—an earlier version which tried to do this did so incorrectly (for example, producing an odd-sided face in Figure B.443), and there has not been time to produce a correct implementation.

RIBALD implements macro-reconstruction from a symmetry element only for mirror chains. An earlier version of RIBALD [173] performed macro-reconstruction for mirror chains before entering the main topology reconstruction mechanism; this idea is no longer preferred, as implementing this as a move allows other, even higher-merit, moves to be performed first (in which case the result will either reinforce or contradict the hypothesis of a mirror chain), and also allows macro-reconstruction from more than one mirror chain in objects with multiple symmetries.

This move type will not deduce hidden faces or edges bisected by the mirror plane; these must be added by later iterations.

10.5.5 Pre-interpreted Sketch Fragments

Draper [23] suggests making use of pre-interpreted picture fragments in drawing reconstruction—for example, in the case of the T-piece problem illustrated in Figure 10.21 on page 218, the correct solution could be hypothesised as a single move.

RIBALD implements macro-moves corresponding to slot and pocket features identified in Chapter 6.

The topology implied by a rectangular underslot is completed by adding four vertices which form the hidden end of the slot, four edges (all in the same bundle) joining these hidden vertices to the corresponding four visible vertices at the visible end of the slot, three edges joining the hidden vertices (the edges are in the same bundle as the corresponding edges at the visible end of the slot), and three faces (two slot walls and a slot ceiling).

The only contentious issue associated with underslot completion is the length

of the four parallel edges—although this is not a topological question, a reasonable estimate of the geometry is required as geometric information may be used in assessing the merits of subsequent moves. RIBALD assumes that these edges are the same length as the nearest wholly-visible edge in the same bundle to a visible vertex of the underslot; no alternatives have been investigated.

Completion of the topology of a valley is similar but easier, in that the length of the valley is usually known.

Chapter 6 did not distinguish holes and pockets—in either case, the mouth is a hole loop. RIBALD's macro-hypothesis assumes a pocket, completing it by creating an identical loop of vertices and edges, side-edges joining the mouth vertices with the bottom vertices, and side and bottom faces. Bottom vertices and edges may already exist. Where the subgraph includes a genuine vertex other than those forming the mouth of the pocket, this is assumed to be at the bottom of the pocket. In this case, the depth of the pocket is known. Where the depth of the pocket is not known, it is estimated in the same way that the length of underslots was estimated—RIBALD uses the length of the nearest wholly-visible edge in the same bundle to the pocket mouth.

RIBALD does not try to create through holes; the problem of determining the topology of a second hole mouth is as yet unresolved (in general, the second hole mouth will form a hole loop within a hidden face, and testing geometrically which face it emerges in is not difficult; however, in some objects, the emerging hole will split edges, as can be imagined by inverting the object portrayed in Figure B.408).

There is no move type corresponding to bosses—it was intended that these would be dealt with by splitting the object into two (or more) pieces, as described in Chapter 2.16. There has not been time to test this idea.

10.5.6 Complete Already

In general, since non-trihedral vertices are allowed, it is not be possible to determine with certainty whether or not the vertex/edge framework is complete. It will, in general, be legitimate to add an extra edge to a complete framework (for example, one which splits a quadrilateral loop into two triangles). The hypothesis that the framework is already complete must therefore be weighed against competing hypotheses,

and to this end RIBALD treats it as a move type.

10.6 Hypothesising Moves

Candidate moves are generated on the basis of hypotheses about the unknown part of the object deduced from what is already known—each hypothesis may generate one or more moves (inappropriate hypotheses may even generate none). This section describes how hypotheses are used to generate moves.

Figures of merit specific to hypotheses are described here. Some adjustments made to figures of merit are common to all hypotheses. These are described in Section 10.7.

10.6.1 Edge Extrapolation Hypotheses

Given the immediate objective of completing the vertex/edge framework, it is desirable that there should be at least one move generated wherever there is an obviously incomplete vertex. To this end, 2D lines are extended from incomplete vertices, in the same manner as in Grimstead's system [38]. These lines are extended both (i) from true incomplete (or potentially incomplete) vertices, in which case the vertex itself is the *originating vertex* for the line and several alternative lines may be generated in different directions, and (ii) from T-junctions, in which case the vertex at the other end of the defining line of the T-junction is the originating vertex, and a single line is extended only along the existing edge.

As can be seen in Figure 10.6, this is comparatively simple when only trihedral vertices are allowed (see [173]). It is more complex when non-trihedral vertices are allowed for two reasons: firstly, it is not always clear how many additional edges are required at a vertex; and secondly, in the cases of K-vertices and Z-vertices, additional edges will be extensions of existing edges, and thus bundled together with an edge already arriving at the vertex (something impossible with trihedral vertices).

The former concern is addressed by ensuring that the figures of merit for lines extrapolated at a potentially-incomplete vertex sum to $(E_{min}-E)+T_e(E_{max}-E_{min})$, where E is the current number of edges at the vertex, E_{min} is the minimum number

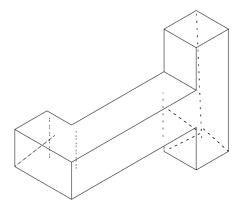


Figure 10.6: Extrapolated Lines for J Block

of edges required by the vertex type, E_{max} is the maximum number of edges required by the vertex type, and T_e is a tuning constant.

Where the incomplete vertex cannot be a K-vertex or Z-vertex, RIBALD extrapolates lines along all bundles not already used by edges arriving at the vertex, and divides the total figure of merit amongst them in proportion to the number of edges in each such bundle.

Where the incomplete vertex is known to be a K-vertex or Z-vertex, and an extension of an existing edge is clearly required, the figure of merit for this is 1; remaining merit (if any) is divided equally amongst extrapolated lines along other bundles, as above.

Where the incomplete vertex may or may not be a K-vertex or Z-vertex, the figure of merit for extending an existing edge is T_z (another tuning constant); remaining merit is divided equally amongst extrapolated lines along other bundles, as above.

The figure of merit for an edge through a T-junction is 1. Since the z coordinates of T-junctions are unreliable, the mean 3D direction for edges in this bundle is used rather than the 3D direction of the defining edge of the T-junction.

Geometrically, an extrapolated line is defined by the coordinates of its originating vertex and the 3D vector associated with the bundle of parallel lines (see Chapter 2.9).

Moves are generated by considering each pair of extrapolated lines. If the two lines are bundled together, the hypothesis is that they are really the same edge; a new-edge move is generated to connect the two originating vertices. The initial new-edge figure of merit is the product of the two extrapolated line figures of merit

and the figure of merit for 3D parallelism between a vector joining the two vertices and the mean vector for edges in the bundle. This initial figure of merit will be adjusted, as described below, for geometric plausibility and numbers of crossings.

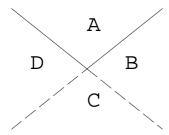
Otherwise, a new-vertex move is generated where the two lines intersect. The predicted x- and y-coordinates of the hypothesised vertex are the coordinates of the intersection; the z-coordinate is predicted by taking the mean of values obtained by extending 3D lines along the bundle vectors from the two originating vertices. The initial new-vertex figure of merit is the product of the two line figures of merit, a bias for crossings including known lines (1 if either line is extended through a T-junction, tuning constant S_x otherwise), and the two figures of merit for parallelism between the mean bundle vectors and the actual vectors between the two originating vertices and the hypothesised new vertex. Again, this initial figure of merit will be adjusted for geometric plausibility and numbers of crossings.

When a crossing location occurs outside the object boundary, it is likely that the hypothesis generating it is incorrect—this is common to all hypotheses, and is discussed in Section 10.7 below.

However, some moves can be ruled out geometrically before being generated and without considering all faces of the object. Consider, for example, Figure B.45. It is clear that however many new edges are to be created at any of the *Lba* junctions, they must all leave their originating vertex in a direction which is within the angle of the L. Similarly, however many new edges are created at the *Lab* junction, they must all leave the vertex in a direction which is outside the angle of the L. This concept can be extended to all visible vertices—see Tables 10.2–10.6 for those junction labels for which RIBALD tests for sensible arcs (arc identifiers are shown in Figures 10.7–10.11). Whenever the target (the other end for a new-edge move or the new vertex for a new-vertex move) is outside the sensible arc, the move is abandoned.

RIBALD implements a similar concept for lines through T-junctions. For these, it is required that $\mathbf{n} \cdot \hat{\mathbf{t}} > |\mathbf{t}|$, where \mathbf{n} is the vector from the originating vertex to the target and \mathbf{t} is the vector from the originating vertex to the T-junction—the target is in the same general direction from the originating vertex as is the T-junction, and the true edge is at least as long as the part of it which is visible in the drawing.

It was suggested in [173] that moves based on lines which cross many other lines should be lower-merit than moves which cross few other lines. To this end,



A B C E D

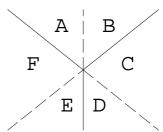
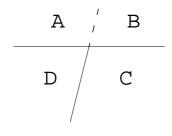


Figure 10.7: Arc Labels for L-junctions

Figure 10.8: Arc Labels for W-junctions

Figure 10.9: Arc Labels for Y-junctions



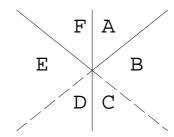


Figure 10.10: Arc Labels for T-junctions

Figure 10.11: Arc Labels for Z-junctions

RIBALD multiplies the merit of moves from crossing and merging hypotheses by $(m_A m_B)/(\sum m_{iA} \sum m_{jB})$, where for every pair of lines A and B, m_A is the merit of line A, m_B is the merit of line B, $\sum m_{iA}$ is the sum of the merits of all lines crossing or merging with line A, and $\sum m_{jB}$ is the sum of the merits of all lines crossing or merging with line B. It is not clear that this is necessary or desirable—it is possible that omitting this stage entirely would have no damaging effect on results, and a sound case can be made for multiplying only by the factor for the less-crossed line

Label	Arc A	Arc B	Arc C	Arc D
Lba	Y	N	N	N
Lab	N	Y	Y	Y
Lac	N	N	N	Y
Lcb	N	Y	N	N
Lbd	Y	N	N	N
Lda	Y	N	N	N
Laa	Y	N	N	Y
Lbb	Y	Y	N	N

Table 10.2: Sensible Arcs for L-Junction Labels

Label	A	В	С	D	Е	F
Wbca	Y	Y	N	N	N	N
Wbda	Y	Y	Ν	N	Ν	N
Wbcd	Y	Y	Ν	N	Ν	N
Wdca	Y	Y	N	N	N	N
Wbaa	Y	N	N	N	N	N
Wbba	N	Y	N	N	N	N
Wbdc	Y	N	N	N	N	N
Wcda	N	Y	N	N	N	N
Wcac	Y	N	N	N	N	N
Wcbc	N	Y	N	N	N	N
Wdac	Y	N	N	N	N	N
Wcbd	N	Y	N	N	N	N
Wdad	Y	N	N	N	N	N
Wdbd	N	Y	N	N	N	N
Wacc	N	Ν	Ν	N	Ν	Y
Wccb	N	Ν	Y	N	Ν	N
Wadc	N	Ν	Ν	N	Ν	Y
Wcdb	N	N	Y	Ν	N	N
Wdcb	N	N	Y	Y	N	N
Wacd	N	N	N	N	Y	Y
Wabd	N	N	Y	Y	Y	Y
Wdab	N	N	Y	Y	Y	Y

Table 10.3: Sensible Arcs for W-Junction Labels

(the higher of $m_A/\sum m_{jB}$ and $m_B/\sum m_{iA}$). Since the optimum values of tuning constants would inevitably be different for these alternatives, and determining such optimum values is time-consuming, they have not been investigated.

Since the resulting geometry is relatively reliable for normalons but less so for non-normalons, the figure of merit for the hypothesised vertex coordinates depends on this, being $F_T^{G_x}$ for normalons and $F_T^{G_y}$ for non-normalons (where F_T is the figure of merit for the topological move and G_x and G_y are tuning constants).

In practice, if lines are extended wherever a vertex's permitted underlying vertex type might be non-trihedral (for example, a Wbca junction can be interpreted as all-convex trihedral, all-convex tetrahedral, or 3-convex+1-concave tetrahedral), the number of lines extended is unduly large, the number of line crossing hypotheses is excessive, and topological reconstruction becomes slow and (since the chance of a particularly bad hypothesis being accepted is increased) less robust. Various options

Label	A	В	С	D	Е	F
Yabd	N	N	Y	Y	Y	Y
Yaab	N	Ν	N	Ν	Y	Y
Yabb	N	N	N	N	Y	Y
Yacc	N	N	N	N	N	Y
Yacd	N	N	N	N	N	Y
Yadd	N	N	N	Ν	Y	Y
Ybdd	Y	Y	Ν	Ν	N	N
Ybcc	Y	N	N	N	N	N
Ybdc	Y	N	N	N	N	N
Yabc	N	N	Y	Y	Y	Y

Table 10.4: Sensible Arcs for Y-Junction Labels

for reducing the number of lines extended have been investigated.

One such idea is for the set of permitted underlying junction types to be chosen to be the trihedral types plus the set of the single simplest interpretations of each evidently non-trihedral vertex (for example, all Wbca junctions must be interpreted as all-convex trihedral unless there are junctions for which the single simplest interpretation is all-convex tetrahedral or 3-convex+1-concave tetrahedral; if there is an all-convex tetrahedral vertex anywhere in the drawing, every Wbca junction may, but need not, be interpreted as all-convex tetrahedral). This cuts down the number of extended lines, but produces unacceptable results. Figure B.279 shows one example where the idea of only allowing non-trihedral underlying types implied by something visible fails; Figures B.189 and B.336 are others.

Limiting the final underlying types of vertices to those implied by the chosen labelling (i.e. if the chosen labelling can be satisfied assuming 3-hedral and 4-hedral vertex types, then 5-hedral and 6-hedral vertex types are not allowed in the topological completion) is safe, but does not necessarily reduce the number of extrapolated lines to a sensible level.

Another idea, incorporated in the current version of RIBALD, is to allow all K-type and Z-type underlying vertex types if the labelling of any vertex implies unambiguously any K-type or Z-type underlying vertex type, and none otherwise, and to allow all X-type and M-type underlying vertex types if the labelling of any vertex implies unambiguously any X-type or M-type underlying vertex type, and none otherwise. This appears at the time of writing to be the best compromise

Label	A	В	С	D
Occluding T	Y	Y	N	N
Tbda	Y	Y	N	N
Tbdc	Y	Y	Ν	N
Tdab	Y	Y	Ν	N
Tdac	Y	Y	Ν	Ν
Tbca	Y	Y	N	N
Tcab	Y	Y	N	N
Taba	N	N	Y	N
Tcca	N	N	Y	N
Tcda	N	N	Y	N
Tdda	N	N	Y	N
Tabb	N	N	N	Y
Tccb	N	N	N	Y
Tdcb	N	N	Ν	Y
Tddb	N	N	N	Y
Tcac	N	Y	Ν	N
Tbcc	Y	N	N	N

Table 10.5: Sensible Arcs for T-Junction Labels

Label	A	В	С	D	Ε	F
Zcbda	N	Y	Y	Y	Y	N

Table 10.6: Sensible Arcs for Z-Junction Labels

between flexibility and robustness.

Assuming that the number of bundles does not depend on the number of lines in the original drawing (not necessarily a good assumption), O(n) lines are extrapolated, and therefore $O(n^2)$ moves are generated per iteration, and the process takes $O(n^2)$ time per iteration.

10.6.2 Local Topology Hypotheses

Inspection of the test drawings shows that quadrilateral, usually rectangular, faces occur frequently in engineering practice. RIBALD generates moves which create or imply rectangular faces, in order to bias selection of moves towards rectangular construction. Two local configurations of vertices lead to useful hypotheses.

Firstly, given any two incomplete vertices A and B separated by a single complete

vertex C, adding a new vertex D to form a parallelogram ACBD is plausible. The figure of merit is based on a fixed value for this type of move (tuning constants T_h , T_i , T_t or T_u) multiplied a factor based on the proportion of quadrilateral faces in the partially-completed object and a fixed minimum value (tuning constant T_x) to encourage such moves even where no quadrilateral faces are present. By way of illustration, this hypothesis would generate the back face of Figure B.91 and the two back faces of Figure B.96.

If two incomplete vertices A and B are separated by two complete vertices C and E, adding an edge to join A and B to form the quadrilateral ACEB is plausible. The figure of merit is based on a fixed value for this type of move (tuning constants T_j , T_k , T_v or T_w) multiplied by a factor based on the product of the proportion of quadrilateral faces in the partially-completed object and the figure of merit for 3D parallelism of the supposedly parallel lines AC and BD, also with a fixed minimum value (tuning constant T_y). By way of illustration, this hypothesis would complete the front, partially-occluded face of Figure B.91.

In principle, the neighbourhood of each complete vertex can be examined in constant time to see if it matches the templates for these two move types, so this process could take O(n) time per iteration.

10.6.3 Occluding T-Junction Hypotheses

Hypotheses can be made for occluding T-junction completion, extending the methods used by Grimstead [38], who used both local and distant T-junction completion.

Local occluding T-junction completion is illustrated in Figure 10.12. A true vertex must exist somewhere further along the defining edge of the T-junction. It is plausible that this vertex is connected to the first non-occluding vertex reached by following the occluded region around along the occluding line. This would, for example, join vertices T to vertices A in Figures 10.2 (page 187) and 10.4 (page 187). The connection may be achieved by adding a single edge, as in the right-hand figure, or it may be achieved by adding a vertex and two edges, as in the left-hand figure. In either case, the figure of merit is the tuning constant T_f multiplied by the proportion of known faces in the object which would have the same number of sides as this method would predict.

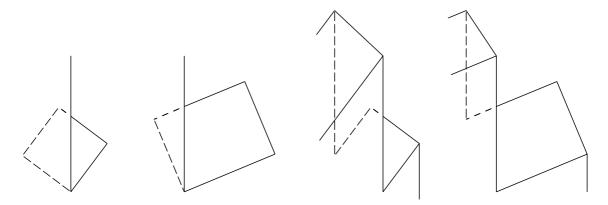


Figure 10.12: T Junctions—Local Alternatives

Figure 10.13: T Junctions—Distant Alternatives

For arrow occluding T-junctions (Tbaa as in Figure 10.4 or Tbab as in Figure 10.2), one pair of moves is generated, obtained by following the occluded region in the appropriate direction (clockwise or anticlockwise, respectively). For non-arrow occluding T-junctions (Tbac and Tbad), both such pairs are generated.

It is also worth generating moves from distant occluding T-junction completion—the hypothesis that for arrow T-junctions, the true vertex is connected to the next incomplete junction in the other direction around the object boundary (this would join vertices T to vertices B in Figures 10.2 and 10.4). This may or may not be true in practice, so has a separate tuning constant, T_g , for its figure of merit. Again, pairs of moves are generated, one which connects by adding a single edge, and another which connects by adding a vertex and two edges.

Generating the first sort of move takes O(n) time per iteration of generating, arbitrating and executing moves. Assuming that the size of the object boundary is O(n), generating the second sort of move takes $O(n^2)$ time per iteration.

10.6.4 Symmetry Hypotheses

Chapter 8 described production of a list of mirror chains, where each mirror chain tracks a potential plane of mirror symmetry across one or more visible faces. For each such chain which has been identified, an attempt is made to create a list of correspondences between the current partially-complete object and the mirror-image which would be generated by reflection. Each atom (vertex, edge or face) in the mirror chain is matched to its equivalent after the reflection operation, and then an attempt is made to propagate the matches through the existing part of the partially-completed object (RIBALD uses the propagation mechanism in [178]).

Atoms in the mirror-image which have no correspondence to atoms in the original are assumed to correspond to extra atoms which will be required, and the appropriate moves for creating them are generated (RIBALD generates moves for vertices and edges, and for quadrilateral faces where the loop of edges already exists).

Single atom hypotheses based on chains of mirrors are allocated figures of merit based on the figure of merit of the chain, as described in Chapter 8. This is multiplied by tuning constant T_l and divided by the number of new atoms required to create a complete pairing. If, when propagated, the pairings pair incompatible edge types, the merit is halved for each such incompatibility (the resulting topological hypotheses sometimes provide useful local information even though it cannot lead to a consistent global solution).

The provisional coordinates of newly-created vertices are determined from the coordinates of the unmatched vertices and the equation of the mirror plane (see [178]). As mirror chains are better at generating topology than geometry, the figure of merit for hypothesised vertex geometry is lower than that for the hypothesised topology (RIBALD uses the square of the topology figure of merit).

In addition to the single-atom hypotheses, RIBALD also generates a macro-move (Section 10.5.4) for each mirror chain for which no macro-move has already been accepted. The figure of merit for the macro-move is the figure of merit for the mirror chain multiplied by tuning constant T_m .

To prevent mirror hypotheses being re-made on every iteration, if there is something wrong with the pairing, the merit of the chain as well as that of the hypothesis is reduced.

Once a mirror macro-move is accepted, the hypothesis which generated it will not be used in generating either single-atom moves or macro-moves in subsequent iterations of topological reconstruction. In practice, the remaining topology required will usually be that which is implied but not required by the symmetry operation, and single-atom moves should be generated for these instead. RIBALD only makes one such inference: where the symmetry operation pairs two vertices which require at least one more edge, a move to create an edge linking them is generated (providing no such edge already exists). The figure of merit for this is the product of the merit of the symmetry hypothesis multiplied by tuning constant S_w .

Similarly, single-atom hypotheses derived from rotational symmetries are produced from pairings of equivalent vertices, edges and faces before and after the symmetry operation. In the case of rotations about a face centre, this process is seeded by noting that the face is its own equivalent, and storing the equivalent vertices and edges before and after the operation for each vertex and edge on the face; the axis of rotation is calculated by finding the best plane through the vertices on the face, and extending a normal to this plane from a point in the centre of the face. In the case of C_2 rotations about an edge mid-point, the equivalence process is seeded by noting that the edge is its own equivalent, the vertices at either end of the edge are equivalent to one another, and the faces adjacent to the edge are equivalent to one another; the axis of rotation is as close as possible to the sum of the face normals of the adjacent faces while being constrained to be perpendicular to the edge direction. In the case of rotations about vertices, the equivalence process is seeded by noting that the vertex is its own equivalent, and storing the equivalence relations of the edges and faces adjacent to the vertex; the axis of rotation is the sum of the face normals of the adjacent faces.

Equivalence is propagated through the object in a similar way to mirror pairing propagation as described above, the only difference being the size of the appropriate symmetry group cycle (e.g. for C_3 , if $A_1 \mapsto A_2$ and $A_2 \mapsto A_3$ then $A_3 \mapsto A_1$).

The base figure of merit for a hypothesis based on rotational symmetry is the figure of merit for the symmetry element. As with mirror planes, this is multiplied by tuning constant T_l , divided by the number of moves generated by the symmetry element, and halved whenever the pairings pair incompatible line types,

Rotations about edge centres are included for completeness, and it may be better to omit reasoning based on them from a practical system: there are few if any drawings for which these give any additional topological information (for example, the two hidden faces of Figure 10.14 which can be deduced from edge-centred rotation hypotheses can also be deduced by other means), and the calculation which produces geometric location estimates for hidden vertices is relatively slow and comparatively inaccurate.

If pairing of vertices and edges is complete, and all vertices have at least the minimum number of edges required, it is plausible that the vertex/edge framework is already complete. In this case, RIBALD reinforces the completeness hypothesis

by the symmetry operation figure of merit.

There are O(n) symmetry elements, and producing a pairing takes $O(n^4)$ time, so generating symmetry moves takes $O(n^5)$ time per iteration.

10.6.5 Feature Hypotheses

Each feature hypothesis from Chapter 6 which remains valid generates a corresponding feature macro-move (feature hypotheses become invalid if the topology of any vertex in the feature template has been changed by a previously-accepted move). The figure of merit of the move is the figure of merit of the feature.

RIBALD does not generate new-vertex, new-edge or new-face moves which would produce part of the feature—this is partly because time to implement this idea was not available, and partly because such moves would reinforce identical moves generated by other hypotheses and thus always be chosen in preference to the feature macro-move.

10.6.6 Classification Hypotheses

Early versions of RIBALD generated moves on the basis of classification identified by the methods of Chapter 9, in order that when a drawing qualified for two or more classes, moves suggested by both classes would reinforce one another. Identification of compatible and incompatible classes, as described in Chapter 9.4, supersedes this idea.

10.6.7 Nearly-Complete Hypotheses

When it can be recognised that the problem of constructing a topology is near a solution, moves which lead towards the solution should be preferred to those which lead away from it. To this end, moves are also generated by analogy with some of the simpler endgame positions described in Section 10.3.

If an object contains no necessarily incomplete vertices and no T junctions, it is reasonable to conclude that all vertices and edges have been found. In these circumstances, the base merit for the "already complete" move is $1/n_F$, where n_F is the number of additional faces (as predicted by Euler's formula) required to complete

the topology (N.B. if there are no additional faces required, no more topology can be added and the object is automatically complete). This is multiplied by the figures of merit for completeness of each vertex.

In any object which contains exactly two necessarily incomplete vertices and these vertices are not connected by an existing edge, it is reasonable to add a single hidden edge connecting them to complete the vertex/edge framework. It is not always a good hypothesis for non-trihedral objects, as Figure 10.15 shows. This move, if generated, has a fixed figure of merit T_a .

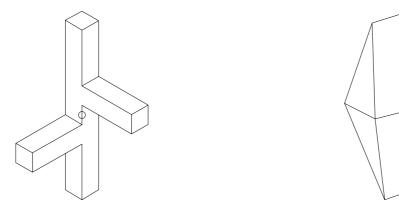


Figure 10.14: Edge-Centred Rotation

Figure 10.15: A Good Move?

The same deduction can be made in any object which contains exactly one necessarily incomplete vertex and exactly one T junction, and the vertex is located approximately on a continuation of the edge defining the T junction. It is reasonable to extend this edge to the biconnected junction to complete the vertex/edge framework. The figure of merit is the product of T_b and the figure of merit for parallelism between the original edge vector and the new edge vector.

If there are exactly three necessarily incomplete vertices remaining, it can be hypothesised that there exists a single trihedral hidden vertex connected to all three. The figure of merit is a tuning constant, T_c . The coordinates of the new vertex are the mean of the coordinates of the closest points of approach of pairs of vectors extended along unused bundles from the three incomplete vertices.

Finally, if there are exactly four necessarily incomplete vertices remaining, sometimes the best way of completing the vertex/edge framework is to add two new edges to join these vertices in pairs; unless edges already exist joining these vertices, there will be three ways of doing this. All six new-edge moves are generated, with a fixed figure of merit given by the tuning constant T_d . These moves have no affect on the order of the overall algorithm as the situations which cause them to be generated occur once only.

10.7 Hypothesis Adjudication

Since each move has an associated figure of merit, selecting a move should simply be a matter of choosing the one with the highest figure of merit (the more complex ideas suggested in [173] have been discarded). It is possible that the generating mechanisms favour one or other type of move disproportionately—the inclusion of various tuning constants is intended to overcome this. There are also general considerations which apply to all moves, irrespective of the hypotheses which generated them, which must be evaluated before selecting the best move. Some moves can be rejected outright; other moves may have their merit reduced because their consequences contradict beliefs about the object; and moves which appear to represent the same additional topology can be combined.

10.7.1 Rejected Moves

Hypothesis adjudication should store the hypotheses and their figures of merit in order that if the top recommendation is rejected, either immediately or later, the next recommendation can be tried instead.

RIBALD rejects a move if, while the object remains incomplete, the resulting tree of subsequent moves is empty. It also rejects a move if, after making the move, the object becomes invalid (the object has exactly two vertices to which further edges are to be added, and these vertices are already linked by an edge) or too complex (the object remains incomplete, but there are already more hidden faces than visible faces—this prunes out long tree-searches which can occur when a poor choice of initial move results in more and more detail added to the back of the object in an attempt to produce a valid object).

10.7.2 Undesirable Moves

RIBALD reduces move figures of merit if the hypothesised move conflicts with beliefs about the object as a whole, or for other reasons which reduce their plausibility.

As described above, whenever a new hidden vertex is hypothesised, a prediction is made of its geometric location, and this geometric prediction has its own figure of merit F_G .

Any hypothesis which places a new vertex in a location in which it would be visible given the existing faces should have its figures of merit reduced—it is probably invalid (if the frontal geometry estimates were perfect, this would be certain). RIBALD reduces the merit of both the move and the hypothesised location $(F'_T = T_p F_T; F'_G = T_p^2 F_G)$ when it detects this case. This is intended to allow for the fact that the geometry of the partial object is at best provisional, and thus locations derived from it are inaccurate, without rejecting better hypotheses which place the new vertex in a hidden location. For reasons of speed, the test is implemented by comparing the hypothesised location with the minimum and maximum x- and y-coordinates and minimum z-coordinate of the object, not as a (higher-order and slower) test that a visible face can be found which would occlude the new vertex. RIBALD should, but does not, make this adjustment for macro-moves (completion from a mirror chain, or completion of a feature) as well as for discrete additions to the topology.

Similarly, on the basis that most objects are drawn as if they were resting on a flat plane, any move which hypothesises a vertex "underneath" this base plane should have its merit reduced, as above. RIBALD should, but does not, implement this.

Moves which would generate edges which are clearly out of place in the object by virtue of being unusually short or unusually long have their merit reduced. In addition, it is reasonable that shorter edges are generally to be preferred to longer edges, as adding a long edge is a more drastic change to the topology (this seems to work reasonably well in general, although it aggravates the problem noted below concerning Figure B.71). New-edge moves are adjusted as follows (T_s and T_r are tuning constants):

- If the length L_E of a new edge is shorter than the length L_S of the shortest visible edge, multiply by $\left(\frac{L_E}{L_S}\right)^{T_S}$
- If the length L_E of a new edge is longer than the length L_L of the longest visible edge, multiply by $T_r \left(\frac{L_L}{L_E}\right)^{T_s}$

\bullet Otherwise, multiply by $\frac{L_E(1-T_r)+L_ST_r-L_L}{L_S-L_L}$

Moves which generate two new edges (and a new vertex) have their merit multiplied by the geometric mean³ of these factors for the two edges. As with adjustments for vertex coordinates, RIBALD should, but does not, multiply macro-move merit figures by edge length adjustments.

If a hypothesised edge (either on its own or part of a vertex+edges move) passes suspiciously close to an existing vertex (particularly an incomplete existing vertex), it is likely that the hypothesis is wrong and the edge should terminate at the nearby vertex. To illustrate the concept, consider Figure B.448 after completion of the underslot: an edge connecting the far bottom corners is a valid addition, but two shorter edges connecting the far bottom corners to the far ends of the underslot is preferable.

RIBALD includes a test for this: when it detects that a hypothesised edge AB (with merit F_E) passes close to an incomplete vertex V, it generates moves to create edges AV and BV (both with merit T_nF_E) and lowers the merit of the move generating AB to $(1-T_n)F_E$. This test should, but does not, include a merit adjustment based on exactly how close the longer edge gets to the incomplete vertices. Care has to be taken to avoid infinite recursion, which can happen when a third vertex is close to one of the vertices linked by the hypothesised edge. For example, new edge AB is hypothesised, and B is close to C, AC is hypothesised instead, but since C is close to B, ...

It was noted in Chapter 2.8 that the presence of more than one subgraph in a drawing sometimes (but not always) indicates the presence of a hole loop feature in the object. For this reason, hypotheses which would create edges which join vertices from different subgraphs should be discouraged but not forbidden. RIBALD multiplies the figure of merit for any such move by tuning constant S_y ; in addition, where the subgraphs have been identified as of different types (e.g. one is a pocket, and the other is a boss) the figure of merit is further multiplied by tuning constant S_z .

In an early version of RIBALD [173] which only processed trihedral drawings, if the figure of merit for the complete object being a normalon was non-zero, any move

³If either edge factor is small, the combined factor should also be small.

which clearly implies an odd-sided face was downgraded ($F'_T = F_T(1 - F_{normalon})$). Although this test could in principle be adapted to the non-trihedral domain, it is less straightforward (consider, for example, the "pentagonal" faces of Figures B.159 and B.265), and it has not been retained.

It is always possible to complete topology by adding triangles to the hidden part of the object, but it is usually a bad idea to do this (Figure B.365 is a rare counter-example). The merit of any new-edge or new-vertex hypothesis which introduces a triangular loop of edges when the frontal geometry contained no complete triangular faces is multiplied by T_o , a tuning constant.

In the earlier, trihedral version of RIBALD [173], when two or more hypotheses generate moves with the same connectivity but with incompatible edge types (one convex, the other concave), choice between them was deferred until a later iteration (by which time one or other of the hypotheses may have been abandoned) providing there were other reasonable moves. This has not been retained—the non-trihedral interpretations of junction labels such as Wbca are ambiguous and likely to remain so however much else of the object is reconstructed, and deferring the best hypothesis until later is likely to do more harm than good. It could be argued that this idea should nevertheless apply to other cases where there are two or more similar hypotheses and choice between them can sensibly be deferred while there is other reconstruction work to do. RIBALD does not include any such tests.

10.7.3 Combining Moves

Where two or more hypotheses suggest the same move, the moves are merged and the figure of merit reinforced. Whenever equivalent moves are merged, the predicted locations are merged by calculating a weighted mean location, the weights being the geometric figures of merit.

RIBALD should, but currently does not, increase figures of merit where different moves suggest different but compatible hypotheses which would create vertices in more or less the same place, e.g. one move generates a vertex V and edges connecting it to A and B, and another move generates a vertex in the same place and edges connecting it to A and C. Adding this would be straightforward, but optimising the necessary tuning constants would be time-consuming.

RIBALD should, but does not, reinforce figures of merit where entirely different move types would have the same consequences. An example of this would be a new vertex move (Section 10.5.1) which, incidentally, also creates a quadrilateral face, and a quadrilateral new face move (Section 10.5.3) which includes three existing vertices in the loop.

10.8 Special-Case Recovery of Topology of Hidden Parts

Reconstruction of the topology of special classes of objects is important for two reasons. Firstly, quick and robust methods exist for some classes (particularly extrusions), improving average performance even if general-case methods are used for irregular objects. Secondly, it may be the case (Section 10.9 appears to indicate this) that no wholly-reliable general-case mechanism exists; if this is so, attempting to decompose objects (as outlined in Chapter 2.16) until a successful classification can be made of each piece provides an alternative route for attacking the general reconstruction problem; there has not been time to investigate this idea.

For comparison purposes, RIBALD includes an option to force use of the generalcase mechanism for classes where it would not normally be used.

10.8.1 Right Extrusions and Frusta

The topology of an extrusion or frustum is easily reconstructed by creating a back end cap with the same topology as a mirror-image of the front end cap, and joining equivalent vertices in the two end caps by side edges and faces. There are benefits both in terms of speed (special-case topological recovery for extrusions is very quick, and extrusions are common) and in reliability (the general-case hypothesis mechanism is given no chance to make mistakes).

For simplicity, RIBALD processes as an extrusion any drawing which has been classified as an extrusion and as *also* belonging to some other compatible class, including prisms, cubes and other axis-aligned extrusions. RIBALD also uses this mechanism for extrusions with through holes in the direction of extrusion (topological completion of extrusions with side-to-side holes or pockets is more complex and

RIBALD uses the general-case mechanism for these). In addition, it is a reasonable assumption that drawings which might be extrusions or frusta but are not classified as such, such as Figures B.514 and B.111, are topologically equivalent to extrusions and frusta; RIBALD completes the topology as if the object were an extrusion and then adjusts vertex x- and y-coordinates to match the drawing.

10.8.2 Normalons

RIBALD uses the general-case mechanism already described for normalons. Before doing so, it limits the underlying vertex types of all vertices to the set of types found in normalons: the trihedral types plus Zcdcd and Zcdcdcd (note that although Yabd junctions cannot be Zcdcd vertices in normalons, they can be Kcdcd vertices in nonnormalons). This limitation, plus the fact that there are only three edge bundles in normalons, significantly reduces the number of extrapolated lines. Section 10.9 shows the resulting improvement in speed and reliability.

10.8.3 Single Symmetry Dominates

RIBALD uses the general-case mechanism to complete the vertex/edge framework of drawings classified as semi-axis-aligned with mirror plane, since the general-case mechanism already includes a macro-move for reconstructing topology from a mirror plane (this will usually be the highest-merit move on the first iteration). Since the resulting framework will often not be complete (for example, constructing topology from a mirror plane will not create an edge bisected by the mirror plane), some general-case processing is in any case required for most such objects.

10.8.4 Platonic and Archimedean Solids

Only the simplest of the Platonic and Archimedean solids can be reconstructed using the general mechanism: the tetrahedron (Figures B.114 and B.115), octahedron (Figure B.117), truncated tetrahedron (Figure B.125) and truncated cube (Figures B.121 and B.122). Failures do not necessarily indicate a deficiency in the ideas in this chapter—it has already been noted that the Platonic and Archimedean solids are those which most often defeat parallel line bundling (Chapter 5) and inflation (Chapter 7).

Even if the topology of Platonic and Archimedean solids can be completed using the general case ideas, there is no benefit in doing this if (as is usually the case) the geometry must then be reconstructed as a special case. Thus, in a practical system, both the topology and the geometry of a Platonic or Archimedean solid should be read from a database of the known finite set of such solids.

Note, however, that the regular solids make useful test cases to determine whether RIBALD handles objects with multiple high-merit symmetries correctly.

10.8.5 Multiple Symmetries

The only test drawings which meet the criteria of multiple high-merit symmetries are the Platonic and Archimedean solids and highly-symmetric axis-aligned solids. RIBALD handles these as described above.

10.9 Results

Analysing the methods outlined in this chapter, it would appear that topological reconstruction always terminates, but may be very slow, particularly if backtracking occurs. It may terminate because it has produced a complete framework, or because it has no valid way of doing so. Even if it has produced a complete framework, there is no guarantee that it is the one the user wanted. The test results in this section therefore consider the problems of time, and of predictability and reliability.

The test cases analysed were all test drawings for which any labelling method analysed in Chapter 4 produced the correct output and for which a valid polyhedral interpretation exists (i.e. drawings such as Figures B.149 and B.146 are excluded, but Figure B.147 is included as it could be a non-normalon polyhedron). Note that the test cases specifically include figures with bosses such as Figure B.429—although it is already known that RIBALD will not interpret these correctly (there has not been time to implement the ideas of Chapter 2.16) it remains to be demonstrated that a valid (albeit necessarily incorrect) interpretation of such drawings will be produced in a reasonable time. Where any bundling option (see Chapter 5) produced correct results, that option was used; otherwise, the default option was used. The default option for inflation (see Chapter 7) was used in all cases.

It was found that the topology completion process terminates for all test cases (however, if the general-case mechanism is used for the Archimedean solids in Figures B.132 and B.126 or the extrusion in Figure B.538, it does not terminate—this is probably an implementation problem). In most cases, it terminates in interactive time—see Section 10.9.3. In many cases, RIBALD produces the expected output; in most cases, RIBALD produces topologically valid solids (in a few cases, these solids are not geometrically realisable in 3D); in some cases, RIBALD found no interpretation or was unable to make progress. Reliability is analysed in Section 10.9.1.

10.9.1 Predictability and Reliability

Three factors must be considered in analysing predictability and reliability: whether or not the process produces the desired output, whether or not the process produces any acceptable (topologically-valid) output, and how sensitive these results are to variations in the process.

Where RIBALD recognises a drawing as an extrusion, it always reconstructs the topology correctly (RIBALD classifies all but one of the 96 drawings of extrusions correctly; the exception is Figure B.420 which is not classified as an extrusion because RIBALD is unsure whether the central hole loop is a hole or a boss). If forced to use the general-case mechanism, 67 extrusions are reconstructed correctly; RIBALD produces incorrect but geometrically-valid interpretations of 13 other drawings, and fails to produce any interpretation for 16 drawings.

RIBALD reconstructs all 6 drawings of frusta tested correctly. If forced to use the general-case mechanism, 5 are reconstructed correctly; the exception is Figure B.141 (see next section).

RIBALD reconstructs all 19 drawings of Platonic and Archimedean solids correctly. If forced to use the general-case mechanism, 6 are reconstructed correctly; RIBALD produces unexpected but topologically valid interpretations of 8 other drawings, and fails to produce any interpretation for 5 (including Figures B.132 and B.126).

Using the ideas in this chapter, 64 of the 80 drawings of non-extrusion normalons can be interpreted correctly, although in many cases this requires hand-chosen values of tuning constants. With the best fixed set of tuning constant values so far

(those listed in Appendix C), RIBALD reconstructs 41 correctly, produces valid but unexpected interpretations of another 7 (including Figure B.74), and fails to produce any interpretation of 32 drawings.

Using the ideas in this chapter, 211 of the 271 remaining drawings have been interpreted correctly; again, doing so sometimes required hand-chosen values of tuning constants. With the tuning constant values listed in Appendix C, RIBALD reconstructs 102 correctly, produces valid but unexpected interpretations of another 85, and fails to produce any interpretation of 84 drawings.

10.9.2 Problems Encountered

In testing complex drawings, it was found that most of the problem cases—failure to find a valid framework, finding a valid but implausible framework, and taking too long—resulted from a few causes.

The most common, and most serious, classifiable error results from the idea of reconstructing the vertex/edge framework without filling in face loops. It was seen in Chapter 5.5 that the direction of turn at a corner on a face, the convexity/concavity of edges leaving the face at that corner, and the direction (above or below) in which they leave the plane of the face are related. Where the face loop has not been completed, the plane of the face is unknown, so it is not possible to reject hypotheses which could, if the plane of the face were known, be rejected as absurd. However, since in many cases only the vertex/edge framework is known, and filling in face loops has been deferred, a crucial datum is missing and RIBALD cannot deduce that the move should be rejected.

For example, when hypothesising a convex edge joining two vertices which are placed at convex turns on existing face loops, it is known that the edge must be below both face planes. If, geometrically, the edge is above one or both face planes, the hypothesis is absurd and should be rejected. Consider Figures 10.16 and 10.17. If it is known that lines A, B and C are convex, the edges to be added at the incomplete vertices at the ends of lines A and B must leave those vertices below the planes of the faces, so adding edge D is wrong. (If, however, lines A, B and C were concave, adding a convex edge D would be valid and probably correct.)

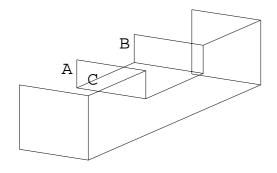


Figure 10.16: Before

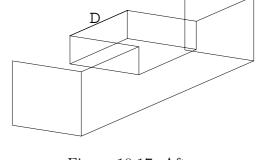


Figure 10.17: After

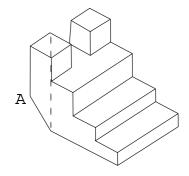


Figure 10.18: Isolation Error

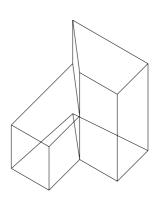


Figure 10.19: No Polyhedral Geometry

A move adding a single edge should not be accepted if the edge splits the vertex/edge graph of the object such that one half contains a single incomplete vertex—see Figure 10.18, where vertex A would be isolated if the edge indicated by the dotted line were to be added. Detecting problems of this sort is not straightforward, and a literature search [33, 37, 106] did not find a known method for this.

With some drawings, lines are extrapolated from incomplete vertices but they do not cross, so no moves are generated. Consider, for example, Figure B.407. After accepting the obvious initial moves, completing the quadrilateral base and adding an edge to join it to the topmost vertex, there remain four extrapolated lines, all parallel. Obviously, where there are four parallel concave lines, it would be a reasonable hypothesis to terminate them all by adding a quadrilateral face normal to the lines to form a pocket, and introducing such a hypothesis into RIBALD would not be difficult. It would, however, only remove the problem in this particular case—the 14 other drawings where a similar problem occurs would require other solutions. The most extreme case is Figure B.89, where, after completing the obviously cuboidal end-pieces, there remain seven extrapolated lines (two groups of three parallel lines

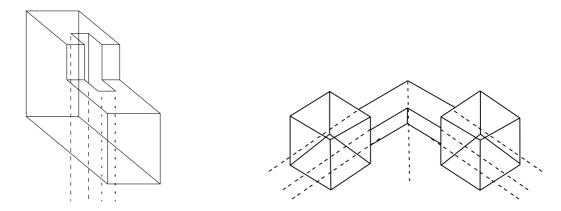


Figure 10.20: Extrapolated Lines do not Cross

and a vertical line down from the central vertex), none of which cross—see the right-hand side of Figure 10.20.

At times, RIBALD can produce a topology which has no valid non-self-intersecting geometric realisation as a polyhedron. One such cases is illustrated in Figure 10.19 (deriving from Figure B.336). This is still under investigation.

One common fault with earlier versions of RIBALD which still appears with some objects is that it splits perfectly good quadrilateral faces into triangles by adding an edge joining opposite corners, when ambiguous underlying vertex types permit this. The "framework is complete" hypothesis alleviates this problem, as does the avoidance of triangular loops when none are visible in the drawing. It may be that a finer balance between tuning constants is required, but more often, the extraneous edge is not added last, so the balance between completeness and adding an extra edge is not tested.

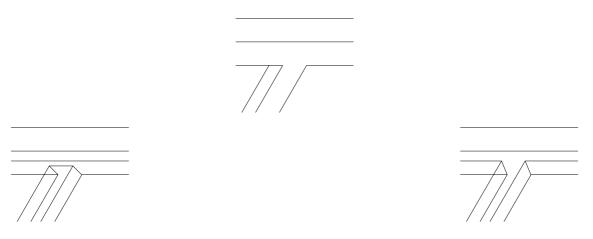


Figure 10.21: Feature and Interpretations

Less seriously, RIBALD frequently interprets the local feature shown in Figure 10.21 (top) incorrectly, as shown on the left, rather than as shown on the right. Depending on the values of tuning constants used, this occurs frequently when completing Figures B.71 and B.74, and a similar problem sometimes occurs when completing Figure B.69. The problem is not so much that the resulting object will be incorrect (the geometry will be almost right, and capable of being "healed" [10]), as that symmetry and regularity artefacts are lost by accepting the poor hypothesis, thus increasing the time taken to find the best completion and the likelihood that a poor completion will be chosen instead.

10.9.3 Timings

The time taken (in seconds) for the general-case topological completion process to terminate for normalons is shown in Table 10.7^4 .

No. of Lines	Minimum	Median	Maximum
9–16	0.02	0.04	0.08
17–24	0.00	0.04	0.81
25–32	0.04	0.30	0.58
33–40	0.32	1.26	2.50
41–48	0.37	1.69	1.70
49–56	_	1.81	_
57+	3.54	_	12.87

Table 10.7: Normalons: Summary of Average Timings (seconds)

Where valid output is produced, timings for normalons are in general satisfactory—in the extreme case, Figure B.74, RIBALD takes 12.87 seconds to produce a valid normalon (albeit not the one expected), but the only other drawing for which RIBALD takes more than a second to produce valid output is Figure B.554 (1.81 seconds), which it interprets correctly. In general, RIBALD takes longer when it fails, because of the backtracking involved in searching for a valid solution.

The time taken (in seconds) for the general-case topological completion process to terminate for non-normalons is shown in Table 10.8.

⁴Where there is only one drawing in a group, no minimum or maximum timings are shown; where there are only two drawings in a group, no median timing is shown.

No. of Lines	Minimum	Median	Maximum
1–8	0.00	0.01	0.48
9–16	0.00	0.04	0.50
17–24	0.00	0.07	5.30
25–32	0.08	1.00	14.20
33–40	0.51	1.35	3.99
41–48	0.63	2.36	6.09
49–56	_	-	_
57+	_	13.47	_

Table 10.8: Non-normalons: Summary of Average Timings (seconds)

Similar, but even more pronounced, differences are observed with non-normalons. Again, it is found that RIBALD takes longer when it fails than when it reconstructs the desired object. In only one case where it produces correct results does RIBALD takes longer than a second: 1.66 seconds for Figure B.509 (and in only one other case more than half a second, 0.66 seconds for Figure B.466). Where RIBALD produces valid but unexpected output, it takes longer—5.32 seconds in the case of Figure B.451, and 1.94 seconds in the case of Figure B.488. Where RIBALD produces no valid interpretation of the drawing, it takes even longer—it takes more than a second before admitting defeat for 21 non-normalon drawings, taking over 13 seconds on Figure B.147 and over 4 seconds on Figures B.513, B.469 and B.459.

About three-quarters of the drawings of extrusions can be reconstructed correctly both by special-case methods and by the general-case method. Timings for some of these drawings are shown in Table 10.9.

Only the simplest two regular solids, Figures B.114 and B.115, are reconstructed correctly using the optimal tuning constants. The rest lead to valid but irregular polyhedra (except when one of RIBALD's internal limits, maximum number of vertices = 120, is exceeded). Comparative timings would be meaningless.

10.9.4 Summary and Recommendations

Without backtracking, topological reconstruction takes $O(n^6)$ time, the rate-determining step being pairing propagation (the algorithm described in Chapter 8.3 is also used here), performed once for each symmetry element for each iteration of

Drawing	Special-Case	General-Case
Figure B.29	0.00	0.00
Figure B.58	0.00	0.00
Figure B.37	0.00	0.00
Figure B.60	0.00	0.00
Figure B.54	0.00	0.01
Figure B.545	0.00	0.01
Figure B.51	0.00	0.02
Figure B.39	0.00	0.02
Figure B.529	0.00	0.02
Figure B.470	0.01	0.02
Figure B.36	0.00	0.03
Figure B.43	0.00	0.05
Figure B.42	0.00	0.05
Figure B.543	0.00	0.05
Figure B.25	0.00	0.06
Figure B.504	0.00	0.07
Figure B.551	0.00	0.09
Figure B.525	0.00	0.13
Figure B.506	0.00	0.16
Figure B.46	0.01	0.20
Figure B.61	0.01	0.28
Figure B.38	0.01	0.40

Table 10.9: Comparison of Special and General-Case Timings (seconds)

generating, arbitrating and executing moves. A lower-order algorithm for this would improve matters. An incremental pairing mechanism, storing the results of pairing propagation from previous iterations and adding any new topology to them, would reduce the time taken to $O(n^5)$ at the cost of considerable additional complexity of implementation. If an incorrect choice is made and backtracking invoked, the process takes exponential-order time.

In practice, topological reconstruction is usually fast enough when it works, but slow when it fails. It is notable that, although the ideas in this chapter have been used to reconstruct about 80% of the test drawings successfully, the best *fixed* set of tuning constants (determined as described in Appendix C) reconstructs correctly only about 50% of normalons and 40% of non-normalons. In order to improve on this, it is necessary either to be able to determine from the drawing which heuristics are most likely to be successful (adjusting the tuning constants accordingly) or to

introduce additional means of ruling out bad choices of move.

The values of this best fixed set of tuning constants allow some analysis of which ideas are most useful in topological reconstruction. It can, for example, be noted that adding a vertex and two edges to complete a quadrilateral face, and adding a single edge to complete a quadrilateral face, are both very successful hypotheses for non-normalons but less convincing for normalons—this apparently counter-intuitive result can be explained since the principal alternative, extrapolating lines and noting intersections, is more reliable for normalons but less reliable for non-normalons. Similarly, creating faces from quadrilateral loops of edges is a reliable method, producing benefits in terms of reliability to counteract the losses in terms of speed.

Another reliable method is that of adding a vertex and three edges to complete the object when only three necessarily incomplete vertices remain. This, although not infallible for non-trihedral objects, remains an effective move. Other methods based on the idea that the object is nearly complete are less reliable.

It can be noted that the base figure of merit for discrete hypotheses based on mirror chains is significantly higher than the base figure of merit for the mirror macro hypothesis—piecewise addition of topology from deductions based on mirror chains is noticeably more robust (albeit slower) than adding as much as possible as soon as possible.

The relatively low merit figure for local occluding T-junction completion suggests that this is not especially useful—in general, line extrapolation makes the same suggestions about topology to be added—and the even lower merit figure for distant occluding T-junction completion suggests that this idea could safely be omitted in most cases.

At the moment, RIBALD does not include CSG-style concepts such as half-spaces. Some of the common problems noted above, particularly those caused by lack of knowledge of the local neighbourhood of a vertex, could be removed by regarding face planes as half-spaces, one solid and one empty, and edges as half-space operators (convex edges as intersection and concave edges as union). By this means, it could be made clear whether edge locations in relation to face planes were sensible.

It is plausible that introduction of CSG-style half-spaces would provide a solution to the problems illustrated in Figure 10.20—obviously, in the case of Figure B.407,

whatever happens to the four extrapolated lines, it happens when (or before) they cross the plane of the base face. Whether this would remove similar problems with other drawings which show the same symptoms requires investigation.

The solution to the problem of producing a topology with no polyhedral geometric interpretation, as illustrated in Figure 10.19, is not obvious. It is not sufficient to identify cases where the provisional geometry is incorrect—with some test drawings, topological reconstruction initially produces incorrect geometry (in particular, intersecting faces) but the following geometric finishing stage is able to correct this. Neither is simple enforcement of Euler's formula a solution. Topologies which fail to meet this criterion are rejected. However, this by itself does not guarantee that it is possible for the realised solid object to have convex and concave edges where these have been identified, or for all faces to be made planar simultaneously.

Ideally, a topological reasoning method must identify and discard those topologies which cannot be realised using planar geometry with appropriate convex and concave edges. In the absence of this, a substitute approach would be to produce a figure of merit for the topological completion. This could be based on geometrical considerations such as self-intersecting geometry as well as topological considerations such as how well the constructed topology matches predicted symmetry and regularity artefacts. If the figure is below a threshold value, the completion is stored but one or more alternative topologies are sought, and the one with the highest merit is the one passed on to the geometric finishing stage. This idea has not been tested.

The process of filling in face loops (Chapter 2.14) sometimes fails, reporting that there is no loop of unused edges which returns from the end of an unused edge to its start. This is still under investigation (it happens for Figure B.303), but is believed to be a fault with the output of topological reconstruction rather than an omission in the algorithm for detecting face loops. The solution seems to be to backtrack to the frontal geometry stage and set the merit of the most likely class to zero. At the moment, RIBALD can backtrack within topological reconstruction, or from one stage of processing to another, but not from a subsequent stage of processing to a point within topological reconstruction.

Chapter 11

Geometric Finishing

11.1 Introduction

This stage of processing takes a topologically-correct object and a group of symmetry and regularity hypotheses, turns these into constraints, and aims to produce geometric information which satisfies as many of the plausible constraints as possible. Specifically, it seeks the set of face equations and vertex coordinates which "achieves as much merit" as possible (see Appendix D). The output of this stage is a complete list of geometric data for vertices, edges and faces which together with the previous topological information determines a boundary representation solid model.

In order to make the problem more tractable, it can be subdivided into determination of face normals and determination of face distances from the origin (once face equations are known, vertex coordinates may be determined by intersection). Both RIBALD and Kumar and Yu [75] subdivide the face equation problem in this way, for the same reason—changing face normals can destroy satisfied distance constraints, but changing face distances cannot destroy satisfied angular constraints.

There are theoretical doubts about this subdivision, related to the *resolvable* representation problem described in Section 11.2.2. It is known that (a) determining face normals first, and then face distances, achieves a resolvable representation for many objects, including all trihedral polyhedra; (b) there are objects which have resolvable representations, but for which determining face normals first, and then face distances, does not achieve a resolvable representation; and (c) there are objects for which no first-order resolvable representation exists. The frequency of occurrence

of objects in the second category has yet to be determined; if it is low, the sequential method used here and by Kumar and Yu [75] can be justified.

The alternative of processing angular and distance constraints simultaneously by methods similar to those described in this chapter can be discarded as too slow for interactive use—briefly, face distance constraints are numerous, and face normal constraints slow to process, so processing them together is impractical. Instead, RIBALD includes a geometric error-detection postprocessing stage which looks for non-trihedral vertices which do not lie on all of their adjacent faces, and adjusts one of the faces to fit. This is not theoretically satisfactory, but it suffices for practical purposes.

The overall algorithm is:

- Make initial estimates of face normals
- Use object classification from Chapter 9 (if any) to restrict face normals
- Identify constraints on face normals
- Adjust face normals to match constraints
- Make initial estimates of face distances
- Identify constraints on face distances
- Adjust face distances to match constraints
- Obtain vertex locations by intersecting three face planes for each
- Detect vertex/face failures and adjust faces to correct them

Section 11.2 describes previous and ongoing work in this area. Section 11.3 lists the types of constraints which RIBALD attempts to enforce. Section 11.4 describes a simple downhill optimisation method for determining face normals. Section 11.5 describes improvements to this method which take account of geometric knowledge—this represents the current state of the art, and is the method implemented in RIBALD. Section 11.6 describes an alternative iterative optimisation method, using geometry to predict the updated face normals; when implemented in practice, it performs better than the simple idea in Section 11.4 but not as well as

the improved version in Section 11.5. Section 11.7 describes an attempt to use a genetic algorithm for the face normal problem; the results were not encouraging. Section 11.8 describes a simple downhill optimisation method for face distances which produces satisfactory results for trihedral objects. Section 11.9 describes ideas for refining this method to allow for geometric distance constraints and non-trihedrality; there has not been time to incorporate a complete implementation of these ideas in RIBALD. Section 11.10 describes how RIBALD obtains vertex coordinates from face equations. Section 11.11 describes how some of the more time-consuming ideas in this chapter can be bypassed for objects which fall into one or more of the special classes described in Chapter 9. Section 11.12 shows some results of geometric fitting.

Some, but not all, of Section 11.3, is new. All of the work in Sections 11.4–11.12 was believed new at the time. It has since emerged that Ge et at [32] were working on ideas similar to those in Section 11.4, and Kumar and Yu [75] on ideas similar to those in Sections 11.4 and 11.8, concurrently—precedence is not clear, and (particularly as the simple ideas in these two Sections do not constitute an adequate solution to the problem addressed by this Chapter) no strong claims are made for the novelty or otherwise of these two Sections. As geometric beautification is an active area of current research, it is likely that there is other similar work in progress which has not yet appeared in the literature.

11.2 History

Beautification of solid models is an important area of current research with a history of its own. Alongside (and often independently of) this, detection and enforcement of geometric constraints has received considerable attention. Finally, general work on constraint enforcement (independent of 3D geometry) is also of relevance. These topics are considered separately, with the most general discussed first.

11.2.1 General Constraints

A constraint is a relationship between variables expressed as a function (an equation or inequality) of those variables [80]. A continuous constraint satisfaction problem (continuous CSP) attempts to find values for the variables which provide a solution

to a system of constraints. Such a problem may be well-constrained, over-constrained or under-constrained [81].

In a well-constrained problem, there are exactly as many constraints as are required in order to find a solution. For example, the equation x = 1 is a (trivial) well-constrained problem. So, it may be noted, is the equation $x^2 = 1$ (assuming real x)—a well-constrained problem does not guarantee a unique solution. What distinguishes a well-constrained problem is that there are exactly as many equations as are required to reduce the number of degrees of freedom in the system to zero—in the case of the equation $x^2 = 1$, there are zero degrees of freedom, as there is no path away from either solution along which the constraint remains satisfied.

In an over-constrained problem, there are more constraints than are required to find a solution. Where there are redundant constraints, there may still be a solution (for example, the system of constraints x = 1; y = 1; x = y has more equations than unknowns but has a unique solution). Where there are incompatible constraints, there is no solution (for example, x = 1; y = 2; x = y has no solution)—at least one of the constraints must be removed before a solution can be found.

In an under-constrained problem, there are not enough constraints to remove all degrees of freedom. For example, there is one degree of freedom in the system of constraints $x^2 + y^2 = 1$; x may be changed continuously as long as y changes to follow suit.

Comparing the number of constraints and the number of variables does not provide a method of determining whether or not a system is over- or under-constrained, since it is possible for a system to be both [81]. For example, the system $x^2 + y^2 = 1$; z = 1; z = 2 is both over- and under-constrained.

In a valued constraint satisfaction problem (VCSP) [144], a numerical value is assigned to each constraint. VCSPs are usually over-constrained, and the numerical values, which are typically either priorities assigned to each constraint or cost penalties for failing to satisfy the constraint, determine which constraints are satisfied in the optimum solution [144].

Continuous CSPs are, in general, more difficult to solve than discrete CSPs. It was seen in Chapter 4, and can also be seen in some of the examples in Kumar's survey [76], that search methods incorporating arc-consistency and backtracking,

although in principle taking exponential-order time, are often fast enough for practical use when applied to discrete CSPs. However, with discrete CSPs, the domain being searched is finite (its size is the product of the domain of each variable). With continuous CSPs, the domain size is (in principle) infinite, and any method which could in principle need to search the entire domain can be rejected.

Lazard [82] describes four general approaches to solution of continuous CSPs. The first two, Gröbner bases and the Wu-Ritt decomposition algorithm [193], are grouped together as "prime decomposition" methods. In dismissing Gröbner bases, Lazard cites one case where calculation of the Gröbner basis took 15 days; the resulting basis was too big to be used in the calculation for which it was intended. The third approach, cylindrical algebraic decomposition, is of limited applicability, and the fourth, "asymptotically-fast algorithms", had not been implemented at the time of writing. It appears that all of these, except possibly the last, can only be used for CSPs expressible as polynomials. Lazard concluded that there were no fully-satisfactory general methods for solving continuous CSPs.

11.2.2 Geometric Constraints

Adapting the definitions in the previous section, a geometric constraint is a geometric relationship between geometric entities (2D or 3D) expressed as one or more functions (equations or inequalities) of those geometric entities [80], and a geometric constraint satisfaction problem attempts to find values (locations and possibly magnitudes and orientations) for those geometric entities which provide a solution to a system of geometric constraints. As such, it is related to the general continuous constraint satisfaction problem, but the knowledge that constraints embody geometric hypotheses introduces additional restrictions based on the properties of three-dimensional space. A geometric constraint system is solved (and the resulting object said to be rigid) if all degrees of freedom, other than those required for location and orientation, have been removed.

For example, if vectors \mathbf{a} and \mathbf{b} are both perpendicular to two non-collinear vectors \mathbf{c} and \mathbf{d} , then \mathbf{a} and \mathbf{b} are parallel. It can therefore be seen that the constraint system

$$\mathbf{a} \cdot \mathbf{c} = 0$$
; $\mathbf{a} \cdot \mathbf{d} = 0$; $\mathbf{b} \cdot \mathbf{c} = 0$; $\mathbf{b} \cdot \mathbf{d} = 0$

tacitly requires that $\mathbf{a} = k\mathbf{b}$, although this is not expressed explicitly in any constraint. Any solution to a 3D geometric constraint problem must satisfy these tacit additional constraints as well as the explicit constraints of the problem itself.

Two particular consequences of these tacit constraints have been investigated in detail: the problem of counting degrees of freedom, and the problem of finding a resolution sequence. Work in these areas is summarised later in this section.

Methods for solving geometric constraint systems can be classified into four general approaches: symbolic, rule-based, graph-based and numerical.

Symbolic approaches to geometric constraint systems such as those of Kondo [72] and Gao and Chou [30] use Gröbner bases to manipulate algebraic expressions of geometric constraints. Kondo [72] has implemented a 2D geometric constraint solver which uses Buchberger's algorithm [9] to find a Gröbner basis for constraint equations. Although more general than a previous geometric constraint solver based on constraint propagation [71], the symbolic approach is very slow even for simple 2D problems. The method should in principle be extensible to 3D problems, but the added number of variables would slow the method down further [72]. As part of a general investigation into various methods of solving geometric CSPs, Gao and Chou [30] have produced a 2D constraint solver similar to Kondo's and, in addition, investigated an alternative to Gröbner bases, the Wu-Ritt decomposition algorithm [193], which appears to be preferable. They note that this method is exponential in both the number of variables and the degree of the polynomial in constraint equations (which is clearly discouraging if one wants to approximate a trigonometric function by taking the first few terms of the polynomial expansion). These approaches are extremely slow and can be rejected for use in interactive systems (neither could they be recommended for batch systems). A somewhat faster alternative based on Dixon resultants has been suggested by Kapur et al [65], who point out that although this method initially appears unpromising as involves computation of a matrix which for most geometric problems is singular, there exist fully-automatic methods for producing non-singular matrix representations of problems. They report that their approach can solve in minutes comparatively simple problems for which Gröbner basis methods take days of computation (when they succeed at all), but acknowledge that a great deal of further investigation is needed to produce general-case solutions using Dixon resultants. This remains the most

promising of the symbolic approaches. No reference has been found to any investigation of cylindrical algebraic decomposition in the context of geometric constraints.

Rule-based approaches attempt to deduce from the constraint system a sequence of rules for fitting geometry to the given topology. Gao and Chou [29] have produced a rule-based 2D constraint solver which runs in interactive time on over 90% of their test cases; however, the rules described are specific to 2D. Verroust et al [179], who demonstrate the capabilities of a 2D rule-based approach, nevertheless note that a rule-based method for fitting a geometry to all possible 2D topologies would require an infinite number of rules; since this presumably also applies in 3D, rule-based methods are clearly inappropriate for the purposes of this thesis. As a further disincentive to their use, it is generally believed (e.g. [191]) that rule-based approaches are inefficient for large systems of constraints even when all required rules are available. Much of the success of Gao and Chou's implementation [29] seems to derive from the decision to build a database of geometric information about the problem before attempting to derive rules.

Graph-based approaches create a graph representation of the variables affected by each constraint in the constraint system; each graph-vertex corresponds to a variable, and each graph-edge corresponds to a constraint.

Kramer [73, 74] describes algorithms which search for rigid groupings of atoms within an object. Owen [119] extends this idea to producing a hierarchy of rigid groupings of atoms, with larger groupings being assembled by applying intergrouping constraints to rigid smaller groupings; although Owen's algorithm is limited to 2D, it is reported [80] that an unpublished 3D version exists. Bouma et al [4] describe ideas similar to those of Owen, although as they make use of Gröbner bases their approach could be considered a hybrid; since they explicitly consider only those geometric constraint systems describing 2D drawings which can be produced by a ruler and compass, it is reasonable that the resulting constraints can be grouped into a hierarchy, at the bottom of which is the original line or point; it is not clear that the idea can be extended simply to general 3D geometry.

Latham and Middleditch [81] extend previous work in this area to allow constraints which restrict more than one degree of freedom (rotation constraints are an example of this) and to choose correctly between prioritised constraints when the system is over-constrained.

The graph constructive method of Li et al [88] uses dependency analysis to break a constraint system down into clusters. This assumes that such clustering is possible; as their interest is in linkages where most constraints are distance constraints between neighbouring vertices in the linkage, this is usually the case. Their illustrative example, a CAD model of a bicycle, illustrates an application for which dependency analysis is ideal: everything is to be constrained relative to a simple basic framework.

Latham, an advocate of graph-based algorithms, nevertheless notes its limitations [80]. Firstly, as they consider which constraints are functions of which variables but not numerical function values, graph-based algorithms may misidentify whether or not the geometry resulting from satisfying the constraint system is rigid (Latham also gives a 2D example where rigidity is misdiagnosed as a direct result of graph analysis, not through ignoring function values). Also as a result of ignoring function values, graph-based algorithms cannot detect inconsistent constraints (although Latham points out that if inconsistency can be detected by other means, graph-based algorithms can locate the cause). The constraint system may not be one which is easily-partitioned by graph analysis (it is found, in practice, that the constraint systems generated by RIBALD do not have simple loose ends which can be pulled to unravel the entire system). For these reasons, this thesis avoids a graph-based approach to constraint satisfaction.

Hence, this thesis takes the numerical approach to solving geometric CSPs. As an example of the pure form of numerical approach, Ge, Gao and Chou [32] use a downhill method (using BFGS [7] as a black-box downhill optimiser) to find solutions to a number of 2D geometric CSPs. On these problems, results are obtained in interactive time; however, the problems are simple ones and they make no mention of the order of the algorithms involved, so although the method could be adapted to 3D CSPs, the results might be disappointing. Ge, Gao and Chou [32] also note that their numerical approach can fail when the downhill optimiser becomes trapped in a local minimum.

It will be seen later that this pure approach is unsatisfactory for more complex drawings, and domain-specific knowledge is required in order to make the problem more tractable. This produces clusters of geometric variables which can be manipulated together rather than individually. Li et al [89] call such methods graph reduction in view of the similarity between the effects of this domain-specific knowledge and the graph-based approaches already described, and prove the unsurprising result that use of graph reduction accelerates numerical approaches to geometric CSPs.

Angular Degrees of Freedom

Methods of calculating the number of degrees of freedom after satisfying a number of geometric constraints have been the subject of several studies—Sugihara [163] lists a number of these. More recently, Owen's algorithm [119] and its implementation are fast enough, but are restricted to 2D and cannot guarantee to find a solution for underconstrained systems. Kramer [74] has shown the difficulty of allowing for geometrical coincidences in two dimensions. Whiteley's [192] 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. Latham [80] acknowledges that graph-based methods can misdiagnose the presence of geometric degrees of freedom.

Essentially, the problem here is that of degeneracy. Through the consequences (some of them subtle) of tacit constraints, different combinations of explicit constraints may reduce to the same information, and existing constraints may become equivalent when a later constraint is accepted.

A method for detecting degeneracy has recently been suggested by Li et al [89]: given a solution to a system of a number of constraints, perturb one of the constraints; if the perturbed system also has a solution, that constraint is not degenerate. This method is not in itself useful for the purposes of this thesis—once a solution has been found to a constraint system, it matters little whether or not the system contains degenerate constraints. However, it suggests a line of investigation for the angular degrees of freedom problem which future work should pursue.

Even when the number of degrees of freedom required for a set of constraints is known (for example, each rotation constraint reduces the total number of degrees of freedom in three planar faces by two) there appears to be no satisfactory method for determining which specific variables lose their freedom when a constraint is enforced. For example, in the above case, there appears to be no satisfactory method for deciding which of the three face normals lose degrees of freedom when such a constraint is accepted.

Resolvable Representations

In principle, constraint satisfaction problems are solved by removing degrees of freedom until a solution is found. In trivial problems, the order in which degrees of freedom are removed is unimportant—in solving the system (x = 1; y = 1) either xor y may be determined first. This is not always the case with geometric constraint systems, as will be shown. In such cases, there is not only the primary problem, that of finding a solution, but the secondary problem of finding a route towards the solution. Such a route is termed a resolution sequence, or resolvable representation.

To illustrate the problem, consider what happens when the methods of this chapter—firstly determine face normals, and then face distances—are applied to a topological model of an octahedron (assuming that although the object is topologically identical to the regular octahedron it may not have the geometric symmetry of an octahedron). Face normals may be derived for each of the eight faces, but (for example) opposite faces may not be parallel. The question to be considered is whether it is possible, with any arbitrary set of face normals, to find a sequence in which the face distances and vertex coordinates can be fixed.

This can be analysed as a simple *qame* where:

- each move towards completing the geometry comprises fixing a face distance or fixing a vertex location
- since the face normals have already been determined, fixing a vertex location fixes the distances of all faces on which that vertex lies
- fixing the face distances of any three faces on which the vertex lies fixes the vertex location
- fourteen moves must be made to complete the object
- to win this game, a sequence of moves is required in which no vertex location is fixed after fixing the face distances of three of the faces on which the vertex lies, and no face distance is fixed after fixing the vertex location of any vertex on that face.

It follows that there is no way to win this game. The final move cannot be to fix a vertex location—each vertex lies on four faces; the vertex location will have been fixed when the distances of three of these faces are fixed, so fixing the distance of the fourth face must follow fixing the vertex location. The final move cannot be fixing a face distance—each face contains three vertices, and fixing the location of any one of these vertices fixes the face distance, so fixing the locations of two vertices must follow fixing the face distance. Since there is no possible final move in the sequence, there is no possible sequence. At least in the case of the octahedron, it may not be possible to determine all face normals in advance, and then find vertex locations and face distances which produce a consistent geometry.

However, if the rules of the game are relaxed to allow vertex coordinates to be determined before face normals, then there is clearly a resolution sequence for the octahedron: determine all vertex coordinates, and everything else follows as all faces are triangular and it is always possible to fit a plane through three points. Thus the octahedron has at least one *resolvable representation*, which is to fix all vertex coordinates first and compute face equations from them.

Obviously, fixing face normals first, and then face distances, provides a resolvable representation for all trihedral polyhedra. Similarly, but less usefully, fixing vertex coordinates first provides a resolvable representation for all deltahedra [165].

Sugihara has shown [165] that all genus-zero polyhedra have resolvable representations, and notes that these can be found using the Hopcroft-Tarjan algorithm [53] for trivalent decomposition of graphs. However, for some non-trihedral genus-zero polyhedra, fixing face normals first fails to achieve a resolvable representation. The octahedron is one example of this.

Sugihara has also shown [165] that some polyhedra with through holes have no resolvable representation, using the solid illustrated in Figures B.410–B.412 (page 325) to demonstrate the point. Mills [110] demonstrates that by use of "scaffolding" (temporary faces or vertices added to the resolution sequence but not forming part of the object) second-order resolution sequences can be found for Sugihara's example and many other objects with no first-order resolvable representation. However, finding such solutions relies on human ingenuity—no algorithm is known. Additionally, although it is plausible that second-order resolution sequences exist for all polyhedra, this has not as yet been proved.

11.2.3 Beautification

Beautification takes a valid solid model and adjusts its geometry (and in some cases its topology) to produce a "more beautiful" object (although, proverbially, beauty is in the eye of the beholder, there is a consensus that, for example, 90° angles are more beautiful than 89° angles, and squares are more beautiful than rectangles with an aspect ratio of 1.01). Beautification need not proceed by identification and enforcement of constraints, but in practice (e.g. [78, 79]) it usually does, and it can therefore be treated as a special case of geometric constraint satisfaction problem. The availability of a starting point—a valid solid model—helps matters in that downhill optimisation methods are more reliable—they are likely to start in the desired valley, and unlikely to move so far away as to become trapped elsewhere [131].

Werghi et al [191] have investigated how solutions to geometric CSPs can be applied to beautification, concentrating on numerical approaches. Although most of their test results are for curved objects, their investigations provide helpful information for the more restricted case of polyhedra (elsewhere [190], the same authors point out that, paradoxically, beautification of polyhedra is often harder than beautification of curved objects—planar surfaces are usually intended to be functional, and must therefore be machined to a much higher accuracy than freeform curves, which are usually intended to be decorative).

Werghi et al [191]:

- Note that since geometric CSPs are in general non-linear, finding analytical solutions to all but the simplest geometric CSPs is impractical. Numerical approaches, being simplest, are recommended.
- In discussing algorithms, they consider both deterministic optimisation methods and evolutionary methods, concentrating on genetic algorithms as an example of the latter. Observing that genetic algorithms are significantly slower than more traditional optimisation methods, they note that although use of genetic algorithms may be justified when the objective function is non-differentiable or has no explicit form, geometric CSP objective functions are normally well-behaved and the overhead of using genetic algorithms cannot be justified.

- The deterministic algorithm they investigate is the Levenberg-Marquardt algorithm [102], enhanced by some ideas of Broyden's [8] and some of their own; effectively, this is related to BFGS [7] but the choice of objective function is limited to least-squares. As this thesis uses amoeba [117] for non-linear optimisations, RIBALD avoids the problems of ill-conditioned Hessians encountered by Werghi et al.
- A good initial variable vector is important. They recommend that this should use those values which, in the absence of any constraints, would fit the data best. This is clearly sensible and has been adopted in this thesis.
- Importantly, when constraints are processed sequentially, addition of a new constraint does not affect the satisfaction of previously-enforced constraints.
- Constraint validity and consistency checking must be done before starting the optimisation process (I disagree with this conclusion—see Section 11.4 onwards).
- The times taken, on equipment similar to that used in this thesis, are of the order of a few minutes for comparatively simple objects. Such times, although satisfactory for reverse engineering, are unacceptable for an interactive design system.

Although Werghi et al [191] used an objective function based on least-squares for their initial investigations, this is a consequence of their optimisation algorithm, not a deliberate choice, and indeed a case can be made for other functions. Where data contain outliers, other objective functions have been recommended, such as least median squares [141] (which has been used with success in other vision applications, e.g. [151]) or least trimmed squares [142]. This might be especially useful in geometric constraint fitting, if combined with the suggestion that more than half of the constraints under consideration at any time are genuine, and the bad ones are "outliers" to be discarded. In this thesis, another alternative, based on the figures of merit defined in Appendix D, has been investigated.

As I have noted elsewhere [172, 171], use of a single symmetry element (such as a mirror chain) to derive geometry directly is particularly sensitive to freehand drawing inaccuracies. It is also inappropriate to many drawings, either because they

contain no symmetry element or because they contain several. This approach can be rejected.

Turner [166] has suggested that beautification is a process of adjusting location estimates by translation or rotation to satisfy constraints, and that providing initial location estimates are good, such adjustments will be small enough that in any rotations by angle θ , the approximations $\sin \theta = \theta$ and $\cos \theta = 1$ will be valid. By this means, beautification becomes a linear optimisation problem. This suggestion would fit in well with the ideas of Section 11.6, but there has not been time to investigate this.

As noted in Chapter 7, the problems of beautification and inflation are not always distinct. For example, Grimstead [38] uses the same linear system of equations $P_f x_v + Q_f y_v + z_v + C_f = 0$ to enforce geometric constraints as he uses for inflation of frontal geometry. His system attempts to iteratively delete constraints which disagree most with the overall best fit, thereby achieving a fit to a consistent set of constraints (there is, as Grimstead admits, no guarantee that "bad" constraints rather than "good" ones are deleted). Finally, it uses the face equations to give the 3D locations of each vertex by intersection (the values of x_v and y_v from the drawing and z_v from the linear system are discarded). Given the restriction to trihedral vertices, this ensures a self-consistent boundary representation model.

Grimstead's approach is unsatisfactory for a number of reasons:

- It does not distinguish between constraints which *must* be met and equations for which a best-fit approach is adequate. Even the constraint that ensures that vertices lie on faces is only approximately satisfied, and has to be enforced explicitly at a later stage. Others, such as making two faces parallel or orthogonal, are also only approximately enforced.
- By using a linear system to solve the constraint system, Grimstead limits the set of constraints which can be considered to those which translate into equations linear in his variables $(P_j, Q_j \text{ and } C_j \text{ for each face } j \text{ and } z_i \text{ for each vertex } i)$. This is a serious limitation as several useful constraints do not translate into equations linear in these variables. For example, the assumption of corner orthogonality, first suggested by Mackworth [97] and used successfully in several systems, does not translate into a linear equation and so is not used.

• The equations of hidden faces are derived from locations of visible vertices, themselves derived from a least-squares fit of various constraints, not all of which are fully-satisfied. Numerical errors are propagated at each stage, and using these error-prone values to calculate the locations of hidden vertices further compounds the error.

Grimstead's approach frequently produces objects with edges which are nearly, but not quite, parallel, and corners which are nearly, but not quite, cubic corners. Despite these disadvantages, the approach has one major advantage in that it runs sufficiently quickly to be interactive for drawings such as Figure B.91.

11.3 Constraints

As stated in the previous section, hypotheses about the geometry of an object become *constraints* when expressed as equations or inequalities. This section considers both the types of constraint available and the hypotheses which lead to them.

Constraints are allocated a figure of merit. In RIBALD, this is based on three contributory factors: the figure of merit for the hypothesis, a fixed value for the particular type of hypothesis and constraint, and a figure of merit for how well the constraint is met in the provisional geometry. As with topological hypotheses which generate the same move, when two or more geometric hypotheses lead to the same constraint, RIBALD reinforces the merit of the first constraint rather than generate a second constraint. An attempt is made 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 a geometry can be found which agrees with both the current and all previously-accepted constraints.

The fixed factor in each figure of merit should, ideally, be a tuning constant. There has not been time to perform tuning, so arbitrary values are used.

Mirror planes can be treated in the same way as faces: they have normals and distances, and can be constrained using the constraint types listed.

It will be seen in the remainder of the section that the number of constraints can be limited to $O(n^2)$.

11.3.1 Parallelism Constraint

A parallelism constraint requires two faces M and N to be parallel (i.e. their normals $\hat{\mathbf{n}}_{\mathbf{M}}$ and $\hat{\mathbf{n}}_{\mathbf{N}}$ must satisfy $\hat{\mathbf{n}}_{\mathbf{M}} = \pm \hat{\mathbf{n}}_{\mathbf{N}}$). There are $O(n^2)$ such constraints possible. Parallelism constraints affect only face normals, not face distances. For all parallelism constraints, the measure of how well the constraint fits the provisional geometry is the figure of merit for parallelism between the two normals hypothesised to be parallel.

In Chapter 5, lines in the drawing are allocated to bundles, in each of which all lines are nearly parallel and expected (very occasionally, bundling is misleading) to correspond to edges which are parallel in 3D. For each face, the bundles to which its edges have been allocated are tabulated. The normals of 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 fixed merit factor for each such constraint is 0.99.

Three 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. 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 (see Section 11.3.2) are generated between each vertical and each horizontal face. The fixed merit factor for each such 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.

In some cases, earlier stages of processing will have 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 normalons, and often the case with many of the faces in semi-normalons. For each pair of faces with normals successfully allocated to the same bundle, a parallelism constraint is generated. The fixed merit factor for this is 1.

The proposed method above 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 11.5). More importantly, if some face normals cannot be made parallel to the desired direction, but could be made parallel to one another, the proposed method can enforce this, whereas the alternative cannot.

Since axis-aligned faces are common, RIBALD generates either a parallelism constraint or a two-way perpendicularity constraint (see Section 11.3.2)—whichever has the higher merit—for each pair of faces in the object. This is intended as a safety net, to ensure that there are *some* constraints to enforce even in the most irregular object. The figure of merit is that for parallelism between the two face normals.

Parallelism constraints can also be generated from face-centred rotation hypotheses: see Section 11.3.6 below.

11.3.2 Two-way Perpendicularity Constraint

A two-way perpendicularity constraint requires two faces M and N to be perpendicular (i.e. $\hat{\mathbf{n}}_{\mathbf{M}} \cdot \hat{\mathbf{n}}_{\mathbf{N}} = 0$). There are $O(n^2)$ such constraints possible. Two-way perpendicularity constraints affect only face normals, not face distances. For all perpendicularity constraints, the measure of how well the constraint fits the provisional geometry is the figure of merit for perpendicularity between the two normals hypothesised to be perpendicular.

Currently, no hypothesis generates *only* two-way perpendicularity constraints. Such constraints are generated by the "safety-net" (see Section 11.3.1 above), from mirror chains (see Section 11.3.5 below) and from cubic corners (see Section 11.3.3).

11.3.3 Three-way Perpendicularity Constraint

A three-way perpendicularity constraint requires three faces M, N and O to be mutually perpendicular (i.e. $\hat{\mathbf{n}}_{\mathbf{M}} \cdot \hat{\mathbf{n}}_{\mathbf{N}} = \hat{\mathbf{n}}_{\mathbf{M}} \cdot \hat{\mathbf{n}}_{\mathbf{O}} = \hat{\mathbf{n}}_{\mathbf{N}} \cdot \hat{\mathbf{n}}_{\mathbf{O}} = 0$). In principle, there are

 $O(n^3)$ such constraints possible, so the number of three-way perpendicularity constraints must be limited by the generating hypotheses. Three-way perpendicularity constraints affect only face normals, not face distances.

For each vertex which could be a cubic corner [125], one three-way perpendicularity constraint and three two-way perpendicularity constraints are generated. The hypothesis merit factor for each constraint is that for the object being a normalon or semi-normalon. As this is the only source of three-way perpendicularity constraints, the number of these is thus O(n).

11.3.4 Face Angle Constraint

A face angle constraint of angle ρ requires two faces M and N to be at a defined, non-perpendicular angle (i.e. $\hat{\mathbf{n}}_{\mathbf{M}} \cdot \hat{\mathbf{n}}_{\mathbf{N}} = \cos \rho$). I found during early experimentation that very implausible perpendicularity constraints were enforced between faces which were clearly not perpendicular in the drawing 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 circumvent this by introducing an arbitrary numerical merit threshold below which all constraints are rejected, RIBALD includes face angle constraints which requires two faces M and N to have a fixed "common" angle ρ between them. As with parallelism and perpendicularity, there are $O(n^2)$ such constraints possible, and face angle constraints affect only face normals, not face distances.

RIBALD assumes ρ to be acute. Extension to obtuse angles would would be straightforward but unnecessary—if, for example, 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. Although the constraint type definition makes no other assumption about ρ , it will in practice be 30°, 45° or 60°, or other angles whose tangent is the ratio of integers each in the range 1–6, as these are the only angles RIBALD looks for when generating constraints. The latter cases arise commonly in semi-axis-aligned wedges, a common design feature according to [143].

RIBALD generates a face angle constraint for each pair of faces in the object, considering the angles 30°, 45°, 60°, and angles the tangent of which is a ratio of two small integers in the range 1–6. The constraint is generated for whichever angle

produces the highest merit figure. This is another safety net, to avoid spurious parallelism and perpendicularity constraints being enforced. The figure of merit is $0.9 \times$ the figure of merit for the angle being correct—parallelism and perpendicularity constraints are somewhat to be preferred, but not if the angle strongly suggests something else.

11.3.5 Mirror Constraint

A mirror constraint requires two faces M and N to be related via a mirror chain C such that reflection through the mirror plane defined by C moves M to the location occupied by N and vice versa. There are $O(n^2)$ such constraints possible, since there are O(n) mirror chains and O(n) initial faces and each mirror chain will reflect each source face into one and only one destination face. Mirror reflection constraints affect both face normals and face distances.

For each mirror chain (C_i) in the object, RIBALD identifies the pairs of distinct faces (N_{ij}, M_{ij}) which are reflected into one another by the mirror chain and generates a constraint. This will normally be a mirror constraint linking $N = N_{ij}, M = M_{ij}, C = C_i$, except for the special case where $M_{ij} = N_{ij}$ (this includes all faces in the mirror chain and possibly others), where M_{ij} must be perpendicular to the mirror plane C_i and a two-way perpendicularity constraint is produced instead.

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 [172], 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. The power terms add further bias in favour of mirrors which (a) propagate through the entire object and (b) reflect the entire object—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 classified as a semi-normalon with mirror chain, the figure of merit is reinforced by that for the classification.

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 $\hat{\mathbf{n}}_{\mathbf{M}}$ and the vector obtained by reflecting normal $\hat{\mathbf{n}}_{\mathbf{N}}$ through the mirror plane.

Additionally, the mirror plane C is perpendicular to each face in the chain, C_1, C_2, \ldots Two-way perpendicularity constraints are generated for these, making $\hat{\mathbf{n}}_{\mathbf{C}} \cdot (\hat{\mathbf{n}}_{\mathbf{C}})_1 = \hat{\mathbf{n}}_{\mathbf{C}} \cdot \hat{\mathbf{n}}_{\mathbf{C}_2} = \ldots = 0$. Since there are O(n) mirror chains and the length of any chain is O(n), there are $O(n^2)$ such face/mirror perpendicularity constraints.

11.3.6 Rotation Constraint

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 the centre of face N to the location and orientation occupied by the centre of face M. There are $O(n^2)$ such constraints possible, since there are O(n) face-based rotation axes and O(n) initial faces and each rotation will rotate each source face into one and only one destination face. Rotation constraints affect both face normals and face distances.

As a consequence of the way rotation constraints are generated, ρ will always be one of the following: 60°, 72°, 90°, 120° or 180°.

For each face (R_i) in the object containing an axis of rotation, RIBALD identifies the face (M_{ij}) to whose location each face (N_{ij}) is rotated and (providing $M_{ij} \neq N_{ij}$) generates 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, except for the trivial case $M_{ij} = N_{ij} = R_i$ which is ignored.

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 [172], 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 $\hat{\mathbf{n}}_{\mathbf{M}}$ and the vector obtained by rotating normal $\hat{\mathbf{n}}_{\mathbf{N}}$ around the rotation axis. Hypotheses based on rotations are allocated figures of merit based on a fixed probability for

each type of rotation, with C_4 and C_6 being given higher probabilities. The assessment is decreased by a factor based on the number of unmatched vertices left after attempting to match the object with its rotated equivalent—the more unmatched vertices which remain, the greater the decrease.

RIBALD does not include geometric constraints based on vertex-centred or edgecentred rotations; these were considered to be less useful and thus lower-priority, and there was not time to incorporate them.

11.3.7 Face Distance Constraint

A face distance constraint requires that the distances of four faces A, B, C and D are related such that $d_A - d_B = d_C - d_D$. There are, in principle, $O(n^4)$ general face distance constraints, so these must be limited further by restricting the hypotheses which lead to them. Face distance constraints affect only face distances, not face normals.

RIBALD does not currently generate face distance constraints. These should be generated from mirrors, in order to enforce the distance relationship between faces 1 and 3, and 6 and 7, shown in Figure 11.2 on page 263.

11.3.8 Face Coplanarity Constraint

A face coplanarity constraint requires that the distances of two faces M and N are equal, $d_M = d_N$. There are $O(n^2)$ possible coplanarity constraints.

RIBALD currently generates face coplanarity constraints from two sources. Firstly, where a mirror chain C_i reflects face N_{ij} into face M_{ij} and either of the two face normals is known to be perpendicular to the mirror normal, the two faces must be coplanar. Secondly, where two faces M and N are approximately coplanar in the provisional geometry, they may be intended to be coplanar.

11.3.9 Edge Length Ratio Constraint

An edge length ratio constraint requires the ratio of lengths of two edges E_1 and E_2 to be $(n_1)/(n_2)$, where n_1 and n_2 are small integers. The most common such constraints will be equi-length constraints, specifying $n_1 = n_2 = 1$. For any predetermined set

of values for n_1 and n_2 , there are $O(n^2)$ possible edge length ratio constraints. Edge length ratio constraints affect only face distances, not face normals.

Since line lengths in other small integer ratios are not a common feature of parts, such constraints have lower merit.

Whether equi-length constraints between edges bundled together should have higher merit than equi-length constraints between edges in different bundles is unresolved; cases can be made for and against the idea.

Currently, RIBALD generates an equi-length constraint and one other length ratio constraint (using the nearest small-integer ratio) for each pair of edges, relying on logical reasoning to rule out low-merit constraints.

11.3.10 Non-Trihedral Vertex Constraint

A non-trihedral vertex constraint forces all faces meeting at vertex V to pass through a single point. There are O(n) possible non-trihedral vertex constraints. Since the presence of a non-trihedral vertex rarely, if ever, provides a useful clue to face orientations, and it is always possible to ensure that all faces meeting at a vertex pass through a single point by adjusting face distances while preserving face orientations, non-trihedral vertex constraints affect only face distances, not face normals.

For each non-trihedral vertex in the object, a non-trihedral vertex constraint is generated. The merit figure for such a constraint is 1, and is not adjusted—the planes of all faces meeting at the vertex *must* pass through the vertex.

11.4 Face Normals—Simple Downhill Optimisation

This section aims to use the vertex coordinates generated during topological reconstruction to provide preliminary estimates of face normals, and adjust these normals until as many high-merit constraints as possible are satisfied. As stated, this is an NP-complete knapsack problem. Instead of attempting a rigorous solution, RIB-ALD uses a "greedy" approach. Constraints are enforced in descending order of merit until they specify a unique object, initially using the naive algorithm:

- the constraint with the highest figure of merit is always accepted and enforced
- for each other constraint, in descending order of merit:
 - if the constraint is already satisfied numerically by the object, it is accepted;
 - attempt to adjust the existing face normals numerically 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 constraint is rejected and the previous face normals are restored;

As the numerical processing required is considerable, this is slow—faster and more sophisticated refinements are described in the next Section. This Section describes necessary parts of the algorithm—initial conditions (the preliminary estimate of face normals), the objective function, and some implementation details.

There is one iteration of the loop per constraint; if the number of constraints is limited to $O(n^2)$, there will be $O(n^2)$ iterations of the loop. Within the loop, the rate-determining step is adjustment of existing face normals, performed using a black-box optimiser amoeba [117, 131]. As seen in Chapter 3, in the worst case, when amoeba fails to converge, it makes a fixed maximum number of calls of the objective function, and its internals take $O(v^2)$ time, where v is the number of variables. Assuming that it takes a fixed time to assess how well or badly a single constraint is met, since there are $O(n^2)$ constraints to be considered, and the number of variables is proportional to the number of faces and therefore O(n), both calls to the objective function and amoeba's internals take $O(n^2)$ time. Thus, overall, the face normal process outlined here takes $O(n^4)$ time, albeit with an uncomfortably large constant.

11.4.1 Preliminary Estimates of Face Normals

The optimisation process requires a preliminary geometry (a) as a basis for computing the numerical estimates of merit of 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.

Preliminary estimates of face normals are calculated from the vertex coordinates generated by inflation (Chapter 7) and topological reconstruction (Chapter 10). For all but triangular faces (which can be solved directly) RIBALD uses a least-squares linear system [3] with weightings which give priority to visible vertices. The algorithm is described in detail in [178].

11.4.2 Objective Function

The objective function measures how well constraints are met by a set of face normals, and is the function to be minimised by the optimisation process. A value of zero indicates that all constraints are met perfectly. It is computed as the numerical sum of terms for each constraint under consideration, as listed here:

- 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 relocated using current estimates of N and C
- the term for a rotation constraint is $1-F(M \parallel M'(R, N, \rho))$, where $M'(R, N, \rho)$ is face M relocated using current estimates of N and R
- the term for an angular constraint is $1 F(M \parallel M'(N, \rho))$, where $M'(N, \rho)$ is face M relocated using current estimate of N

where figures of merit F are as listed in Appendix D^1 .

11.4.3 Implementation Details

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. RIBALD uses 1000 iterations as the maximum; this is not quite free from either problem but is a reasonable compromise.

¹It could be objected that the 1- terms are redundant, but this overhead does not affect the conclusions reached in this Chapter.

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. RIBALD use 1/1000 (the numerical value is meaningful only in terms of the objective function).

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

11.4.4 Alternatives Investigated

Two variants of the method of simple downhill optimisation of face normals were investigated in an attempt to overcome a problem observed in practice. Although amoeba always moves downhill, it can become trapped in a local minimum. When this happens, some "good" constraints are rejected because they cannot be satisfied within the locality, and in some but not all cases this affects the resulting geometry (sometimes a later constraint, expressing the same geometrical relationship in a different way, may be accepted).

In an attempt to remove the local minimum problem, I used a version of amoeba incorporating simulated annealing [131]. 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 approach was rejected. Another non-deterministic alternative is described in Section 11.7.

I also attempted to refine the initial normal estimates using skewed symmetry [63]. I found no benefit in doing this. Even without 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, I found the resulting time differences to be negligible.

11.5 Face Normals—Enhanced Downhill Optimisation

The time-consuming part of the method in the previous Section is numerical adjustment of face normals using a downhill optimiser. To improve on it, the more sophisticated algorithm outlined below is designed to use logical reasoning to bypass numerical processing as often as possible. Although the order of the algorithm is unchanged, 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 always accepted and enforced
- for each other constraint, in descending order of merit:
 - if logical reasoning using known information about the face normals can show that the constraint is necessarily valid, it is accepted;
 - if logical reasoning can show that the constraint is necessarily invalid, it is rejected;
 - if the constraint is already satisfied numerically by the object, it is accepted;
 - if enough angular degrees of freedom remain in the affected faces, the constraint is accepted;
 - if the object has enough angular degrees of freedom elsewhere, an attempt is made to adjust all movable 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 discarded;
 - otherwise, the constraint is discarded.

This section discusses the additional methods required by this more sophisticated algorithm: logical reasoning which can in certain circumstances show that a constraint is necessarily valid or invalid, and an attempt to calculate the number of degrees of freedom left both at a particular face and in the object as a whole. Two types of logical reasoning are considered: that concerning the relationship between pairs of face normals, and that concerning the relationship between each face normal and the main axes of the object.

11.5.1 2-Face Relationships

A considerable improvement in performance is possible if deduction can determine whenever two faces are of necessity either parallel or perpendicular to one another—the performance improvement is particularly significant for semi-normalons. In many cases, parallelism and perpendicularity constraints can be accepted or rejected without any numerical processing, and in some cases mirror constraints and rotation constraints (particularly from C_2 and C_4 symmetry) can also be accepted or rejected immediately.

To carry out this logical process, two faces are considered to be in one of three mutually-exclusive 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 faces meeting at a vertex can only be parallel if that vertex is extended trihedral or K-type. If the object is a normalon 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 deduction of whether a given relationship is necessary (the only remaining state) or invalid (not in the set of remaining states).

Interrogating the status of relationships (known to be true, possible but uncertain, or known to be false) is straightforward. To help the process along, extra inference rules embodying the tacit constraints of 3D space can be used to restrict the set of states further. RIBALD includes the following:

- 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

In some cases, when a constraint is accepted, it may be possible to re-interpret existing constraints—it may, for example, be possible to convert accepted mirror or rotation constraints to parallelism and perpendicularity constraints, in which case the knowledge could be added to the two-face database and the constraints removed from the degrees of freedom database (Section 11.5.4). This has not been investigated.

11.5.2 Face-Axis Alignment

The relationship between each face normal and the three main axes (I, J, K) of the object must logically be one of the following:

- The face normal lies along the I axis
- The face normal lies along the J axis
- The face normal lies along the K axis
- The face normal lies in the plane of the I and J axes, but not along an axis
- The face normal lies in the plane of the I and K axes, but not along an axis
- The face normal lies in the plane of the J and K axes, but not along an axis
- The face normal lies somewhere else entirely

Initially:

- The normal of the arbitrarily-chosen reference face (see Section 11.5.3) is aligned along the I axis.
- The normal of the second reference face (see Section 11.5.3) is either along the J axis or in the IJ plane.
- Every other face has all seven possibilities

It would require considerable effort (and many inference rules) to keep this database in step with the two-face database (Section 11.5.1). However, it is not necessary to ensure a perfect match, as long as the two do not contain contradictory information.

Whenever a constraint is accepted, the databases are compared so that (for example) when the two-face database lists that two faces are necessarily parallel, the alignment database contains the same possibilities for each face. Currently, RIBALD updates the two-face database whenever a change is made to the alignment database, checking known parallelism, known perpendicularity, known other-angleness, and known non-parallelism, but not vice versa.

Note that each of the two databases contains information not in the other. For example, if it is known that two faces are parallel to one another but their orientation relative to the fixed face is still unknown, the fact of parallelism appears in the two-face database. Conversely, the alignment database is the more useful when assessing mirror constraints (for which purpose the two-face database is usually of little use). In particular, the alignment database allows determination of whether or not the normals of all faces in a mirror chain can be coplanar (if they cannot, the mirror constraint must necessarily be rejected).

One particularly effective inference rule should be noted in the context of the alignment database: in semi-normalons with a predominant mirror chain, if the mirror bundle is axis-aligned, then any face containing an edge bundled in the mirror bundle must be coplanar with its reflection.

I suggest as a further improvement that for drawings classified as semi-normalons, logical reasoning alone could be used to determine which constraints to enforce. Axis-aligned face normals could be aligned to the main axes, as in Section 11.11, and any remaining face normals determined either by a single numerical optimisation (which would be very quick) or by deskewing whichever face or faces will give the best estimates of the non-axis-aligned normals (which would be even quicker). As an illustration of the latter, consider Figure B.91—the two non-axis-aligned face normals could be determined simply by deskewing the end cap. There has not been time to produce a working implementation to test this idea.

11.5.3 Angular Degrees of Freedom I

It was seen in Section 11.2 that the problem of calculating the total number of degrees of freedom left in an object after a number of constraints have been enforced has not been fully solved, and the problem of calculating the number of degrees of freedom of a particular face has hardly been addressed. Even the best currently-available graph-based approaches to determining degrees of freedom sometimes make mistakes [80]. Recognising that attempting to improve on the state of the art in the time available was impractical, early versions of RIBALD [174] used a stochastic method which, however theoretically unsound, worked reasonably well in practice.

Firstly:

the object has remaining degrees of freedom if, using the methods to be described, any face has remaining degrees of freedom.

The upper limit for the number of angular degrees of freedom 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 this must then be reduced to allow for previously-accepted mirror, rotational and angular constraints; this is the non-trivial problem to which no perfect solution is available. The solution adopted by RIBALD is:

- repeat a number of times (RIBALD uses 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 currently being constrained than any other
 try so far, note this as the candidate best solution
- return the number of DoF in the best solution

This method is known to be flawed—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. Occurrences of this phenomenon are rare in practice.

11.5.4 Angular Degrees of Freedom II

The following ideas, developed initially for the purpose of reducing the number of variables in (and thus the time taken by) numerical optimisation, also bear on the problem of angular degrees of freedom.

Faces are categorised as being fixed, wobbly, free, or unknown. Initially, all but two are unknown. Any face which is fixed, wobbly or free is *generatable*.

- A fixed face normal can be generated by a defined *generating method* (these will be detailed in due course).
- A wobbly face N has a normal which moves freely in plane perpendicular to a vector $\hat{\mathbf{p}}$, which must be generatable. It has one degree of freedom, U; the value of the normal $(\hat{\mathbf{n}}_{\mathbf{N}})_i$ is recalculated on each iteration of the downhill optimiser from U and from its initial value $(\hat{\mathbf{n}}_{\mathbf{N}})_0$:

$$\hat{\mathbf{q}} = (\hat{\mathbf{n}}_{\mathbf{N}})_0 \times \hat{\mathbf{p}}; \hat{\mathbf{r}} = \hat{\mathbf{p}} \times \hat{\mathbf{q}}; (\hat{\mathbf{n}}_{\mathbf{N}})_i = \hat{\mathbf{r}} \cos U + \hat{\mathbf{q}} \sin U.$$

A free face has a normal which moves anywhere around the surface of a sphere.
 It has two degrees of freedom, U and V, from which the value of the normal is recalculated on each iteration of the downhill optimiser:

$$\begin{split} &(\hat{\mathbf{n}}_{\mathbf{N}})_i = (\hat{\mathbf{n}}_{\mathbf{N}})_0 \cos U \cos V + ((\hat{\mathbf{n}}_{\mathbf{N}})_0 \times \hat{\mathbf{p}}) \sin U \cos V + ((\hat{\mathbf{n}}_{\mathbf{N}})_0 \times \hat{\mathbf{q}}) \sin V. \\ &\hat{\mathbf{p}} \text{ and } \hat{\mathbf{q}} \text{ are arbitrary unit vectors perpendicular to one another and to } (\hat{\mathbf{n}}_{\mathbf{N}})_0. \end{split}$$

The algorithm for making all faces *qeneratable* is as follows:

- Choose a visible face, giving preference to faces believed to be axis-aligned and to larger rather than smaller faces. Label this face A and set this face fixed. The generating method for $\hat{\mathbf{n}}_{\mathbf{A}}$ is to preserve the initial value.
- Choose another visible face adjacent to face A, again giving preference to faces believed to be axis-aligned and to larger rather than smaller faces. Label this face B. If the dihedral angle between face A and face B is known (i.e. perpendicular or known-angle constraints have been accepted), set this face fixed, and otherwise set it wobbly with its plane of rotation defined by its starting value and $\hat{\mathbf{n}}_{\mathbf{A}}$.
- set all other faces to unknown
- while any face is unknown
 - If any unknown face N is parallel to a *generatable* face F, set it fixed. The generating method is $(\hat{\mathbf{n}}_{\mathbf{N}})_i = (\hat{\mathbf{n}}_{\mathbf{F}})_i$.
 - Else, if any unknown face N is perpendicular to two non-coplanar generatable faces F and G, set it fixed. The generating method is $(\hat{\mathbf{n}}_{\mathbf{N}})_i = (\hat{\mathbf{n}}_{\mathbf{F}})_i \times (\hat{\mathbf{n}}_{\mathbf{G}})_i$.
 - Else, if any unknown face N is perpendicular to a mirror chain C (i.e. $\hat{\mathbf{n}}_{\mathbf{N}}$ is parallel to $\hat{\mathbf{n}}_{\mathbf{C}}$) in which two consecutive faces F and G in C are both generatable, set it fixed. The generating method is $(\hat{\mathbf{n}}_{\mathbf{N}})_i = (\hat{\mathbf{n}}_{\mathbf{F}})_i \times (\hat{\mathbf{n}}_{\mathbf{G}})_i$.
 - Else, if any unknown face N is the reflection of a generatable face F in a mirror chain C in which two consecutive faces G and H in C are both generatable, set it fixed. The generating method is to reflect $(\hat{\mathbf{n}}_{\mathbf{F}})_i$ through the mirror defined by the normal $(\hat{\mathbf{n}}_{\mathbf{G}})_i \times (\hat{\mathbf{n}}_{\mathbf{H}})_i$.
 - Else, (use rotation symmetry along the same lines as above)

- Else, if any unknown face N is perpendicular to a *generatable* face F, set it wobbly. The generating method is based on two vectors, one being the nearest vector to $(\hat{\mathbf{n}}_{\mathbf{N}})_0$ perpendicular to $(\hat{\mathbf{n}}_{\mathbf{F}})_i$, and the second being the cross-product of this and $(\hat{\mathbf{n}}_{\mathbf{F}})_i$ and requires a single parameter U.
- Else, choose an arbitrary face and set it free. The generating method requires two parameters, U and V, as described above.

The total number of variables required in the optimisation process is the sum of the number of parameters required for each face normal. Since a total of zero indicates that the face normal structure is rigid, this clearly has a bearing on the angular degrees of freedom problem, and presents a number of options.

Firstly, the ideas of Section 11.5.3 could be discarded. Zero variables indicates that there are no angular degrees of freedom, and can thus be used as the termination test for constraint enforcement.

This is academically the most respectable, but does not work all that well in practice (it is also noticeably slower than other options). Early on, where some faces remain completely unconstrained, the variables defining their face normals change freely. This (a) gives them absurd values and (b) reduces the ability of the optimiser to make small adjustments to the faces which should be moved, with the result that the objective function never drops below the acceptance threshold and the constraint is rejected.

In principle, this problem could also occur with other alternatives considered here, but in practice it does not, as it is only a problem when there are unconstrained faces whose normals do not affect the objective function in any way. In sensible objects, this only happens early on, when there are many angular degrees of freedom left, and constraints can be accepted automatically.

Secondly, as an improvement on this idea, the optimiser could only adjust those normals which are entangled in some way with the constraint under consideration. This improves not only robustness but also speed.

Particularly early on, this can lead to some optimisations having only one variable. The use of *amoeba* is inappropriate for these, and adding special-case code for 1-dimensional optimisation is justified. Although downhill optimisation is still used for any case with two or more variables, "golden mean" optimisation [131] is

used when there is only one variable (the objective function remains unchanged from Section 11.4).

Determination of which variables affect a particular constraint is not simple. For example, if the aim is to make a free face A parallel to face B and it is known that faces B and C are perpendicular, the remaining degree of freedom in face C is irrelevant even though a constraint relating it to one of the faces under consideration has already been accepted, and including it is actively harmful to the optimisation process. It seems that a face is only "entangled" with a constraint if it is included directly in the constraint, or if constraints have been accepted relating it to two or more faces included directly in the constraint.

This option is the default in the current version of RIBALD, and is the one with which the timings in Section 11.12 were obtained. However, as it is not clear that the implementation is bug-free (or even that all possible problems have been considered), RIBALD also includes a third choice, which is to retain the ideas of Section 11.5.3 for counting angular degrees of freedom and to use the ideas in this section solely for their original purpose, to reduce the time taken by numerical optimisation.

11.6 Face Normals—Geometric Optimisation

The black-box optimiser *amoeba* is ignorant of geometry. It may reasonably be asked whether an optimisation process which uses geometric knowledge would be more effective—faster or more robust.

To test this, RIBALD includes an option to select an alternative iterative optimisation process which uses accepted constraints plus the constraint under consideration to predict updated values of face normals. On each iteration i+1, RIBALD adjusts the face normal $\hat{\mathbf{n}}_{\mathbf{N}}$ of each face N which can move to try to meet the accepted constraints and the constraint under consideration. It estimates $(\hat{\mathbf{n}}_{\mathbf{N}})_{i+1}$ based on each constraint and the values of other normals calculated in iteration i, and uses a weighted average for the overall estimate of $(\hat{\mathbf{n}}_{\mathbf{N}})_{i+1}$, 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 $(\hat{\mathbf{n}}_{\mathbf{M}})_{i+1} = \pm (\hat{\mathbf{n}}_{\mathbf{N}})_i$ and vice versa (the estimate nearer $(\hat{\mathbf{n}}_{\mathbf{M}})_i$ is chosen). If this constraint were

the only constraint affecting the two faces, they would swap normals and never converge. Such behaviour has not been observed in practice, but the alternative of setting both normals to a mean value looks natural and should be investigated.

A perpendicularity constraint between faces M and N predicts a new value $(\hat{\mathbf{n}}_{\mathbf{M}})_{i+1} = \pm \hat{}(((\hat{\mathbf{n}}_{\mathbf{N}})_i \times (\hat{\mathbf{n}}_{\mathbf{M}})_i) \times (\hat{\mathbf{n}}_{\mathbf{N}})_i)$ and vice versa. In principle there should be defensive programming to protect against the possibility that M and N are parallel; in practice, if the situation occurs, something is already drastically wrong.

A mirror constraint between faces M and N and mirror chain C predicts as a new value for $\hat{\mathbf{n}}_{\mathbf{M}}$ the vector obtained by reflecting $\hat{\mathbf{n}}_{\mathbf{N}}$ through the mirror plane: $(\hat{\mathbf{n}}_{\mathbf{M}})_{i+1} = \pm ((\hat{\mathbf{n}}_{\mathbf{N}})_i - 2((\hat{\mathbf{n}}_{\mathbf{N}})_i \cdot (\hat{\mathbf{n}}_{\mathbf{C}})_i)(\hat{\mathbf{n}}_{\mathbf{C}})_i)$, 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 $(\hat{\mathbf{n}}_{\mathbf{A}})_{i+1} = \pm \hat{}(((\hat{\mathbf{n}}_{\mathbf{C}})_i \times (\hat{\mathbf{n}}_{\mathbf{A}})_i) \times (\hat{\mathbf{n}}_{\mathbf{C}})_i)$ as described above.

For a rotation constraint, rotating face N an angle ρ about a normal $\hat{\mathbf{n}}_{\mathbf{R}}$ through the centre of face R to obtain face M:

- if $\hat{\mathbf{n}}_{\mathbf{R}}$ is known, $\hat{\mathbf{n}}_{\mathbf{M}}$ and $\hat{\mathbf{n}}_{\mathbf{N}}$ can be estimated using standard geometry: $(\hat{\mathbf{n}}_{\mathbf{M}})_{i+1} = \pm \Re(\rho, (\hat{\mathbf{n}}_{\mathbf{R}})_i) (\hat{\mathbf{n}}_{\mathbf{N}})_i$ where $\Re(\rho, (\hat{\mathbf{n}}_{\mathbf{R}})_i)$ is the rotation matrix for rotating through an angle ρ about $\hat{\mathbf{n}}_{\mathbf{R}}$.
- the method for estimating $\hat{\mathbf{n}}_{\mathbf{R}}$ when it is $\hat{\mathbf{n}}_{\mathbf{M}}$ and $\hat{\mathbf{n}}_{\mathbf{N}}$ that are known is described in Appendix F.1.

An angular constraint predicts as the estimate for $(\hat{\mathbf{n}}_{\mathbf{M}})_{i+1}$ the vector in the plane of $\hat{\mathbf{n}}_{\mathbf{M}}$ and $\hat{\mathbf{n}}_{\mathbf{N}}$ which is at an angle ρ from $\hat{\mathbf{n}}_{\mathbf{N}}$, i.e.

$$\left(\mathbf{\hat{n}_M}\right)_{i+1} = \left(\mathbf{\hat{n}_N}\right)_i \cos \rho + \left(\left(\left(\mathbf{\hat{n}_N}\right)_i \times \left(\mathbf{\hat{n}_M}\right)_i\right) \times \left(\mathbf{\hat{n}_N}\right)_i\right) \sin \rho.$$

Iterations terminate either when the objective function returns a value below a given threshold, or when the face normals are effectively stationary.

This method is unsatisfactory in 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 could be taking the object away from the optimum geometry (in principle, it could even be oscillatory, although I have not observed this in practice).

Section 11.12 compares the time taken by this idea with that taken by the geometrically-ignorant downhill optimisation of Section 11.4.

Following the ideas of Turner [166], it has been suggested that the speed problems encountered throughout this chapter stem from the fact that 3D geometric constraints are non-linear. It is, in principle, possible that non-linearity could be ignored in an attempt to speed the process up. For example, although the perpendicularity constraint expression $\hat{\mathbf{n}}_{\mathbf{A}} \cdot \hat{\mathbf{n}}_{\mathbf{B}} = 0$ is clearly non-linear if both $\hat{\mathbf{n}}_{\mathbf{A}}$ and $\hat{\mathbf{n}}_{\mathbf{B}}$ are variables, if there is reasonable certainty that the current set of normals are fairly close to the final solution, the perpendicularity constraint could be expressed as two linear equations $(\hat{\mathbf{n}}_{\mathbf{A}})_1 \cdot (\hat{\mathbf{n}}_{\mathbf{B}})_0 = 0$ and $(\hat{\mathbf{n}}_{\mathbf{A}})_0 \cdot (\hat{\mathbf{n}}_{\mathbf{B}})_1 = 0$ where $(\hat{\mathbf{n}}_{\mathbf{A}})_1$ and $(\hat{\mathbf{n}}_{\mathbf{B}})_1$ are variables but $(\hat{\mathbf{n}}_{\mathbf{A}})_0$ and $(\hat{\mathbf{n}}_{\mathbf{B}})_0$ are constants, being the current values. There has not been time to investigate this idea, or even to derive the corresponding expressions for face-angle, mirror and rotation constraints. Presumably, when this idea is tested, equations of the form $(\hat{\mathbf{n}}_{\mathbf{A}})_1 = (\hat{\mathbf{n}}_{\mathbf{A}})_0$ should be included in the linear system in order to preclude the possibility of oscillation.

11.7 Face Normals using a Genetic Algorithm

In Chapter 4, a non-deterministic algorithm was found to be very much faster, but also somewhat less reliable, than the corresponding deterministic algorithm at solving a discrete constraint satisfaction problem. Since the main problem in solving the continuous constraint satisfaction problem posed in this chapter is speed, it is possible that a non-deterministic algorithm could provide an acceptable solution. This has been investigated using a genetic algorithm [35, 52]. This section describes the outline algorithm, some implementation details, and the initial results, which were sufficiently discouraging that the idea was not pursued.

11.7.1 Overall Algorithm

It is clear that a genetic algorithm will be slower than *amoeba* for a single downhill optimisation; thus, using the genetic algorithm in place of the downhill optimisation step in the algorithm outlined in Section 11.4 will inevitably be slow. Instead, the idea tested was to replace the entire process of considering (and accepting or

rejecting) constraints individually in descending order of merit by a genetic algorithm which used a single figure of merit to assess how well the geometry matched all constraints. Note that this has the disadvantage of being a best-fit rather than a selective approach. The algorithm used was:

- Generate the starting population (the problem of what constitutes a population is considered below)
- Evaluate the merit of each of the population
- Remember (as "fittest ever") the best of the population
- Loop
 - Breed two of the population, chosen at random, to produce a new individual
 - Replace the weaker of the two parents by the new individual
 - Evaluate the merit of the new individual
 - If the new individual is better than the fittest ever, remember the new one instead
 - If the maximum number of iterations has been reached, or ten thousand new individuals have been created since the current fittest ever was generated, then exit the loop
- End loop
- Interpret the fittest ever member of the population as a geometry, and use this

Traditionally [35, 52], information is encoded as "genes" which are bit strings. Encoding face normal information as a bit string was not straightforward—it must be relatively concise, in order that no bits in the gene are irrelevant, but it must also be flexible enough to allow for realistic geometric information. The compromise chosen used 15 bits per face:

- ullet Choose face 0 and face 1 as described in Section 11.5; their face normals are $\hat{\mathbf{n}}_0$ and $\hat{\mathbf{n}}_1$ respectively
- ullet Set vector \hat{p} equal to the nearest vector to \hat{n}_1 which is perpendicular to \hat{n}_0

- \bullet Set vector $\hat{\mathbf{q}}$ equal to $\hat{\mathbf{n}}_0 \times \hat{\mathbf{p}}$
- For each face f,
 - Interpret the lower seven bits of the gene as an angle θ , where $0^{\circ} <= \theta < 180^{\circ}$.
 - Interpret the upper eight bits of the gene as an angle ϕ , where $0^{\circ} <= \phi < 360^{\circ}$.
 - The normal for the face is then $\hat{\mathbf{n}}_{\mathbf{f}} = \hat{\mathbf{n}}_{\mathbf{0}} \cos \theta \cos \phi + \hat{\mathbf{p}} \sin \theta \cos \phi + \hat{\mathbf{q}} \sin \theta$.

Constraint identification remained unchanged: N_C constraints C_x have already been listed and assigned a figure of merit M_x to each, and there is a function F(G, C) for assessing how well each geometry G matches a particular constraint C_x . The merit function² is $\sum_{j=1}^{N_C} (F(G, C_x) \times M_x^{\alpha})$.

Each new individual was generated by uniform crossover of the parents. Myers [114] appears to recommend using uniform crossover in early, exploratory phases of the genetic algorithm but switching to multipoint crossover as "areas of optimality" evolve—I did not investigate this idea.

The mutation rate, 1 bit per new individual, was chosen because initial experiments failed through premature convergence. Although it is possible to justify a high mutation rate on the basis that not all bits are equally significant, this rate may still be too high. It can be noted that although on theoretical grounds Myers [114] recommends starting with a low mutation rate and increasing it during the course of the algorithm, his experimental results suggest that mutation rate strategy is not generally a major factor affecting the performance of genetic algorithms.

The method described above breeds around 32,000 individuals for each drawing. Most test results were obtained using Figure B.476. Processing this took approximately 30 seconds, and produced results which varied from mediocre to dreadful depending on values of α . With Grimstead's bracket, Figure B.91, the method took over two minutes and the results were even worse.

Speed could, potentially, be improved by being more selective about which constraints are generated—poor constraints have little impact on RIBALD's speed as they will generally be rejected by logical reasoning, but with the genetic algorithm

 $^{^{2}\}alpha$ was a tuning constant; I found that different values of α worked better for different drawings.

outlined poor constraints have as much impact on running time as good constraints. It is also possible that filtering out poor constraints would improve the quality of output, as even with low figures of merit they will have some effect on the output. However, in view of the poor results obtained with even very simple drawings (the genetic algorithm was slower and produced worse results even than the naive algorithm in Section 11.4) this was not investigated.

Goldberg's [35] "hybrid algorithm", where in each generation selected individuals are improved by non-genetic methods such as downhill optimisation, may be a more promising approach than a "pure" genetic algorithm. This was not investigated.

If the ideas in this Section are revived, a different selection strategy should be tried, a lower mutation rate is recommended, and Myers's idea [114] of switching to multipoint crossover during the course of the algorithm could also be investigated.

11.8 Face Distances—Simple Downhill Optimisation

Once the face normals have been fixed, a unique geometry can be produced by allocating distances (from the origin) for each face. The objective here is to provide values of face distances for each face which fit the faces of the object as closely as possible to the visible vertices while enforcing accepted constraints.

Its simplest form is an n-dimensional iterative optimisation [117, 131] where n is the number of faces in the object and the variables being optimised are the face distances. The objective function being optimised is the sum of the squares of the 2D distances between the predicted x and y coordinates of each vertex and its actual location in the original sketch—this is chosen in order to spread changes from the drawing evenly throughout the solid object, an approach recommended by Grimstead [38]. It can be noted that if one part of the object is particularly badly drawn (such as the misplaced vertex A in Figure 11.1), this objective function will effectively hide the error by spreading it evenly through the object rather than by correcting the error locally; according to Grimstead [38], such errors are less common in practice than sketches where all junctions are close to, but not precisely at, their proper locations. It can also be noted that the consequences of choosing to move

the wrong vertex to "fix" the error are undesirable.

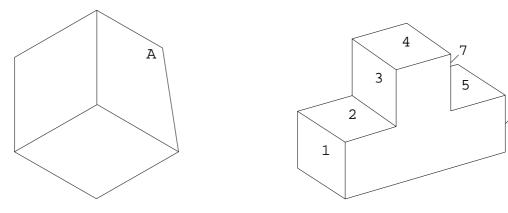


Figure 11.1: Misplaced Vertex

Figure 11.2: T Block

6

This simplest method does not take account of constraints, and suffers from a number of other deficiencies. Most seriously, it does not constrain faces containing no visible vertices. As implemented in RIBALD, the distances of these faces are not changed from the preliminary estimates calculated in Section 11.4.1. 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 most of the drawings 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 11.11.6.

The objective function could be modified to take account of constraints as well as of visible vertex locations. In this method, only those constraints which were accepted during the face normal optimisation process would be used in the objective function (this is not necessarily correct, as consideration of Figure B.454 shows—one topological mirror plane constraint is met by face normals but would be inappropriate for face distances). This variant of the method has not been investigated, as it is clearly both slower and less accurate in enforcing constraints than the variant described next.

This is to reduce the number of variables n after consideration of symmetry, basing this on a subset of the constraints on face normals:

• any constraint rejected during the process of adjusting face normals is discarded;

- parallelism, perpendicularity and angularity constraints have no effect on distances and are not used in distance optimisation;
- 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 11.2)
- 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 11.2)
- the use of rotational symmetry in adjusting face distances remains to be studied—this would be needed if the method were to be applied to drawings such as the dodecahedron, Figure B.116, which has a completely hidden face for which visible vertex locations provide no information.

This variant, implemented in RIBALD, is found to be acceptable in practice for topologically-valid trihedral objects. The time taken is also acceptable provided that the process is seeded with plausible initial values—RIBALD uses 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).

This latter variant is an improvement on its predecessor, in that constraints, if enforced at all, are enforced exactly. It nevertheless retains the inherent problems of all variants of the method in this section: faces with no visible vertices and no accepted mirror or rotation constraints are not constrained, and the choice of

constraints to enforce—those enforced on face normals—may not be appropriate. 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 consequence of the use of face normal constraints, not of the choice of algorithm. Finally, this method does not address the resolvable representation problem. To address these matters requires more complex (and slower) ideas such as those in the next section.

11.9 Face Distances—Enhanced Downhill Optimisation

By analogy with the face normal problem considered in Section 11.5, a general solution to the face distance problem could follow this outline algorithm:

- the constraint with the highest figure of merit is always accepted and enforced
- for each other constraint, in descending order of merit:
 - if logical reasoning using known information about the face distances can show that the constraint is necessarily valid, it is accepted;
 - if logical reasoning can show that the constraint is necessarily invalid, it is rejected;
 - if the constraint is already satisfied numerically by the object, it is accepted;
 - if enough linear degrees of freedom remain in the affected faces, the constraint is accepted;
 - if the object has enough linear degrees of freedom elsewhere, an attempt is made to adjust the existing face distances to accommodate the new constraint as well as all previous accepted constraints; if this succeeds, the new face distances are stored and the constraint is accepted; otherwise, the previous face distances are restored and the constraint is discarded;
 - otherwise, the constraint is discarded.

The principal problems here are: that different types of constraint embody different knowledge, and that of finding a resolution sequence when analysing linear degrees of freedom. There is also the awkward problem of knowing when to stop considering constraints. To illustrate this, consider Figures B.18 and B.17. If all possible symmetry operations are enforced, these will be interpreted as square extrusions (if not as cubes), as C_4 rotations of the end-cap are possible, even though not likely. However, if only the highest-merit symmetry operation is enforced, Figure B.50 will not be interpreted as a hexagonal prism as its highest-merit symmetry operation is a mirror plane. Use of edge length ratio constraints, even if not justified by intentional edge length ratios in the drawing, is justified by the need to avoid such incorrect interpretations.

This section considers knowledge about three types of constraint: face distance relationships (of which face coplanarity is a special case), edge length ratios (of which edge length equality is a special case) and non-trihedral vertices. It then discusses linear degrees of freedom and the unsolved problem of resolution sequences.

Consideration of the constraint types in Section 11.3 shows that unlike face normal constraints, which are always specified in terms of the faces being constrained, constraints on face distances may be specified in terms of vertices, edges or faces, thereby complicating the problem of satisfying them.

RIBALD does not currently identify those faces which do not affect the set of constraints under consideration. This is a serious deficiency. Face distances which make no contribution to the objective function can be moved drastically during the optimisation process, leading to absurd final geometries. Additionally, implementing this will also improve speed somewhat.

11.9.1 Face Distance Relationships

Currently, RIBALD stores a list of accepted face distance constraints. This information is used to reduce the number of variables being optimised, and to enable logical rejection of face distance constraints which clearly contradict accepted constraints.

The effects of face coplanarity in reducing the number of linear degrees of freedom are difficult to assess and have not been resolved. Consider Figure B.328. Between them, the two non-trihedral vertices touch six faces, so the number of degrees of freedom for those six faces is reduced to four. The coplanarity constraint between vertical faces does not reduce this further—there are still four degrees of freedom.

However, the coplanarity constraint between horizontal faces does reduce the number of degrees of freedom.

11.9.2 Edge Length Ratios

Currently, RIBALD stores a list of accepted edge length ratio constraints (equilength constraints are a special case of this, with the ratio being 1). This information is used to enable logical rejection of incompatible edge length ratio constraints. It is clearly possible to use edge length information to reject face distance constraints, and vice versa, but there has not been time to implement this idea.

There are unresolved problems here. Clearly, in drawings such as Figure B.44, there are edges which are almost the same length but which, because of the topology, cannot be exactly the same length. It should, in principle, be possible to deduce this given the knowledge already available.

The extent to which logical reasoning can be used to make deductions about face distance relationships from accepted edge ratio constraints, and vice versa, has not been investigated—there has not been time. Clearly, for example, face coplanarity and edge length equality are related (see, for example, Figure B.91), so there are potentially useful deductions to be made.

It is not clear how much logical reasoning is required. For example, it might seem obvious that if two edges are parallel and join the same pair of parallel faces then the edges must be the same length. However, this conclusion is not helpful—RIBALD's existing mechanisms already handle it implicitly, as the numerical error for the constraint is inevitably zero.

Interestingly, it is when identifying edge length ratio constraints that RIBALD finally realises that Figure B.146 is erroneous—two of the edges *must* be zero-length.

11.9.3 Non-Trihedral Vertices

Currently, RIBALD stores a list of accepted non-trihedral vertex constraints, listing which three faces are to be used as the reference faces and which other face distances are derived from these.

Non-trihedral vertex constraints, if present, must be enforced first.

Even ignoring the resolvable representation problem, there are limits on which

three faces can be used as the reference faces—two of the faces meeting at a non-trihedral vertex may be coplanar, or three might meet along a common line. It seems to be a practical necessity to choose the three "most orthogonal" (e.g. largest volume product of normals) as the basis set from which the vertex location is calculated and derive the face distances for other faces at the vertex from that.

The extent to which logical reasoning can be used to make deductions about face distance relationships and edge ratio constraints from accepted non-trihedral vertex constraints has not been investigated—there has not been time.

11.9.4 Linear Degrees of Freedom

There has not been time to address the linear degrees of freedom problem. It is suggested that a stochastic approach similar to that described for face normals in Section 11.5.3 would in practice be acceptable, but I have no test results to support or disprove this.

11.10 Intersecting Faces

The final stage of Grimstead's system [38] is a three-dimensional tidying process in which the x-, y- and z-coordinates of each vertex are recalculated from the equations of the three faces on which it lies.

This requires minor adaptation when non-trihedral vertices are allowed. Although it is to be hoped that intersecting any three non-coplanar faces on which the vertex lies will give the same coordinates, this cannot be guaranteed in view of the resolvable representation problem described in Section 11.2.2. The most robust method of those tested, as when processing non-trihedral vertex constraints, is to choose the three faces whose unit normals have the largest volume product.

11.11 Special Classes

For some of the more complex test cases, the general-case optimisation process described in previous sections is too slow to be considered interactive. There are also theoretical concerns about the general-case algorithm for face normals and serious doubts about general solutions to the face distance problem. Object classification (Chapter 9) is therefore used to take short cuts through the process for some classes of object.

Special-case methods for normals and distances are inevitably faster, and are also demonstrably more robust. These advantages may outweigh the disadvantages of special-case methods for commonly-occurring classes.

11.11.1 Normalons

For normalons, three perpendicular axes are formed, as close as possible to the average values of the appropriate face normals (which at this stage are not necessarily perpendicular), and each face normal is constrained to the appropriate axis. The general-case face normal process is bypassed, but mirror and rotation constraints are still generated for use during distance optimisation—this is required, for example, in order to ensure that the sides of the T-block (Figure B.34 etc) are the same height and length.

In normalons, each face normal should be aligned with the appropriate orthogonal axis. 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 $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}$
- \bullet reorder $(\hat{\mathbf{A}},\hat{\mathbf{B}},\hat{\mathbf{C}})$ if necessary to form a right-handed coordinate system
- iterate

$$\begin{split} &-\text{ set }\hat{\mathbf{A}}_{\mathbf{i+1}}=\hat{\ }(\hat{\mathbf{B}}_{\mathbf{i}}\times\hat{\mathbf{C}}_{\mathbf{i}})\\ &-\text{ set }\hat{\mathbf{B}}_{\mathbf{i+1}}=\hat{\ }(\hat{\mathbf{C}}_{\mathbf{i}}\times\hat{\mathbf{A}}_{\mathbf{i}}) \end{split}$$

$$- \text{ set } \mathbf{\hat{C}_{i+1}} = \hat{\ } (\mathbf{\hat{A}_i} \times \mathbf{\hat{B}_i})$$

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

The assumption of axis alignment has no effect on face distances, which are optimised using general-case methods.

11.11.2 Semi-Normalons

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

For semi-normalons, three perpendicular axes are formed, as for normalons (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). Each face normal which should be axis-aligned is constrained to the appropriate axis. No part of the general-case process is bypassed—optimisation of face normals takes place as for the general case, in order that relationships between non-axis-aligned faces can be established, but will be quicker by virtue of the knowledge already gained concerning parallelism and perpendicularity of axis-aligned faces and the relationships between non-axis-aligned faces and the mirror plane.

All constraint types listed for the general case are still generated, including parallelism and perpendicularity (to allow for the possibility that the sketch represents a normalon but was not identified as such because of sketching inaccuracies). All face distances are computed using the general-case face distance method.

11.11.3 Semi-Normalons with Mirror Symmetry

Semi-normalons with mirror symmetry follow the same route as those without mirror symmetry, but extra information is deduced from the mirror plane 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 [172] identifies this as one of four possibilities, listed in Table 11.1 using the notation of Chapter 5.7). The logical datasets are preset with this information.

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

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 11.1: Semi-Axis-Aligned Mirror Planes

11.11.4 Right Extrusions

If the object is believed to be a right extrusion, the end caps are made parallel and the sides are made perpendicular to the end caps. 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 front end face is believed to have mirror or rotational symmetry, the face normals of the sides are adjusted to preserve this symmetry. Since there are no further constraints on face distances, no further constraints are generated for extrusions, and the general-case process is bypassed entirely. Otherwise, the general-case method is used for determining face normals, but with a significantly reduced number of 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 [63] 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 impose plausible symmetries and regularities to generate the final normals. Vertices are also moved, as appropriate, to make the face being deskewed either axis-aligned or semi-axis-aligned, and to enforce any appropriate symmetry operations. This idea was considered but rejected as adding yet another special case. However, as this idea may also be useful for semi-normalons, it could be revived.

RIBALD determines the aspect ratio (and all other face distances) by the generalcase distance optimisation method. 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.

11.11.5 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.

11.11.6 Platonic and Archimedean Objects

The general-case method for finalising geometry is particularly slow for Platonic and Archimedean objects, 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 simple 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, it is recommended 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.

11.11.7 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) for semi-normalons 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 class would require new methods. We believe that normalons and semi-normalons are common in engineering

practice, and a survey supports this view [143], but this is not a guarantee that these will be a common feature of the sketches input by any particular user.

11.12 Results

The test results presented here concentrate on timing, as this is the major problem experienced with previous geometric CSP solvers such as those described in Section 11.2. Rather than provide a comprehensive survey, I analyse in more detail the results of fitting geometry to nine drawings, chosen from the most complex objects for which RIBALD can reliably reproduce the desired topology in order to test the methods of this chapter in conditions as close as possible to real-life use. In each of the following sections, the left-hand drawing shows the initial line drawing, the middle drawing shows the output of topological reconstruction, and the right-hand drawing shows the output of face normal optimisation.

Although the number of face distance constraints identified is listed for each object, the effects of trying to enforce them are not shown as there has not been time to complete implementation of this. Results of a more limited attempt to adjust face distances, placing visible vertices as close as possible to their locations in the original line drawing while enforcing mirror symmetry, have already been published [174] and are not repeated here.

11.12.1 Normalon

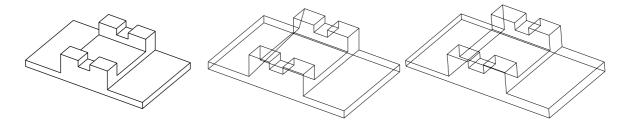


Figure 11.3: Normalon

The original drawing on the left of Figure 11.3 has 49 lines. RIBALD identifies 48 face normal constraints (which are ignored) and 3411 possible constraints on face distances.

Fixing face normals using downhill optimisation takes 0.11 seconds regardless of which RIBALD options are selected, as normalons are treated as a special case.

The need for face distance constraints (either edge equality or face coplanarity) is shown clearly by the right-hand figure, where the wings of the object are clearly not the same thickness.

11.12.2 Grimstead's Bracket

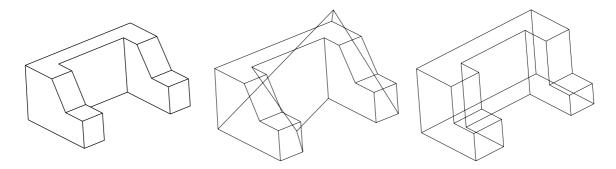


Figure 11.4: Grimstead's Bracket [38]

Grimstead [38] used the left-hand drawing in Figure 11.4 as the final test of his ideas. Topological completion is straightforward, but the geometry of the topological completion is visibly in need of correction. The original drawing has 31 lines. RIBALD identifies 472 possible constraints on face normals and 1263 possible constraints on face distances. Of the face normal constraints, 185 are actively considered (with 81 being accepted and 104 being rejected) before all angular degrees of freedom are removed. Of these, 181 can be accepted or rejected using logical reasoning, leaving 4 which require numerical processing.

Fixing face normals using downhill optimisation takes 0.04 seconds (0.12 seconds if logical reasoning is not used). Fixing face normals using geometric optimisation takes 0.05 seconds (0.14 seconds if logical reasoning is not used). Since timings depend on the provisional geometry output by topological reconstruction, they are particularly sensitive to small changes (such as adjustment of tuning constants) in topological reconstruction.

It can be seen that constraint enforcement has been over-enthusiastic, making faces perpendicular which should not be.

Preliminary timings suggest that individual distance constraint enforcement without a domain-specific knowledge base is too slow, but that all other options are adequate.

11.12.3 Semi-Axis-Aligned

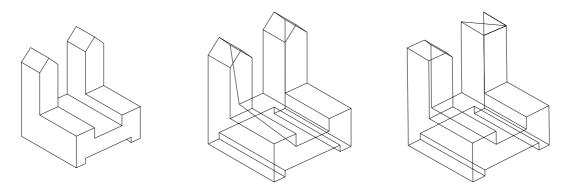


Figure 11.5: Semi-Axis-Aligned Object [194]

The original drawing on the left of Figure 11.5 has 38 lines. RIBALD identifies 1038 possible constraints on face normals and 2694 possible constraints on face distances. Of the face normal constraints, 346 are actively considered (with 150 being accepted and 196 being rejected) before all angular degrees of freedom are removed. All of these were accepted or rejected using logical reasoning—no numerical processing was required.

It is evident from the right-hand drawing that a "bad" constraint has been enforced—a quadrilateral face at the top of the right column has collapsed, with two opposed edges now being collinear. The source of this error has not been identified, but it appears that constraints enforcing mirror symmetry, and constraints enforcing parallelism between non-axis-aligned faces, are undervalued in comparison with constraints enforcing axis-alignment.

Furthermore, the regularity of the left-hand column has also been lost, showing that local mirror planes which do not propagate across the entire object are also undervalued in this case.

Fixing face normals using downhill optimisation takes 0.14 seconds (0.40 seconds if logical reasoning is not used). Fixing face normals using geometric optimisation takes 0.14 seconds (0.28 seconds if logical reasoning is not used).

11.12.4 Semi-Axis-Aligned

The original drawing on the left of Figure 11.6 has 39 lines; it is included here for comparison purposes (the next drawing is similar but without the mirror symmetry).

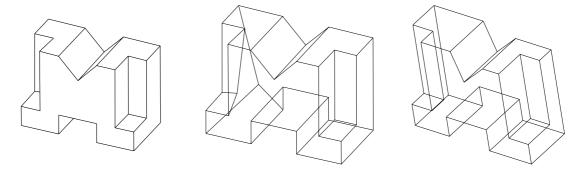


Figure 11.6: Semi-Axis-Aligned Object

RIBALD identifies 895 possible constraints on face normals and 2551 possible constraints on face distances. Of the face normal constraints, 327 are actively considered (with 177 being accepted and 150 being rejected) before all angular degrees of freedom are removed. Of these, 321 were accepted or rejected using logical reasoning, and one was accepted because it fit the existing geometry, leaving 5 which require numerical processing.

Fixing face normals using downhill optimisation takes 0.12 seconds (0.27 seconds if logical reasoning is not used). Fixing face normals using geometric optimisation takes 1.10 seconds (approximately 140 seconds if logical reasoning is not used).

Again, the need for face coplanarity or edge length equality constraints can be seen: the output of face normal optimisation has lost geometric mirror symmetry.

11.12.5 Semi-Axis-Aligned

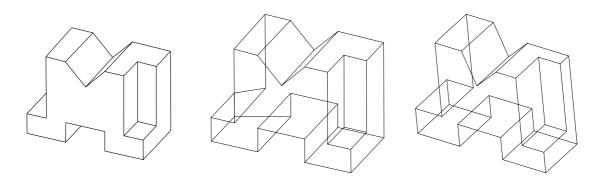


Figure 11.7: Semi-Axis-Aligned Object [194]

The original drawing on the left of Figure 11.7 has 35 lines. RIBALD identifies 751 possible constraints on face normals and 2257 possible constraints on face distances. Of the face normal constraints, 271 are actively considered (with 140 being

accepted and 131 being rejected) before all angular degrees of freedom are removed. Of these, 267 can be accepted or rejected using logical reasoning, leaving 4 which require numerical processing.

Fixing face normals using downhill optimisation takes 0.19 seconds (0.45 seconds if logical reasoning is not used). Fixing face normals using geometric optimisation takes 0.21 seconds (0.35 seconds if logical reasoning is not used). The need for face coplanarity constraints can again be seen.

11.12.6 Semi-Axis-Aligned

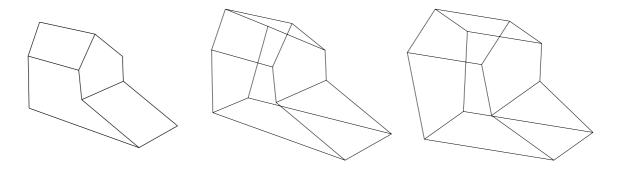


Figure 11.8: Semi-Axis-Aligned Object

The original drawing on the left of Figure 11.8 has 13 lines. The increased difficulty here is that there are two sets of deviations from axis-alignment. RIBALD identifies 155 possible constraints on face normals and 306 possible constraints on face distances. Of the face normal constraints, 49 are actively considered (with 19 being accepted and 30 being rejected) before all angular degrees of freedom are removed. Of these, 30 were accepted or rejected using logical reasoning, and none were accepted because they already fit the geometry, leaving 19 which require numerical processing.

It can be seen that in this example mirror symmetry has been enforced, but axis-alignment has not.

Fixing face normals using downhill optimisation takes 0.04 seconds (0.09 seconds if logical reasoning is not used). Fixing face normals using geometric optimisation takes 0.03 seconds (0.08 seconds if logical reasoning is not used).

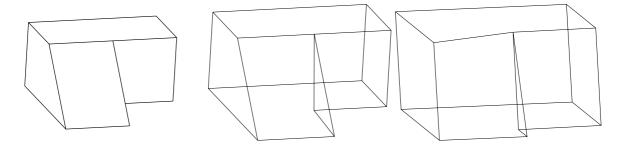


Figure 11.9: Non-Trihedral Object

11.12.7 Non-Trihedral, Convex

The original drawing on the left of Figure 11.9 has 11 lines. RIBALD identifies 153 possible constraints on face normals and 273 possible constraints on face distances. Of the face normal constraints, 51 are actively considered (with 20 being accepted and 31 being rejected) before all angular degrees of freedom are removed. Of these, 35 can be accepted or rejected using logical reasoning, leaving 16 which require numerical processing.

It is clear that the "finished" geometry is wrong, as one face is not planar, and the two collinear edges meeting at the *T*-junction are no longer collinear. This is thought to be an problem with the implementation in RIBALD (edges meeting at a vertex are somehow prevented from being collinear even when, as here, they should be) rather than anything inherent in the method.

Fixing face normals using downhill optimisation takes 0.03 seconds (0.10 seconds if logical reasoning is not used). Fixing face normals using geometric optimisation takes 0.03 seconds (0.09 seconds if logical reasoning is not used).

11.12.8 Non-Trihedral, Concave

The original drawing on the left of Figure 11.10 has 19 lines. RIBALD identifies 202 possible constraints on face normals and 507 possible constraints on face distances. Of the face normal constraints, 98 are actively considered (with 33 being accepted and 65 being rejected) before all angular degrees of freedom are removed. Of these, 78 can be accepted or rejected using logical reasoning, and 3 can be accepted because they match the existing geometry, leaving 17 which require numerical processing.

As with the previous example, the two collinear edges meeting at the T-junction are no longer collinear.

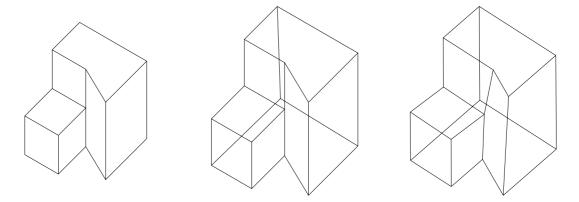


Figure 11.10: Non-Trihedral Object

Fixing face normals using downhill optimisation takes 0.05 seconds (0.18 seconds if logical reasoning is not used). Fixing face normals using geometric optimisation takes 0.07 seconds (0.17 seconds if logical reasoning is not used).

11.12.9 Semi-Axis-Aligned

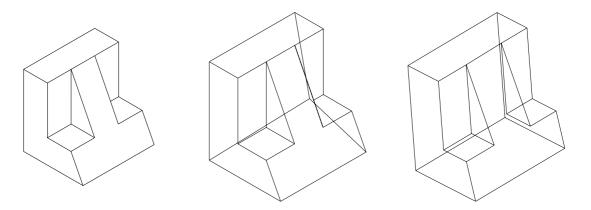


Figure 11.11: Non-Trihedral Object [194]

The original drawing on the left of Figure 11.11 has 21 lines. RIBALD identifies 313 possible constraints on face normals and 760 possible constraints on face distances. Of the face normal constraints, 108 are actively considered (with 38 being accepted and 70 being rejected) before all angular degrees of freedom are removed. Of these, 106 were accepted or rejected using logical reasoning, and none were accepted because they already fit the geometry, leaving 2 which required numerical processing.

Fixing face normals using downhill optimisation takes 0.03 seconds (0.09 seconds if logical reasoning is not used). Fixing face normals using geometric optimisation

takes 0.03 seconds (0.06 seconds if logical reasoning is not used).

11.12.10 Summary

The results presented here suggest that further work is required before the methods described in this chapter can be considered robust. However, the only method which is provably inadequate is determination of angular degrees of freedom (Sections 11.5.3 and 11.5.4).

The faults in Figures 11.9 and 11.10 result from an implementation problem which could easily be corrected given more time.

Figure 11.11 can be considered satisfactory. The angle between the slanting face and the others has changed visibly from the original line drawing—this results from treating vertex z-coordinates as being equally as valid as vertex x- and y-coordinates when generating the three main axes of a semi-normalon from the output of inflation (Chapter 7). Changing this would be straightforward, but improvements in inflation would make such changes unnecessary.

Figure 11.8 (Section 11.12.6) illustrates a common problem. As shown, a correct mirror constraint is enforced in preference to (equally-correct) perpendicularity constraints. With small changes to numerical constants, the output can be changed so that perpendicularity constraints, including one between the two sloping roof faces, are enforced but mirror symmetry is not. Although it is obvious visually that the two constraints do not conflict, the current (inadequate) algorithm for degrees of freedom only permits enforcement of the higher-merit constraint.

Sensitivity of this sort is observed frequently with more complex drawings. Other examples are Figure 11.4, where unconvincing perpendicularity constraints are enforced in preference to face angle constraints, Figure 11.5, where entirely erroneous perpendicularity constraints are enforced in preference to a variety of other, more valid, constraints, and Figures 11.6 and 11.7, where the V-notch at the top of the object is distorted because perpendicularity constraints are enforced in preference to face angle constraints. These last two examples highlight one omission from RIB-ALD, which is that if a face angle constraint is accepted, it becomes more plausible that the same face angle appears elsewhere in the object—other face angle constraints requiring the same face angle should be reassessed.

Many of the constants in this Chapter are arbitrary—there was insufficient time to optimise tuning constants, unlike in other Chapters. It is possible that an optimal set of tuning constants would have overcome many of the problems noted in this Section. It is also possible, however, that no fully satisfactory set of tuning constants exists, and if this proves to be the case, the decision to separate face normal and face distance optimisation may need to be reconsidered (face distances could provide useful clues to the merit of face normal constraints).

Assuming RIBALD implements correctly the ideas of Section 11.5 (this assumption is not necessarily good), Figure 11.8 also provides evidence that better solutions are required to the problem of angular degrees of freedom. Clearly, it is possible to enforce *both* mirror and perpendicularity constraints, but RIBALD's method of determining angular degrees of freedom does not allow for this.

Timings are, in general, acceptable, but this situation is also not robust. Unacceptable timings such as those with Figure 11.6 (Section 11.12.4) occur sporadically and, as yet, unpredictably.

11.12.11 Face Distance Constraints

As RIBALD's implementation of face distance constraints is incomplete, no results are presented. One preliminary observation concerning edge length equality constraints may however be worth noting.

In Figure B.503 a false constraint shows up quite early on (in amongst the ones which can be accepted automatically and well before many desirable ones), constraining the slanting edge at the front of the object to be the same length as the base of the object. Giving edge length constraints between parallel edges higher merit than edge length constraints between non-parallel edges solves the problem in this case and leads to a good geometry. However, this is not necessarily a good general solution, as part of the purpose of edge length constraints is to make near-cubes cubic, which necessitates enforcing them between non-parallel edges.

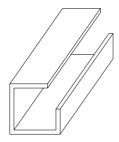
Chapter 12

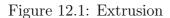
Results

Previous chapters included analyses of the various components based on their performance in processing the entire set of test drawings in Appendix B. However, the motivation behind this work is the interpretation of line drawings of engineering objects. This chapter selects ten "typical engineering objects" from the test drawings and analyses how well RIBALD reconstructs the topology of the corresponding solid objects. Geometric fitting (see Chapter 11) is not discussed here as work on this did not reach a satisfactory conclusion in the time available. The chapter concludes with general remarks about accuracy of interpretation and timings.

12.1 Axis-Aligned Extrusion

Figure 12.1 is the most difficult axis-aligned extrusion to process, as it includes a line both ends of which are T-junctions. The drawing comprises 22 lines.





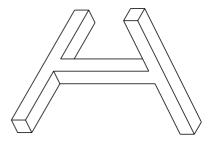


Figure 12.2: Axis-Aligned

Set-intersection methods (see Chapter 4) always label this drawing correctly; relaxation methods label the drawing correctly provided that the full catalogue is

used for all junction types including L-junctions, W-junctions and Y-junctions (why this should make a difference here is unclear).

Bundling of parallel lines (see Chapter 5) always operates correctly, producing three bundles of lines.

RIBALD finds no cofacial configurations or slot features (see Chapter 6) in this drawing; this is correct.

Inflation (see Chapter 7) produces correct depth ordering for neighbouring vertices if and only if the options to generate depth equations from bundles of parallel lines and to use JLP rather than corner orthogonality are selected. Without using parallel line information, the direction of the double-T-junction line cannot be determined. The assumptions behind corner orthogonality do not apply to this projection, and it fails for those vertices which do not meet the Perkins criteria.

For each quadrilateral face, RIBALD identifies potential mirror symmetries (see Chapter 8) from edge to edge (merit 0.97) and from vertex to vertex (merit 0.40). The dominant reflection planes (merit 0.63) are the two obtained from chaining pairs of single-face mirror symmetries along the sides of the extrusion; all other candidate reflection planes have very low merit.

Attempts to find candidate rotation axes illustrate problems also found with several other objects considered in this Chapter. Firstly, the object has no rotational symmetry. Secondly, for the two faces for which C_2 and C_4 candidate symmetries are correctly found, the merit figures (0.80 for C_2 and 0.90 for C_4) are far too high, particularly that for the C_4 rotation as the faces are visibly not square. Thirdly, the two side faces which occlude T-junctions are incorrectly categorised as pentagonal, and considered as candidates for C_5 rotations.

RIBALD classifies (see Chapter 9) the object as a trihedral normalon extrusion. The merit figures for the object being a normalon and it being an extrusion are both 1.00, and merit figures for any alternatives are all 0.00.

Having identified that the object is an extrusion and which face is the end-cap, RIBALD reconstructs the complete object topology correctly.

12.2 Axis-Aligned Non-Extrusion

Figure 12.2 shows two perpendicular beams joined by a cross-beam. The drawing comprises 25 lines.

All labelling variants label this drawing correctly. Bundling of parallel lines always operates correctly, producing three bundles of lines.

RIBALD finds no cofacial configurations or slot features in this drawing; this is correct.

If depth equations are generated from bundles of parallel lines, inflation always produces correct ordering of adjacent vertices regardless of which other options are chosen. If parallelism equations are not used, there are always errors, with one or both of the lines ending at T-junctions being misdirected.

RIBALD identifies face mirror planes as before. It correctly identifies both major planes of reflection of the object; one, starting at the front and crossing four faces, has merit 0.87, and the other, running along the front of the object and crossing two faces, has merit 0.29 (this seems somewhat low).

Candidate C_2 and C_4 rotational symmetry axes are identified for three faces. The C_2 merits are all close to 0.80; the C_4 merits are 0.99 for the two small squarish faces and 0.90 for the (clearly non-square) fully-visible rectangular face; this last figure is clearly too high.

RIBALD classifies the object as a trihedral normalon (merit 1.00), with the merit figures for all other special classes being 0.00.

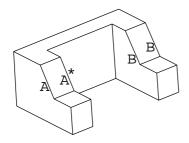
With the optimal set of tuning constants for topological reconstruction, RIB-ALD produces the correct topology of the desired object. However, optimising the tuning constants took time, and during development, reconstruction was sometimes marred by the T-piece problem discussed in Chapter 10.9.2. The correct topology is symmetrical about both of the two major planes of reflection; the valid but incorrect topology only about the first of them.

12.3 Grimstead's Block

Figure 12.3 is Grimstead's test drawing [38]. Although apparently complex (the drawing comprises 31 lines), interpretation should present little difficulty as there

are only two hidden vertices to find and their topology and geometry should be obvious.

Set-intersection methods label this drawing correctly. Relaxation methods sometimes fail (labelling the line indicated "*" as concave rather than occluding), depending on the initial values used and the number of iterations allowed.



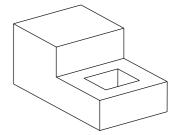


Figure 12.3: Grimstead's Block

Figure 12.4: Hole Loop

Continuing with a correctly-labelled drawing, bundling of parallel lines normally results in four bundles, but using the "strict" option produces five: the two pairs of lines "A" and "B" in the diagram are not bundled together. These lines are not quite parallel in the drawing, the clue which suggests that they ought to be parallel is the mirror symmetry of the object, and object symmetry is determined after bundling of parallel lines.

RIBALD finds no cofacial configurations or slot features in this drawing; this is correct.

Inflation produces correct depth ordering of adjacent vertices if either of two options is selected: parallelism equations from bundles, or inclusion of entries for lines terminating in T-junctions in the JLP tables. Failing this, the line terminating in T-junction is usually misdirected.

RIBALD identifies face mirror planes as before. It correctly identifies that the dominant plane of reflection (merit 0.73) is that cutting the top of the object, although this crosses only one face; the two chains of three mirror planes crossing the arms of the object have merit figures of less than 0.01.

Candidate axes of rotation are identified for the two squarish front faces, with merit 0.79 for C_2 and 0.64 for C_4 ; these are again too high as these rotations provide no clue to the topology of the object. There is also an erroneous candidate C_5 rotation axis for the face which occludes the T-junction.

RIBALD classifies the object as a trihedral semi-normalon with mirror symmetry (merit 0.69); as a result of this, the merit of the dominant plane of reflection is increased to 0.92.

With the optimal set of tuning constants for topological reconstruction, RIBALD produces the correct topology of the desired object. As with the previous example, a problem encountered during development (in this case, a valid topology with a "step" at the bottom of the left-hand end of the bracket) does not appear with the tuned version of the program. The correct topology is symmetrical about the reflection plane; the valid but incorrect one is not.

12.4 Hole Loop

Figure 12.4 is a simple drawing including a hole loop, portraying an L-block with a through hole. The drawing comprises 20 lines.

All labelling variants label this drawing correctly. Bundling of parallel lines always operates correctly, producing three bundles of lines.

RIBALD identifies that the cofacial configuration in this drawing corresponds to a hole or pocket, and that there are no slot features; this is correct.

Inflation produces correct depth ordering of adjacent vertices if either of two options is selected: parallelism equations from bundles, or generation of equations to place the occluded lines at T-junctions a fixed distance behind the occluding lines. Failing this, the line down into the hole terminating in a T-junction is misdirected.

RIBALD identifies face mirror planes as before, and also a vertex-to-vertex mirror plane (merit 0.80) crossing the L-shaped face. It correctly identifies that the dominant plane of reflection (merit 0.79) is that crossing the top four faces of the object, with the plane formed by chaining the concave edge with the vertex-to-vertex mirror plane being another plausible candidate (merit 0.38).

Candidate axes of rotation are identified for the two squarish faces at the top and front of the object, with merit 0.79 for C_2 and 0.67–0.71 for C_4 .

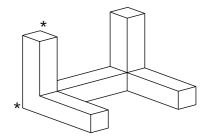
RIBALD classifies the object as a trihedral normalon (merit 0.998); the merit figures for other special classes, including "extrusion", are 0.00.

Topological reconstruction illustrates a known limitation of RIBALD. RIBALD cannot reconstruct through holes (determination of which rear face or faces are

penetrated by a through hole is left for future research), so this drawing's feature is reconstructed as a pocket. Furthermore, as the bottom of the pocket is not visible in the drawing, the depth of the pocket is arbitrary. The resulting object is topologically correct, but not the best interpretation of the drawing: if the feature were a pocket rather than a hole, the most informative viewpoint would be one which showed the pocket bottom.

12.5 Extended Trihedral Normalon

Figure 12.5 illustrates an extended trihedral normalon, two L-blocks joined by a cross-beam. The drawing comprises 31 lines.



* ?

Figure 12.5: Extended Trihedral Normalon

Figure 12.6: Extended Trihedral Semi-Normalon

All labelling variants label this drawing correctly (when asked to use trihedral labelling, RIBALD automatically uses extended trihedral because of the presence of tetrahedral and hexahedral junctions).

Bundling of parallel lines always operates correctly, producing three bundles of lines.

RIBALD finds no cofacial configurations or slot features in this drawing; this is correct.

If bundles of parallel lines are used to generate equations, inflation always produces correct depth ordering of neighbouring vertices. With other combinations of options, RIBALD sometimes achieves correct results but more often does not—lines terminating at T-junctions are sometimes misdirected, as are lines terminating at the non-trihedral junctions.

RIBALD identifies face mirror planes as before, and also a vertex-to-vertex mirror plane (merit 0.80) crossing the fully-visible L-shaped face. It finds no dominant plane of reflection, the best (merit 0.18) being that formed by chaining the long concave/convex/concave edge with the vertex-to-vertex mirror plane; two others (merit 0.06), each formed by chaining four faces along the tops of the Ls, are evaluated as inferior even to this.

Candidate axes of rotation are identified for the four squarish faces at the tops and fronts of the Ls, with merit 0.80 for C_2 and 0.42–0.52 for C_4 . There is also two erroneous candidate C_5 rotation axes for the faces which occlude the T-junctions.

RIBALD classifies the object as a non-trihedral normalon (merit 0.999); the merit figures for other special classes are 0.00.

Despite the apparent simplicity of this drawing, RIBALD does not reconstruct the topology of the object correctly. In reconstructing the vertex/edge framework, it starts by mistakenly adding a new vertex and edges connecting it to the two *L*-junctions marked "*". This is not detected as erroneous because, having made this mistake, it nevertheless manages to find a self-consistent topology by adding further edges to link the various incomplete vertices. The resulting topology is certainly not axis-aligned and appears to be impossible to interpret geometrically, but examined purely as a topology it is valid, so backtracking is not invoked.

It is not clear that reconstructing the object using face planes, rather than by reconstructing the vertex/edge framework first, would avoid the initial error—the two "*" L-junctions clearly lie on the same face, and vertices must be added somewhere to complete this face. However, it is clear that analysis of face planes would detect that the topology RIBALD actually obtains is incorrect.

Even using the methods of this thesis, it should be possible to reject the topology obtained on the grounds that it conflicts with the strong (merit 0.999) assumption that the object is a normalon (it also conflicts with the two planes of reflection of the object, but these have much lower merit). The clues to this invalidity are not obvious (for example, in the object obtained, all trihedral vertices have one edge allocated to each bundle, as would be required of a normalon), and RIBALD does not as yet include this refinement.

12.6 Non-Trihedral Semi-Normalon

Figure 12.6 illustrates an extended trihedral semi-normalon resembling a bookshelf. The drawing comprises 17 lines. Interpretation ought to be straightforward—like Grimstead's block, there are only two hidden vertices to be deduced, and their topology and geometry should be obvious, and (also like Grimstead's block) the object's mirror symmetry should act as supporting evidence for the correct interpretation.

None of the labelling methods tried labelled this drawing correctly. All methods tried produce label convex lines as concave and vice versa for the three lines marked "?". Set intersection labels the line marked "*" as convex and relaxation methods label it as occluding (it should be concave). Both interpretations are incorrect—in particular, the result from set intersection is a geometrically possible interpretation but ignores the obvious symmetry of the object. The problem occurs because the heuristic which minimises the number of types of non-trihedral vertex in the final object cannot distinguish between the 3-convex+1-concave K-junctions obtained by RIBALD and the 3-convex+1-concave K-junctions in the correct labelling.

Continuing with a correctly-labelled drawing produced by hand, bundling of parallel lines always operates correctly, producing four bundles of lines.

RIBALD finds no cofacial configurations or slot features in this drawing; this is correct.

Depth ordering of neighbouring vertices is erratic. Using the two options (a) to use JLP and (b) to generate equations from bundling information, RIBALD obtains a correct depth ordering more often than not, depending on which other options are selected; if either of these options is not selected, RIBALD only occasionally obtains a correct depth ordering. JLP is required because corner orthogonality does not cope so well with corners which are not axis-aligned. Bundling is required in order to generate correct orientation both for the line terminating at a T-junction and also for the other two lines meeting the concave trihedral junction (although it is visually obvious which direction a Kcdcc-Yddd line takes, such lines are so uncommon in practice that no entry was included for this in the JLP tables). There is a further problem which makes JLP unreliable here. In an isometric projection, ends of the diagonal lines would be equidistant from the viewer. The projection here is not quite isometric, and the right-hand ends of the lines genuinely are slightly closer to

the viewer, but JLP, being based on labellings and the assumption of isometricity, does not enforce this with any weight and indeed generates equations to make the ends of symmetrical lines (Yccc-Yccc etc) equidistant.

RIBALD identifies face mirror planes as before, and also four vertex-to-edge mirror planes, one (merit 0.90) crossing the pentagonal end face and three (all merit 0.75) crossing the fully-visible internal triangular face. Two planes of reflection, formed by chaining a vertex-to-edge mirror plane with an edge-to-edge mirror plane, are regarded as almost certain (merit figures 0.997 and 0.993)—although the finished object is topologically symmetrical about this plane, it is not clear from the drawing that it should be geometrically symmetrical, so these merit figures are somewhat high. The incomplete internal faces prevent the two face mirror planes detected along the true plane of rotation from being chained together; these, separately, each have merit 0.62.

Candidate axes of rotation are identified for the two rectangular faces at the top and front of the object, with merit 0.80 for C_2 and 0.59 for C_4 ; these are, as before, too high, as is the merit figure (0.70) for a C_5 rotation of the pentagonal end face.

RIBALD classifies the object as a tetrahedral semi-normalon with mirror symmetry (merit 0.73); as a result of this, the merit of the assumed dominant plane of reflection is increased to 0.999.

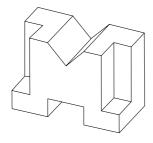
As with the previous object, a poor choice of first move results in an incorrect (and apparently invalid) topology.

12.7 Semi-Normalon

Figure 12.7 is a mirror-symmetric semi-normalon, adapted from [194] (the original is not mirror-symmetric). The drawing comprises 39 lines. Analysis of this drawing is confused by the presence of an underslot feature (see Chapter 6) with two reasonable interpretations.

Set-intersection methods always label this drawing correctly; relaxation methods label the drawing correctly provided that the full catalogue is used for all junction types including L-junctions, W-junctions and Y-junctions.

Bundling of parallel lines always operates correctly, producing five bundles of lines.



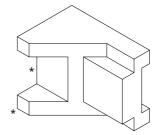


Figure 12.7: Semi-Normalon

Figure 12.8: Semi-Normalon

RIBALD correctly identifies the or slot features on the underside of the object portrayed in this drawing, and also that there are no cofacial loops.

Inflation produces correct depth ordering of adjacent vertices if both of two options are selected: parallelism equations from bundles, and omission of equations to place the occluded lines at T-junctions a fixed distance behind the occluding lines. Failing this, the line towards the left of the drawing from a Y-junction to a T-junction is frequently misdirected, and other lines are also occasionally misdirected.

RIBALD identifies face mirror planes as before, and additionally two others, a vertex-to-edge mirror (merit 0.86) corresponding to the true reflection plane of the object, and a vertex-to-vertex mirror (merit 0.77) vertically down the concave edge at the right-hand front of the object. It recognises that the vertex-to-edge mirror corresponds to the dominant reflection plane of the object; the only mirror planes which can be chained (the two crossing the indentation at the top of the object) together have a negligible merit as this cannot be propagated in either direction.

There are no candidate axes of rotation.

RIBALD classifies the object as a trihedral semi-normalon with mirror symmetry (merit 0.77); as a result of this, the merit of the dominant plane of reflection is increased to 0.94.

RIBALD reconstructs a valid and sensible topology for the object, albeit not quite the best interpretation of the drawing. The natural interpretation of the drawing would be that, as the slot on the top of the object runs all the way from front to back, the slot on the underside of the object should do too; in the object actually produced, the slot terminates mid-way through the object, and the rear face touches the "ground" along its entire length. During development, with different values of topological tuning constants, it has been possible to produce the preferred

interpretation.

12.8 Semi-Normalon

Figure 12.8 is a mirror-symmetric semi-normalon taken from a drawing exercise [194]. The drawing comprises 39 lines. Despite the different appearance of the two drawings, the object portrayed here is topologically close to that in the previous example, and similar results could be expected. Again, the underslot feature presents the main problem of interpretation.

Set-intersection methods always label this drawing correctly; relaxation methods label the drawing correctly provided that the full catalogue is used for all junction types including L-junctions, W-junctions and Y-junctions.

The "lenient" versions of bundling correctly produce four bundles of parallel lines; the "normal" and "strict" versions produce five. The short line at the bottom of the drawing, ending in a T-junction, is not drawn correctly (if it were, there would be an accidental coincidence of lines). The topological clue that it should be bundled with other similarly- oriented lines is the mirror symmetry of the object, which has not been determined at this stage. In the absence of this clue, only the "lenient" version of bundling allows a wide-enough range of orientations for it to be included in the bundle.

RIBALD correctly identifies the or slot features on the rear of the object portrayed in this drawing (feature detection ignores orientation), and also that there are no cofacial loops.

As with Figure 12.6, depth ordering of neighbouring vertices is erratic. Using the two options (a) to use JLP and (b) to generate equations from bundling information, RIBALD obtains a correct depth ordering more often than not, depending on which other options are selected; if bundling is not selected, RIBALD only occasionally obtains a correct depth ordering, and if corner orthogonality is selected in place of JLP, RIBALD never obtains a correct depth ordering. Again, corner orthogonality does not cope well with corners which are not axis-aligned. Bundling information is required in order to direct correctly lines terminating in T-junctions, but with the "normal" bundling options one of these lines has not been bundled properly, so some additional means of directing this line is required.

RIBALD identifies face mirror planes as before, and additionally one low-merit (0.28) vertex-to-vertex mirror plane crossing the quadrilateral face with two concave and two convex edges. It finds two plausible planes of reflection by chaining pairs of edge-to-edge mirror planes, one (merit 0.50), the true one, at the front of the object (at the right of the drawing), and the other (merit 0.25) from the two faces at the bottom left in the drawing.

Candidate axes of rotation are identified for the two small faces at the top of the object (at the left of the drawing), with merit 0.69–0.72 for C_2 and 0.69–0.75 for C_4 . There is also one erroneous candidate C_5 rotation axes for a faces which occludes a T-junction.

RIBALD classifies the object as a trihedral semi-normalon (merit 0.35) rather than a trihedral semi-normalon with mirror symmetry (merit 0.28); the merit of the (correct) plane of reflection is increased to 0.64.

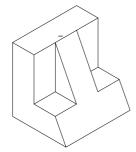
RIBALD produces an incorrect but apparently valid topology for this object. After completing the slot feature as a first move, a very poor choice of second move (reversing the direction of the line marked "*" so as to complete a quadrilateral face by adding an edge meeting the vertex marked "*") means that even after sensible additions on subsequent moves, RIBALD cannot reach an object with a plane of reflection. The resulting topology is self-consistent and even meets the requirements of Euler's formula, but it is difficult to see how a consistent geometry could be fitted to it.

With hand-chosen values of topology tuning constants, it has been possible to produce the correct interpretation, but these are not the values which produce optimal results over the entire set of test drawings.

12.9 Non-Trihedral Semi-Normalon

Figure 12.9 appeared in Chapter 1.1 as an illustration of a simple line drawing, the interpretation of which is straightforward to anyone from an engineering background. The drawing comprises 21 lines.

No set-intersection labelling method labels this drawing correctly—the line marked "-" is labelled as concave when it should clearly be convex (it does not seem possible to fit a frontal geometry to the drawing in which the line is concave). Relaxation



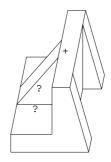


Figure 12.9: Non-Trihedral Semi-Normalon

Figure 12.10: Non-Trihedral Bracket

methods label the drawing correctly provided that the full catalogue is used for all junction types including L-junctions, W-junctions and Y-junctions.

Continuing with a correctly-labelled drawing, bundling of parallel lines always operates correctly, producing four bundles of lines.

RIBALD finds no cofacial configurations or slot features in this drawing; this is correct.

Inflation produces correct depth ordering if and only if equations are generated from bundles of parallel lines and equations are not generated to place the occluded lines at T-junctions a fixed distance behind the occluding lines. If either of these conditions is not met, the line terminating at a T-junction is misdirected. Other options do not affect the depth ordering.

It is worth remarking that the correct depth ordering is not visually obvious. The drawing is in isometric projection, and (although it may not appear so) the bottom-most junction in the drawing is further away from the viewer than the Yccc junction immediately above it. The closest junction to the viewer is the Yccc junction at the front of the top face.

RIBALD identifies face mirror planes as before, and additionally one vertex-to-vertex mirror plane (merit 0.25) across the L-shaped face and three vertex-to-edge mirror planes (merit 0.75) crossing the visible internal triangular face. The dominant plane of reflection (merit 0.82), formed by chaining two face mirror planes, is the true plane of reflection of the object; no other candidate has a merit greater than 0.01.

There are no candidate axes of rotation.

RIBALD classifies the object as a tetrahedral semi-normalon with mirror symmetry (merit 0.56); as a result of this, the merit of the dominant plane of reflection is increased to 0.92.

RIBALD reconstructs the expected topology for the object.

12.10 Non-Trihedral Bracket

Figure 12.10 illustrates a problem with the "most informative viewpoint" rule. The front of the object is not greatly different from those portrayed in Figures 12.6 and 12.9, but drawing a front view would leave no clues as to the topology of the back of the object. The drawing comprises 22 lines.

Set intersection produces a suboptimal but plausible labelling in which the line segment marked "+" is occluding rather than convex—this results from giving higher merit to trihedral than to non-trihedral interpretations. Relaxation labelling also does this and adds a further error: the lines marked "?" are convex when they should be concave and vice versa.

Continuing with a correctly-labelled drawing produced by hand, the "normal" and "strict" versions of bundling correctly produce five bundles of parallel lines. The "lenient" versions produce four, making incorrect groupings in doing so.

RIBALD finds no cofacial configurations or slot features in this drawing; this is correct.

Inflation usually produces correct depth ordering of neighbouring vertices if equations are generated from bundles of parallel lines (if they are not, there are always errors, for reasons similar to those noted with other drawings). The sole failure is when options are selected for corner orthogonality and occluded lines a fixed distance behind occluding at T-junctions; this results in the L-T line in the top left-hand corner of the drawing being misdirected.

RIBALD identifies face mirror planes as before, including three vertex-to-edge mirror planes (merit 0.75) crossing the visible triangular face. The only plausible plane of reflection found (merit 0.20) is the one corresponding to the true mirror symmetry of the object.

Candidate axes of rotation are identified for the two rectangular faces of the buttresses (at the right of the drawing), with merit 0.79–0.80 for C_2 and 0.63 for

 C_4 . These merit figures are, as with most other objects considered in this Chapter, too high.

RIBALD classifies the object as tetrahedral with "no special class" (merit 0.52) rather than as a semi-normalon (merit 0.33); as a result of this, the merit of the dominant plane of reflection is increased to 0.47.

Reconstruction of hidden topology produces an uncompletable object which causes filling-in of face loops (Chapter 2.14) to fail.

12.11 Conclusions

The set of drawings which can be interpreted plausibly as solid objects, and for which a topological model with provisional geometry can be produced, improves on Grimstead's method.

Objects which meet one of the special-case classes are in general classified correctly. More commonly, objects will meet the requirements of several of the special-case classes, and the class to which they are allotted can be arbitrary. This can also vary depending on how well-drawn the sketch is—different versions of Figure B.54 (page 310) are classified as a frustum, a semi-axis-aligned sketch with a mirror plane, or an extrusion.

12.12 Timing

As discussed in the previous chapters, all of the algorithms used in Chapters 4–10 are provably polynomial except for line labelling and topological reconstruction. Line labelling is polynomial if (and only if) there is a single sensible labelling. Topological reconstruction is polynomial if (and only if) the greedy approach finds a satisfactory solution.

On the whole, the system meets the goal of interactive response times. Most exceptions occur when a poor choice of move is made during the early stages of topological reconstruction. This can make the process very slow—of the order of several seconds or even minutes—before backtracking brings the system back to a more sensible choice.

Line labelling is less problematic—even the worst case of the slowest method (set

intersection) takes approximately 28 seconds, and this is exceptional. Most other difficult cases take a few seconds using the slowest method, and all cases take less than a second using relaxation methods.

For each of the drawings considered in this Chapter, the entire process from line labelling to topological reconstruction takes place in a time which could be considered interactive. In most cases, RIBALD takes longer to draw the full object than it takes to construct it.

Chapter 13

Conclusions

This chapter draws conclusions from the results presented in the preceding Chapters, and makes recommendations for future work.

In each of Chapters 4–11, the ideas presented advance the state of the art. In two cases, the generation and listing of the tetrahedral junction catalogue in Chapter 4.3 and Appendix E [175] and the elaboration and analysis of Junction Label Pairs presented in Chapter 7 [176], specific ideas have led to published papers, and the overall approaches to frontal geometry (Chapters 4–9) [172], hidden topology (Chapter 10) [173] and geometric fitting (Chapter 11) [174] have also been published as conference papers. These are summarised in Section 13.1. Within this area, several areas require further research, and these too are described in Section 13.1.

Conversion of sketches to line drawings is not the topic of this thesis, which assumes that such conversion is possible. This assumption is evaluated in Section 13.2 in the light of the results summarised in Chapter 12.

The most obvious deficiency of the ideas in this thesis is that they are restricted to polyhedra. Section 13.3 considers the merits of attempting to interpret curved line drawings.

Concerning future work related to, but outside the scope of, this thesis, Section 13.4 makes recommendations concerning features, and Section 13.5 lists (without attempting to answer) questions of psychology which have a bearing on line drawing interpretation.

13.1 Line Drawing Interpretation

In each of Chapters 4–11, the ideas presented advance the state of the art. In all cases there remain areas still to be resolved. In some cases, incremental improvements would be sufficient to produce reliable methods, but in other cases, new ideas are required.

In order to meet the requirement for interactive performance, a faster algorithm for propagating vertex, edge and face pairings (page 151) is required. This is required both for initial detection of local symmetry (Chapter 8) and for evaluating the merits, consequences and implications of those hypotheses in Chapters 10 and 11 which depend on local symmetry. $O(n^3)$ time should be possible in theory, but the algorithm I have reported in [178] is $O(n^4)$.

The line-labelling methods described in Chapter 4 improve on the state of the art, in that they not only label trihedral drawings correctly (as do many previous methods), but also (more often than not) label non-trihedral drawings correctly (a more difficult problem which previous methods do not attempt to solve). Nevertheless, they are insufficiently reliable: using relaxation labelling, the output is too often incorrect, and although set intersection labelling is somewhat more reliable, it still produces an incorrect labelling about 20% of the time, and it is unacceptably slow for drawings of 50 or more lines. These results seem to be approaching the limit of what is possible when line-labelling is treated purely as a combinatorial problem. As line labels (and in particular, the junction labels also produced as part of this process) are so useful, attempts should be made to overcome these problems. It is recommended that further investigation into line-labelling should start by investigating how geometric inferences can be incorporated into labelling algorithms (see Figure 4.42, page 88); attempts to take account of potential symmetries would also be useful (see Figure 4.43, page 88).

Bundling of parallel lines (Chapter 5) is more robust than Grimstead's bucketing [38] and more flexible than Sugihara's assumption [163] that edges are parallel in the object if and only if the corresponding lines are parallel in the drawing. Bundling appears to be reaching the limits of what is possible given the requirement of allowing for freehand drawing errors. There may be further inferences which can

be drawn which allow incorrect bundlings to be rejected, but such incremental improvements are unlikely to make a dramatic difference. It is recommended that, as in RIBALD, bundling of parallel lines is treated as a hypothesis, not as a fact.

The ordering of line-labelling and bundling of parallel lines was constrained by the use of line labels in bundling to reject impossible bundlings. However, it is clear that parallel line information could be of use in the labelling process. Where there are only three bundles of parallel lines, use of the extended trihedral catalogue rather than the full catalogue is clearly justified. It is also plausible that in drawings with four or more bundles of lines, junctions which use only the three primary bundles should be restricted to the extended trihedral catalogue—this may fail in some cases, but could be a useful heuristic and should be investigated.

The inflation methods described in Chapter 7 improve on the state of the art by adding the Junction Line Pair (JLP) compliance function. The methods described appear satisfactory. Even though the results obtained are far from perfect, the methods described are flexible as (a) the use of a linear system makes addition of extra compliance functions, or varying the weighting of existing compliance functions, easy, and (b) it is also easy to change weightings and add extra entries to the JLP tables. There is therefore room for significant improvement without any requirement for radically new ideas.

The ordering of line-labelling and inflation is constrained by the use of the JLP compliance function and by the intuitive requirement that one should attempt to gain topological information ("which lines are concave?") before trying to fit a geometry ("what is the dihedral angle?"). However, since the main problem encountered in line-labelling is the lack of geometric information, it is worth investigating how the two components may be combined. I suggest, as two possibilities, (a) interleaving iterations of relaxation labelling and inflation, and (b) using a genetic algorithm in which the genes determine inflation geometry, and fitness is assessed using line-labelling heuristics such as those in Chapter 4.

Reconstruction of hidden topology (Chapter 10) improves on Grimstead [38] by evaluating the merits of competing hypothesis rather than working through a fixed list of possible moves. As a result, a valid topology is produced for drawings for which Grimstead's method cannot even attempt to construct a topology, and the *correct* topology is produced for some drawings which Grimstead's method

produces implausible results. Nevertheless, topological reconstruction still presents serious unsolved problems. Although it is possible that reliability may be improved somewhat by further tuning, it is unlikely that this alone will be sufficient. Further ideas are needed, and two promising approaches are recommended. The first is that by treating face planes as half-spaces, and edges as half-space operators (union or intersection), obviously incorrect hypotheses can be rejected without further analysis. The second is that, since topological reconstruction is robust for simple objects, splitting a complex object into two or more simple ones and constructing the hidden topology separately for these would increase reliability (it should also increase speed).

The results of geometric finishing (Chapter 11) are inconclusive. The ideas presented are intuitively sound, but there is no experimental confirmation of their validity. This was due (a) partly to lack of time (tuning constants were not optimised), and (b) partly because the algorithm (presented in Chapter 11.5.3) for distributing angular degrees of freedom through the faces of an object after enforcing orientation constraints is unsatisfactory both theoretically and in practice. An improved algorithm is required. There is also, as yet, no solution to the resolvable representation problem, but it is not clear how serious this omission is in practice.

13.2 Sketch to Line Drawing

For the purpose of the thesis, it has been assumed that conversion of freehand sketches to line drawings is straightforward. However, some ideas in this thesis go beyond what is currently available in the area, so this process could usefully be reinvestigated.

The approach of Qin et al [137, 138] is interesting and successful in achieving their aims, but their interventionist ideas and their choice of using wireframe input are incompatible with the assumptions behind RIBALD.

JMsketch [112], a state-of-the-art sketching program which can produce line drawings as output, is slow when compared with the frontal geometry components of RIBALD, and does not handle *T*-junctions satisfactorily.

In detection of lines intended to be parallel in line drawings, the ideas of Chapter 5 approach the limits of what is possible. If it is accepted that parallelism must be a

hypothesis, it may be possible to form a more reliable assessment of the merit of the hypothesis from the sketched lines drawn by the user than from the "tidied" lines of a line drawing.

13.3 Curves

An obvious extension to the ideas of this thesis is a system which can deal with simple curved objects. This is perhaps a less pressing problem than it may seem, since many surfaces in engineering objects are either blends or cylindrical (drilled) holes, and both of these are easily added within CAD packages.

If this problem is investigated, several of the ideas in this thesis will require modification as they embody assumptions which no longer hold. For example:

RIBALD assumes that edges join two vertices. This is not necessarily the case for curved objects—even such a simple curved object as a cylinder has edges but no vertices. Invalidating such a fundamental assumption also invalidates most of the algorithms in this thesis.

In addition, in polyhedra, the geometry of an edge is easily determined. As seen in Figure 1.10 (page 11), not only is it more difficult to determine a geometry for a curved edge, but such determinations must be subject to repeated validation in the light of knowledge of the rest of the object.

The line labelling algorithms of Chapter 4 assume *arc consistency*–a line has the same label throughout its length. As noted by Huffman when first proposing line labelling [56], this is not always true of drawings of curved objects. This also invalidates most of the methods described in this thesis.

Much of Chapter 11 relies on being able to express relationships (such as *parallel* and *perpendicular*) between two face normals. Curved faces do not have single-value face normals, so expressing such relationships is problematic at best and impossible in the general case.

13.4 Features

Feature hypotheses (Chapter 6) are found to improve interpretation considerably, both by acting as an aid to line-labelling (Chapter 4) and (in particular) assisting in

the problematic area of deducing hidden topology (Chapter 10). These advantages would be lost if the wrong feature set were incorporated.

It is an axiom of this thesis that interpretation of line drawings is a learned skill. Extending this, I hypothesise that the skill is learned by encountering examples of object features which become so familiar that they are recognised unconsciously when they occur in line drawings. This hypothesis is more contentious, and should be investigated. If accepted, it implies that different users, learning different skills, recognise different features in drawings.

Although RIBALD demonstrates the concept, it is limited to hole-loop features and varieties of slot. Before a commercial equivalent is produced, a survey should be performed of the intended application area to identify other common features.

13.5 Psychology

During the course of the research in this thesis, several interesting questions have arisen which are more related to psychology than to geometry or computer science:

- Do people normally draw things in any standard projection (e.g. isometric)?
 - Does this vary with profession?
- Do people draw things as if they lie on an invisible table?
- Is there any general rule about what people draw first when doing line drawings?
 - If there is, what information can be gleaned from drawing order?
 - Again, does this vary with profession?
- Do left-handed and right-handed people draw things differently?
 - If so, how? Is it simple lateral inversion, or are there other, more subtle, differences?
 - Which version of a line-drawing is the left-handed one?
- Are there statistically-favoured interpretations of the "problem" drawings?

- Does interpretation vary with profession or handedness?

Although beyond the scope of this thesis, these questions should be investigated. This thesis suggests that it is, in principle, possible for a machine to duplicate the performance of a human in interpreting line drawings. Before going too much further, it would be sensible to determine what it is that we are trying to duplicate.

Appendix A

Glossary

The following terms and abbreviations have defined meanings when used in this thesis:

Atom: anything which is indistinguishable except by position from any other atom of the same kind; in a boundary representation model, these are *vertices*, *edges* and *faces*.

Coordinate: used in its normal geometric sense.

Corner: a vertex, considered only as something which bounds a face (so a corner is always connected to two *sides*, one preceding it and the other following it in a *loop*) (c.f. *junction*, *vertex*).

CSP: Constraint Satisfaction Problem.

Edge: the locus of intersection of two faces of a polyhedral object (c.f. line, side).

Extended trihedral: a vertex is extended trihedral if the (four or six) faces meeting at it lie in exactly three planes; an object is extended trihedral if all of its vertices are either trihedral or extended trihedral.

Face: a face of a polyhedral object, bounded by *loops* of *sides* and *corners* (c.f. region).

Figure of Merit: a real number indicating confidence in a hypothesis, ranging from 0.0 (the hypothesis is clearly untrue) to 1.0 (the hypothesis is clearly true).

FoM: abbreviation for Figure of Merit, q.v.

General Viewpoint: a drawing is made from a general viewpoint if no small change in the location of the viewpoint results in a change in the topology of the drawing.

Geometry: the continuous data associated with an object, describing the locations of vertices, edges and faces in space; this conforms (roughly) to CAD usage.

Junction: a point in the 2D drawing at which two or more lines meet (c.f. vertex).

Line: (i) a (visible) line between two junctions in a 2D drawing (c.f. edge, side); (ii) more generally, the everyday usage (the shortest distance between any two 2D or 3D points).

Location: a location in 2D (xy) or 3D (xyz) space, specified by two or three coordinates.

Loop: a cyclic alternating sequence of *corners* and *sides*.

Normalon: an object in which all edges and all face normals are parallel to one of the three coordinate axes.

Oojit: a seven-sided polyhedron with seven vertices obtained by removing a triangular pyramid from a cube.

Point: any location in 2D or 3D space, irrespective of the presence of a junction or vertex.

Position: a static situation to be evaluated.

Region: an area of a 2D drawing bounded by lines (c.f. face).

Semi-normalon: an object in which most edges and most face normals are parallel to one of the three coordinate axes.

Side: an edge, considered only as something which joins two corners and bounds a face (effectively the same as a half-edge) (c.f. edge, line).

Topology: the discrete data associated with an object, describing how vertices, edges and faces combine; this conforms (roughly) to CAD usage (except that in this thesis edge *vexity* is considered to be part of the topology); it is clearly distinct from the mathematical usage.

Trihedral: a vertex is trihedral if exactly three edges meet at it; an object is trihedral if all of its vertices are trihedral.

 \mathbf{T} -vertex: the true vertex at which the occluded line at an occluding T-junction terminates.

Vertex: a point on a polyhedral object at which three or more edges meet (c.f. junction).

Vexity: an abbreviation for *convexity/concavity*.

Appendix B

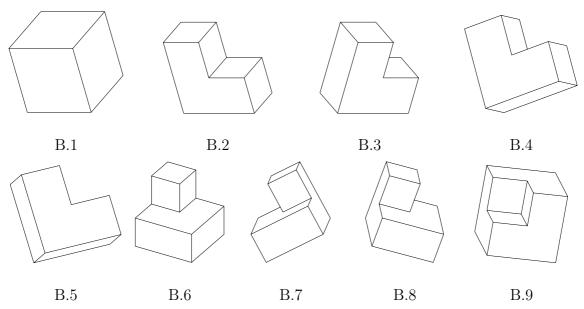
Test Drawings

These test drawings can be found in electronic form at http://ralph.cs.cf.ac.uk/Data/Sketch.html.

B.1 Trihedral Genus Zero Polyhedra

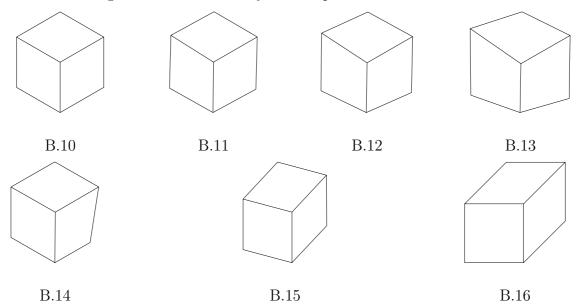
B.1.1 Trihedral Junction Catalogue

These drawings illustrate all possible trihedral junction labels (the labelled versions can be found in Appendix E).



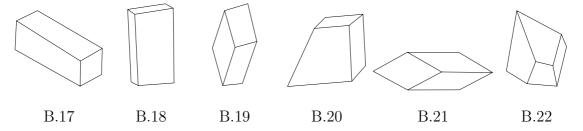
B.1.2 Single Cubes

Various drawings which should ideally be interpreted as cubes.



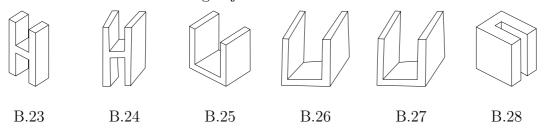
B.1.3 Not Cubes

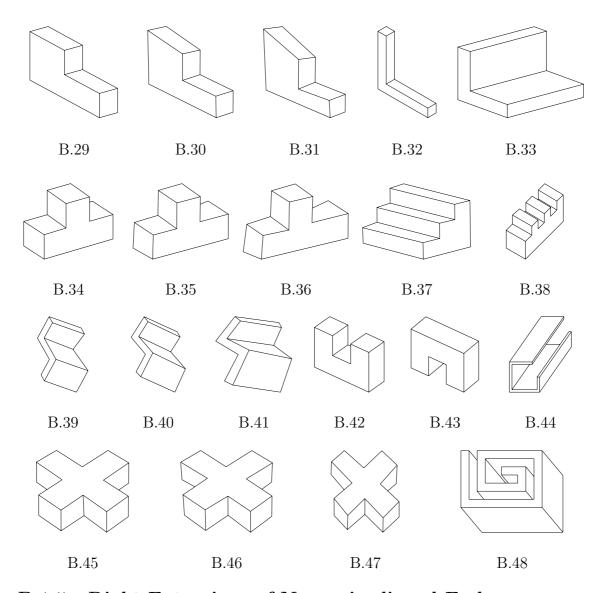
Various sketches, topologically equivalent to cubes, which should not be interpreted as cubes.



B.1.4 Axis-Aligned Extrusions

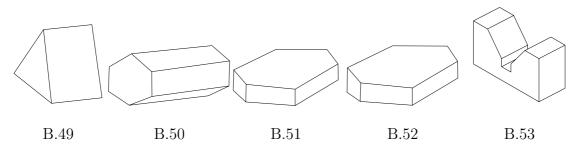
Beams and channels are standard engineering components. The remaining drawings are inspired by letters (e.g. the L-, T- and X-blocks) or are extrapolations of these ideas. Figure B.44 illustrates the point that drawings of extrusions can include lines both ends of which are occluding T-junctions.

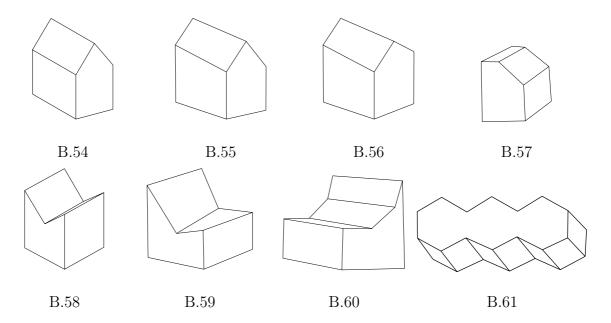




B.1.5 Right Extrusions of Non-axis-aligned End-caps

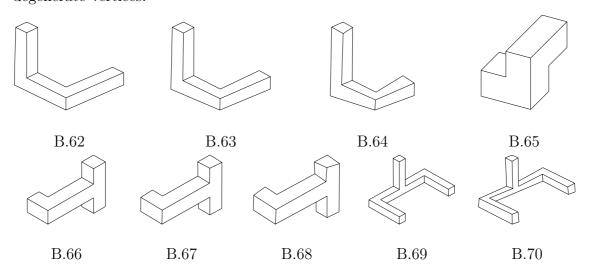
Prisms are common geometric objects. Figures B.54–B.57 can cause problems if the merit for rotational symmetry is too high—they are clearly not intended to be regular pentagonal prisms. Figure B.53 is a simplification of Figure B.452; Figure B.58 takes the idea further. Figure B.61 was inspired by the Anthracene molecule.

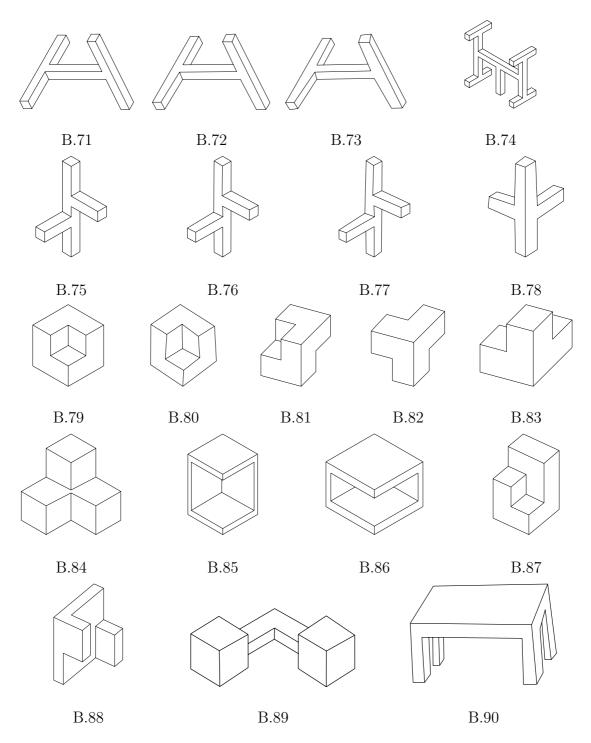




B.1.6 Axis-Aligned Non-Extrusions

The Z-block, Figure B.62 appears in many previous investigations as being the simplest normalon with no other "clues"—it is not an extrusion and has no axis of mirror symmetry. Other figures were inspired by other letters of the alphabet. Figure B.87 illustrates a particular uncommon trihedral junction label pair. Figures B.85 and B.86 show that hidden topology can sometimes be at the front, not the back, of the object. Figure B.84 illustrates Kanatani's suggestion for labelling non-trihedral vertices. Note that the proper interpretation of Figure B.65 is non-trihedral and a geometrically-accurate interpretation of Figure B.81 would contain degenerate vertices.

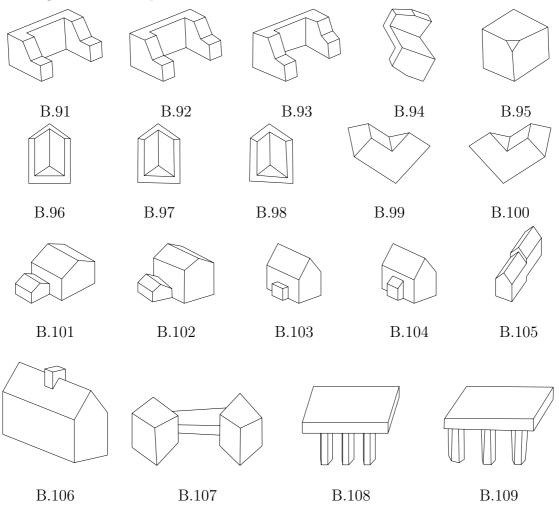




B.1.7 Semi-Axis-Aligned with Mirror Plane

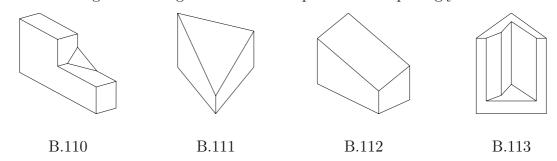
Grimstead's bracket (Figures B.91 to B.93) was the figure chosen to demonstrate the capabilities of his system [38]. The poorly-drawn Angle bracket (Figure B.98) appears in [163] and other references to illustrate a common drawing error. Figure B.99 is a problem drawing—should it be mirror-symmetric or semi-axis-aligned with one non-axis-aligned face? Architecture can often be approximated by semi-axis-aligned

drawings with mirror planes.



B.1.8 Semi-Axis-Aligned without Mirror Plane

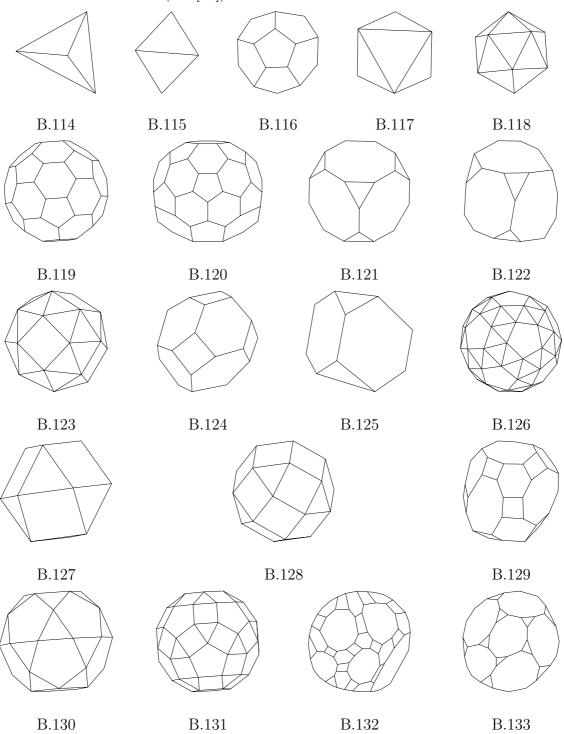
Semi-axis-aligned drawings without mirror planes are surprisingly uncommon.



B.1.9 Regular and Semi-Regular

Although best handled as special cases, drawings of Platonic and Archimedean solids also make useful test cases for topological reconstruction using symmetry. Figures B.134 and B.135 show two views of one of the semi-regular convex solids (all

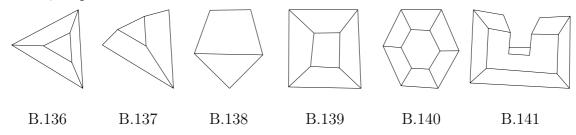
faces are regular pentagons or squares, but not all vertices are interchangeable); there are several others (see [19]).





B.1.10 Right Frusta (by definition, not axis-aligned)

Inaccurate versions of Figure B.136 appear in several references, usually to illustrate the point that strictly mathematical approaches are intolerant of freehand drawing errors. Figure B.140 is a useful illustration of which edges can, and which edges cannot, be parallel.

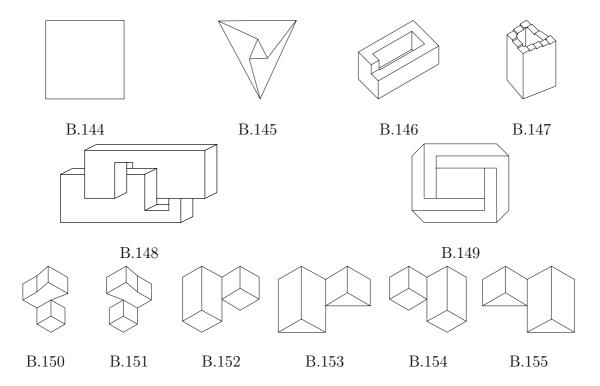


B.1.11 Other Trihedral



B.1.12 Impossible Objects and Invalid Drawings

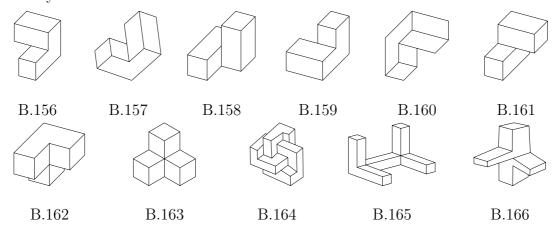
The square (Figure B.144) contravenes either the general viewpoint or the most informative viewpoint assumptions. The impossible objects, Penrose's frustum (Figure B.145 [124]), Sugihara's Box (Figure B.146 [163]), Escher's Tower (Figure B.147), Huffman's Combs (Figure B.148 [56]) and Cowan's Ring (Figure B.149), are a reminder that not every valid topology can be realised geometrically. The degenerate objects (Figures B.150–B.155) illustrate why certain junction labels should *not* be included in the tetrahedral catalogue.



B.2 Non-Trihedral Genus Zero Polyhedra

B.2.1 Extended Trihedral

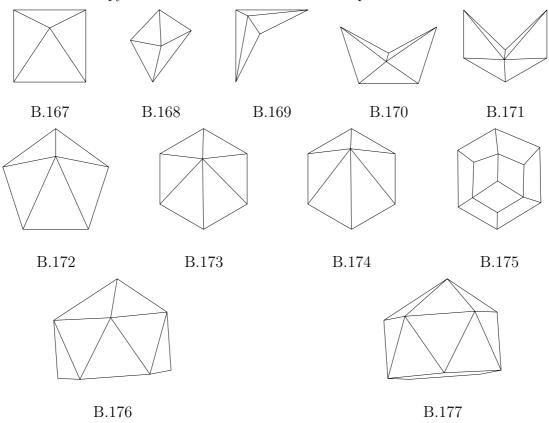
Figures B.156–B.163 illustrate the entire extended trihedral junction catalogue. The trefoil, Figure B.164 [19], tests line labelling and topological reconstruction more seriously.



B.2.2 Non-Trihedral Pyramids

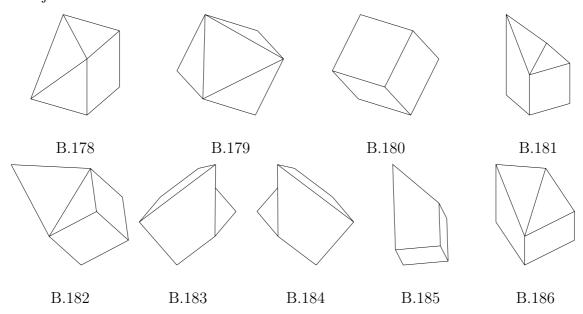
These drawings illustrate the view that all-convex pyramid vertices are commonly found in engineering objects, but single-concave pyramid vertices (Figures B.169–B.171 are not. Figure B.175 is an interesting optical illusion—the central vertex

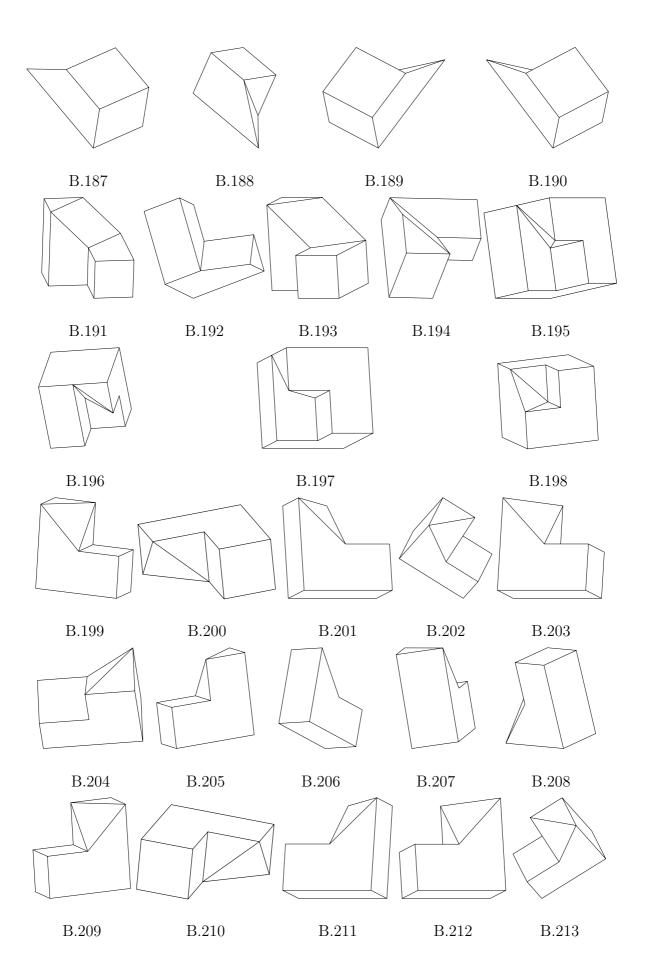
appears concave (compare with Figure B.79), but the best geometric realisation is as an all-convex pyramid which is shallower at the top than the bottom.

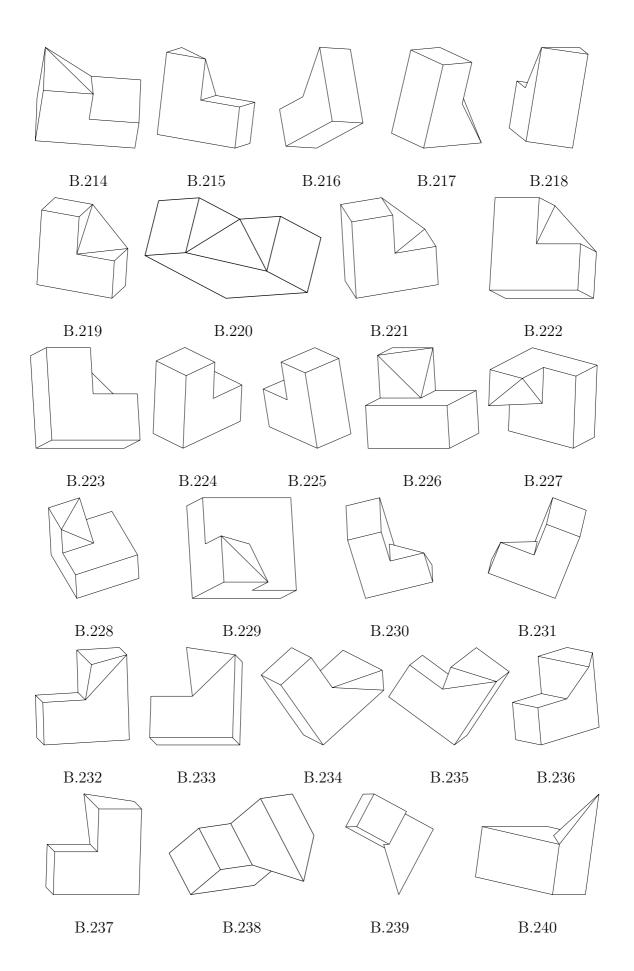


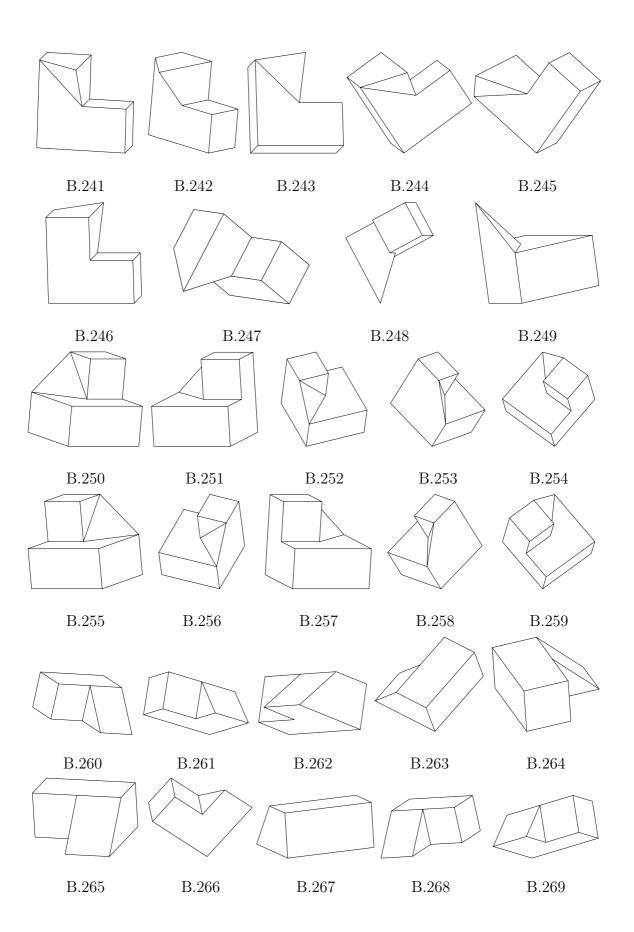
B.2.3 Tetrahedral Junction Catalogue

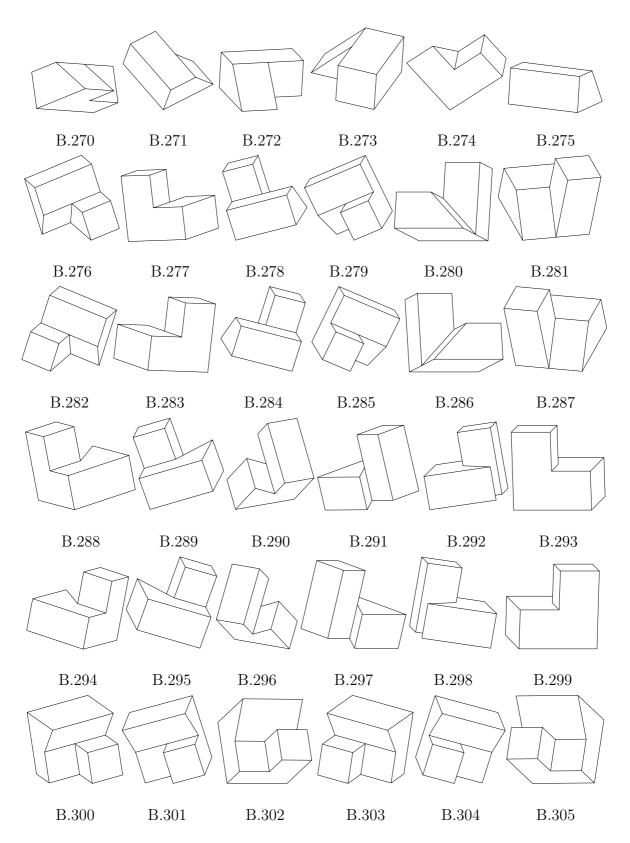
These drawings illustrate the tetrahedral junction catalogue—see Appendix E for context. Their inclusion ensures that the implementation of each possible tetrahedral junction label is tested.







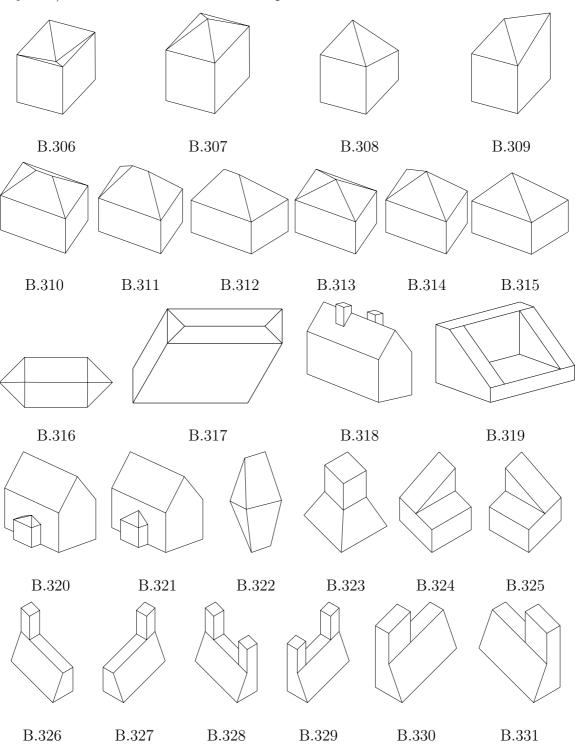


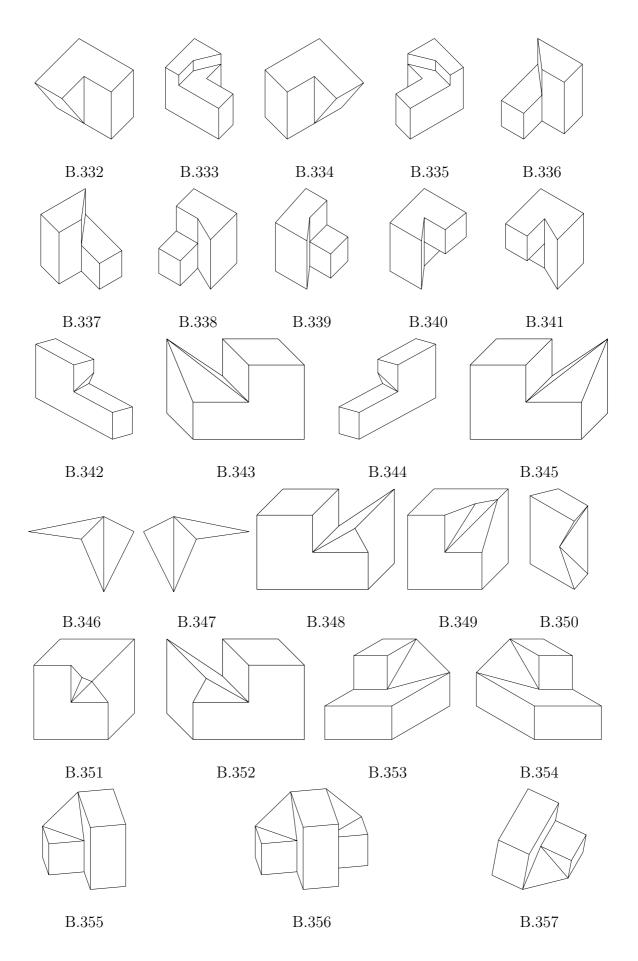


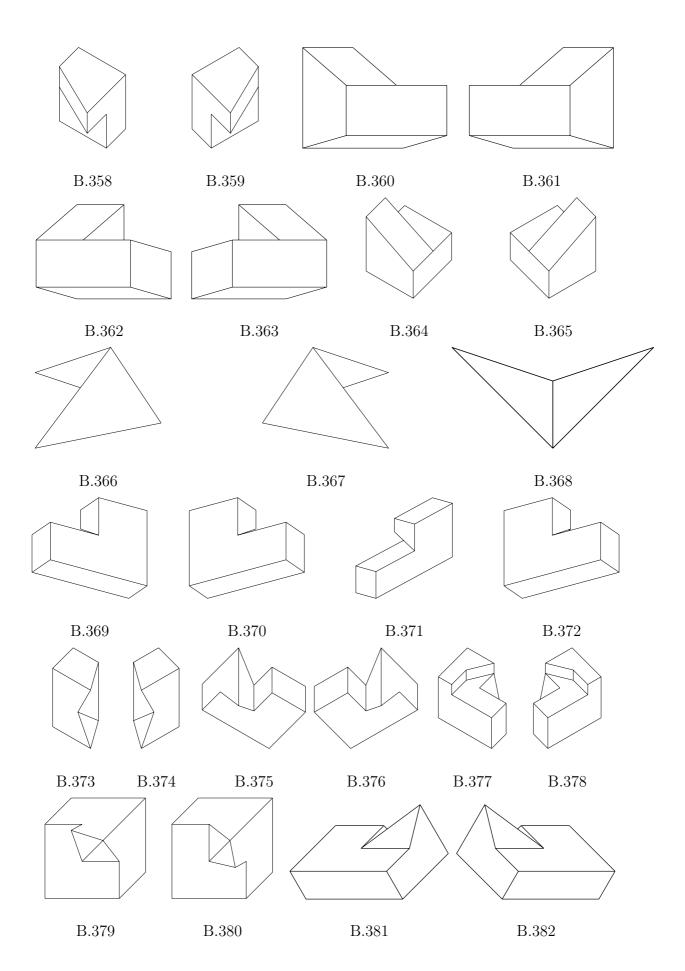
B.2.4 General Non-trihedral Objects

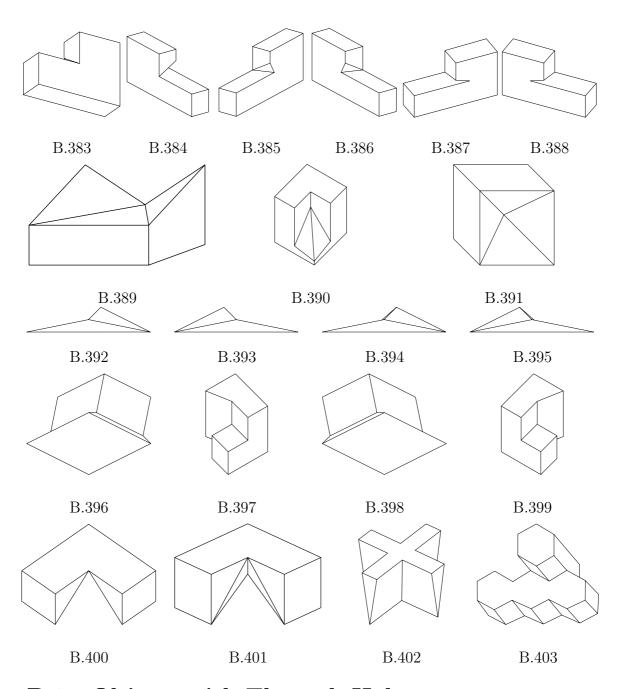
Figures B.306–B.309 illustrate an unsolved problem line labelling, that of incorporating geometric information. Although architecture usually remains semi-axis-aligned

and usually retains its mirror plane, non-trihedral vertices are common. Other drawings in this section illustrate non-trihedral vertices in engineering contexts (not all of the drawings in the previous section could be considered "common engineering objects") or are variants of those in the previous section.







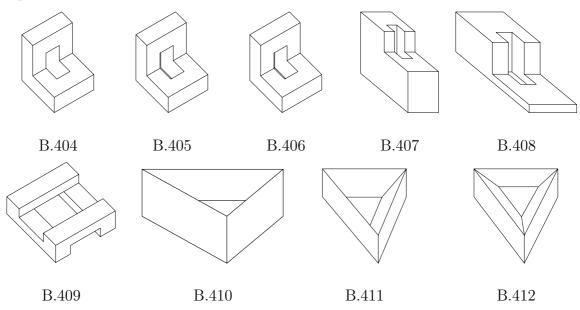


B.3 Objects with Through Holes

B.3.1 Through Holes Without Hole Loops

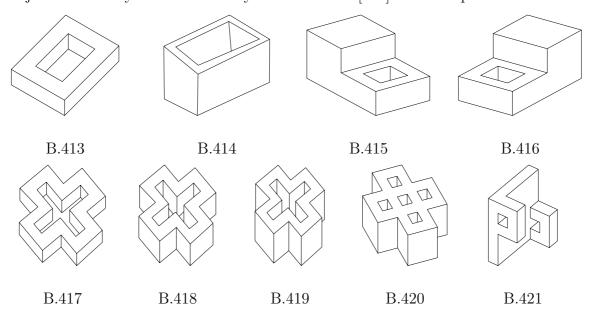
These drawings test object validation—in applying Euler's formula, it cannot be assumed that an object with no hole loops has no through holes. The Hannoid (Figure B.409) was taken from [152]. Figures B.410–B.412 illustrate another problem with line labelling (what object do they represent?); one interpretation is the object used by Sugihara [165] to illustrate a polyhedron with no first-order resolvable

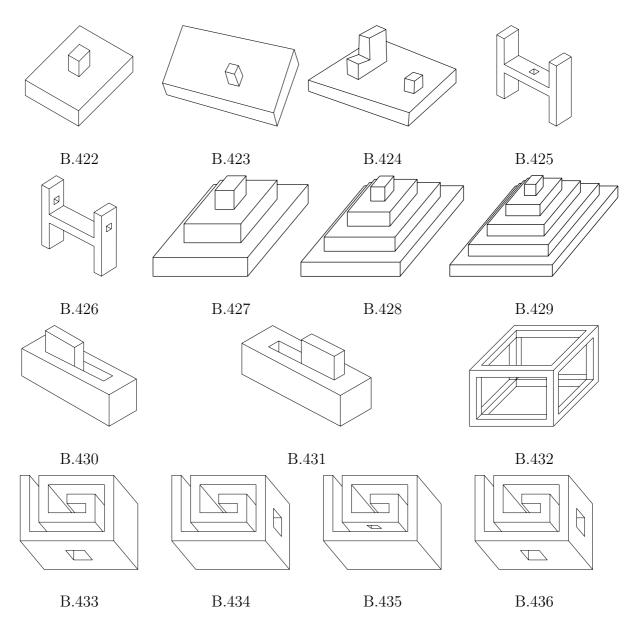
representation.



B.3.2 Axis-Aligned with Hole Loops

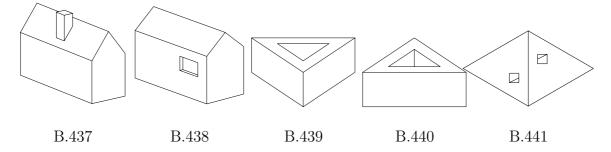
Distinguishing holes from bosses is usually straightforward (Figures B.430 and B.431 are counterexamples). Distinguishing holes from pockets is not (e.g. Figure B.419). Identifying where a hole stops can also present problems when the face in which the hole terminates is not visible, as in Figures B.433–B.436. Figure B.432 shows an object more easily reconstructed by CSG methods [182] than B-rep.





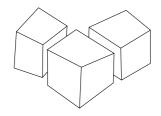
B.3.3 Non-Axis-Aligned with Hole Loops

The method for distinguishing holes/pockets from bosses was derived for the axisaligned case. These drawings test whether it works for non-axis-aligned drawings.



B.4 Multiple Polyhedra

RIBALD assumes that a drawing shows a single polyhedron.

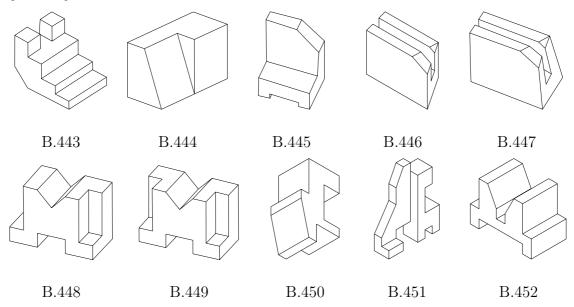


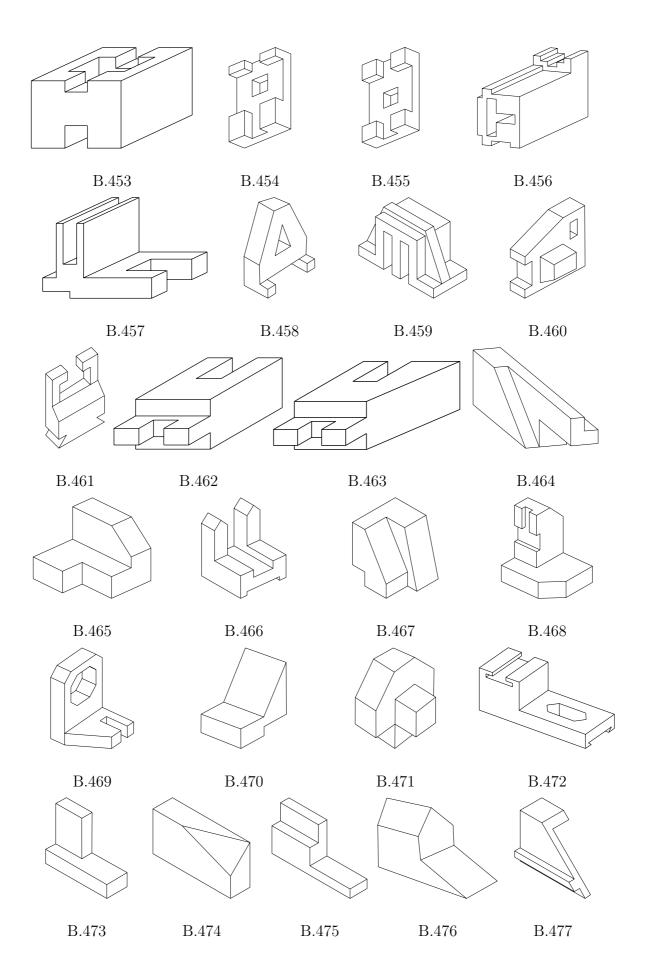
B.442

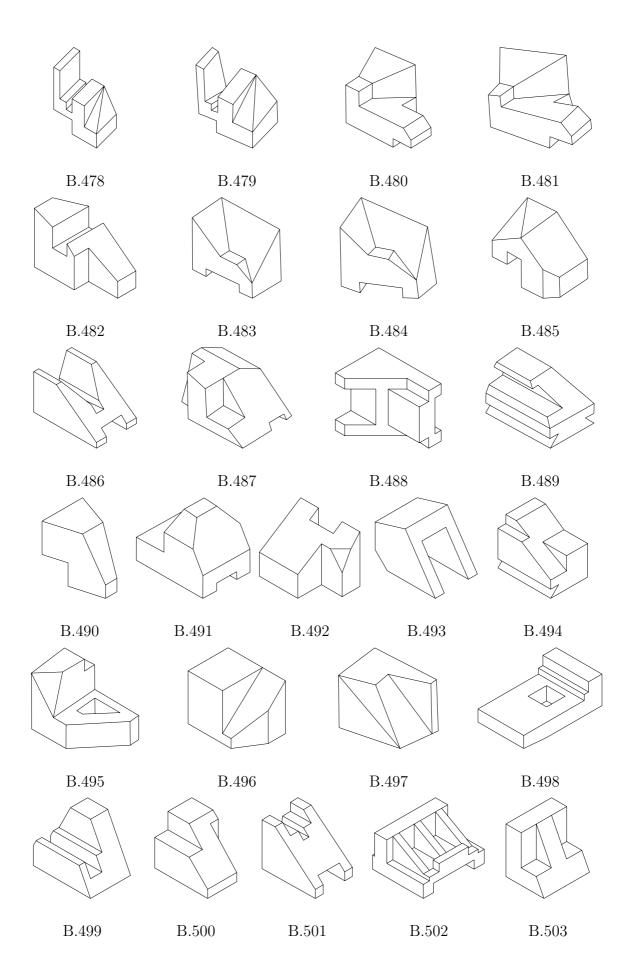
B.5 Figures based on Collections

B.5.1 Figures based on Yankee [194]

As isometric projection can produce coincidences which break the "general view-point" rule, the viewpoints of most of these drawings have been changed slightly. Some drawings which originally included curves have been included: cylindrical through holes were either omitted from the object or converted to square or octagonal through holes, and corner blends were either omitted from the object or converted to octagonal corners. Where such simple adjustments were not available, the drawing was omitted. Some drawings have been duplicated, either in well-drawn and poorly-drawn versions, or (in the case of Figure B.449) to add a plane of mirror symmetry.

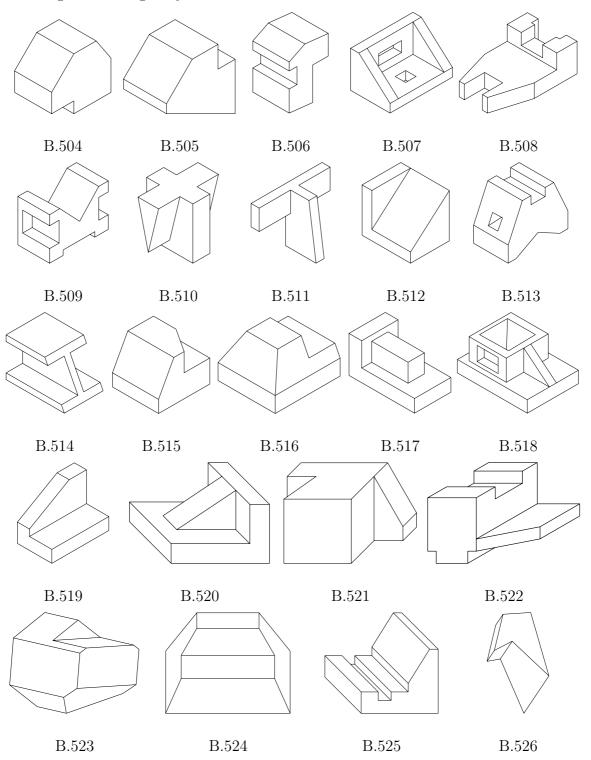


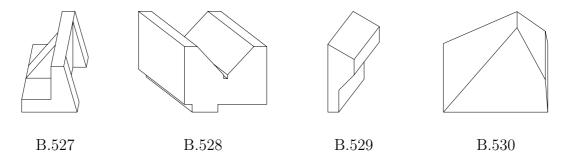




B.5.2 Figures based on Pickup and Parker [128, 129]

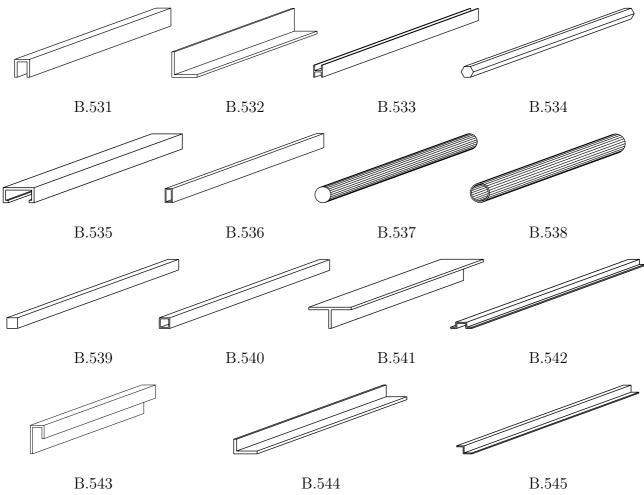
As isometric projection can produce coincidences which break the "general view-point" rule, the viewpoints of most of these drawings have been changed slightly. Drawings which originally included curves have been omitted.





B.5.3 Figures from an Extrusion Catalogue [12]

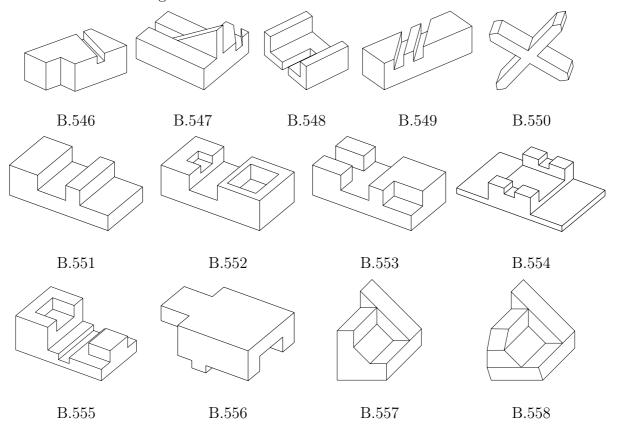
These extrusions appear in a catalogue [12] of standard parts. Figure B.537 illustrates one way in which RIBALD could process "curved" objects—although neither elegant nor ergonomic, it works.



B.5.4 Figures from Other Sources [91], [107] and [148]

Figures B.546–B.550 are test drawings from Lipson and Shpitalni [91], included to provide a comparison between their methods and RIBALD's. Figures B.551–B.556 are from Meeran and Taib [107], whose interest is feature detection. Figure B.558

comes from Shirai [148]; the simplification in Figure B.557 looks more like an engineering component, but still includes a pentahedral (extended tetrahedral) vertex with two concave edges.



Appendix C

Tuning Constants

In various algorithms in this thesis, numerical estimates are made of hypotheses suggested by heuristics. Many of these numerical estimates are multiplied or divided by arbitrary values in order to take some account of their relative importance. In most places, these arbitrary values have been implemented in RIBALD as tuning constants, run-time constants for which the default value can be changed as a command-line option.

Some attempt has been made to optimise these tuning constants, as described in Section C.2 below. The initial values for this optimisation process were guessed; it is to be hoped (but cannot be guaranteed) that the results of these guesses were sufficiently close to the global minimum.

C.1 Tuning: Configurable Constants

The default values and the use of each tuning constant are listed.

 F_b (1.06), page 107: higher values provide more discouragement for interpreting subgraphs with boundary edges as bosses.

 F_c (3), page 107: lower values provide more encouragement for non-hole-loop interpretation of outer subgraphs in cofacial configurations.

 F_o (0.025), page 107: a bias to favour non-hole-loop interpretations of subgraphs.

 F_u (0.995), page 104: the base figure of merit for an underslot feature.

 F_v (0.516), page 104: the base figure of merit for a valley feature.

 G_x (0.0), page 199: used in calculating geometric figures of merit for normalon

vertex locations given topological figures of merit (low values correspond to increased confidence)

 G_y (2.5), page 199: used in calculating geometric figures of merit for non-normalon vertex locations given topological figures of merit (low values correspond to increased confidence)

 k_E (0.000), page 74: used in assessing the merit of a labelling, based on the proportion of lines labelled as occluding.

 M_d (1.0412), page 345: used in the figure of merit for two points being in the same location (the higher the value, the stricter the test)

 M_p (47.4), page 93: used in the figure of merit for line or edge parallelism (the higher the value, the stricter the test)

 M_r (1.032), page 345: used in the figure of merit for equality of commensurate quantities (the higher the value, the stricter the test)

 S_w (0.750), page 204: figure of merit multiplier for cross-mirror edges.

 S_x (0.790), page 197: figure of merit multiplier for crossings of hypothesised (i.e. non-extended-T-junction) lines.

 S_y (0.850), page 210: figure of merit multiplier for hypothesised edges connecting different subgraphs.

 S_z (0.850), page 210: figure of merit multiplier for hypothesised edges connecting different subgraph types.

 T_a (0.600), page 207: base figure of merit for an edge between the last two incomplete vertices.

 T_b (0.238), page 207: base figure of merit for adding a vertex and two edges when only two necessarily incomplete vertices remain.

 T_c (0.936), page 207: base figure of merit for adding a vertex and three edges when only three necessarily incomplete vertices remain.

 T_d (0.267), page 207: base figure of merit for hypothesising two edges when only four necessarily incomplete vertices remain.

 T_e (0.558), page 195: figure of merit for hypothesising an edge at a vertex allows but does not require an extra edge.

 T_f (0.430), page 202: base figure of merit for local occluding T-junction completion

 T_g (0.246), page 203: base figure of merit for distant occluding T-junction completion

 T_h (0.719), page 202: base figure of merit for adding a vertex and two edges to complete a quadrilateral face, if the object is a normalon and both edges will be convex

 T_i (0.497), page 202: base figure of merit for adding a vertex and two edges to complete a quadrilateral face, if the object is a normalon but one or both edges will be concave

 T_j (0.606), page 202: base figure of merit for adding a single edge to complete a quadrilateral face, if the object is a normalon and the edge will be convex

 T_k (0.510), page 202: base figure of merit for adding a single edge to complete a quadrilateral face, if the object is a normalon and the edge will be concave

 T_l (0.884), page 204: base figure of merit for discrete hypotheses based on mirror chains

 T_m (0.655), page 204: base figure of merit for the mirror macro hypothesis

 T_n (0.412), page 210: figure of merit for splitting edge hypotheses when the hypothesised edge passes close to an incomplete vertex.

 T_o (0.563), page 211: figure of merit multiplier for a hypothesis which introduces a triangular loop of edges where no triangles are visible in the original drawing

 T_p (0.144), page 209: figure of merit multiplier for improperly-placed vertices

 T_q (0.923), page 192: base figure of merit for the quadrilateral loop hypothesis

 T_r (0.695), page 209: edge length dropoff, used in the figure of merit for long edges

 T_s (0.747), page 209: edge length power, used in the figure of merit for long and short edges

 T_t (0.975), page 202: base figure of merit for adding a vertex and two edges to complete a quadrilateral face, if the object is not a normalon and both edges will be convex

 T_u (0.918), page 202: base figure of merit for adding a vertex and two edges to complete a quadrilateral face, if the object is not a normalon but one or both edges will be concave

 T_v (0.997), page 202: base figure of merit for adding a single edge to complete a quadrilateral face, if the object is not a normalon and the edge will be convex

 T_w (0.949), page 202: base figure of merit for adding a single edge to complete a quadrilateral face, if the object is not a normalon and the edge will be concave

 T_x (0.244), page 202: figure of merit bias for adding a single edge to complete a quadrilateral face

 T_y (0.084), page 202: figure of merit bias for adding a vertex and two edges to complete a quadrilateral face

 T_z (0.554), page 196: figure of merit for choosing K-vertex and Z-vertex interpretations when X-vertex and M-vertex interpretations are also possible.

Constants for relaxation labelling are shown in Tables C.1– C.5. The columns are, respectively, frequencies derived from shape pair statistics, label frequencies derived from correct labelling of the test drawings in Appendix B, and the best set of tuning constants for six, ten and twenty iterations of relaxation labelling.

Label	Shape Pair	Statistical	Rel-6	Rel-10	Rel-20
Convex	0.25	0.25	0.246	0.246	0.246
Concave	0.25	0.25	0.300	0.300	0.300

Table C.1: Constants for Relaxation Labelling—Lines

Label	Shape Pair	Statistical	Rel-6	Rel-10	Rel-20
Lba	1.000	0.097	0.040	0.051	0.044
Lab	0.157	0.157	0.070	0.073	0.062
Lac	0.313	0.313	0.337	0.340	0.331
Lbd	0.056	0.056	0.182	0.184	0.181
Laa	0.004	0.004	0.264	0.268	0.271

Table C.2: Constants for Relaxation Labelling—2-Edge Junctions

C.2 Tuning: Introduction

In several parts of the sketch interpretation system, it has been found necessary to use numerical heuristics to choose between alternatives. Each alternative is allocated a *figure of merit*, and the alternative with the highest figure of merit is chosen. Figures of merit for alternatives (e.g. "which is the best labelling?", "which is the

Label	Shape Pair	Statistical	Rel-6	Rel-10	Rel-20
Tbaa	1.000	0.174	0.184	0.211	0.188
Tbac	0.109	0.109	0.108	0.107	0.109
Tbad	0.356	0.356	0.246	0.254	0.247
Tbda	0.009	0.009	0.060	0.046	0.063
Tbdc	0.021	0.021	0.099	0.096	0.096
Taba	0.002	0.002	0.160	0.141	0.153
Tbca	0.002	0.002	0.001	0.001	0.001
Tbcc	0.009	0.009	0.033	0.033	0.036
Tcca	0.030	0.030	0.126	0.145	0.124
Tcda	0.012	0.012	0.108	0.126	0.136
Tdda	0.007	0.007	0.050	0.044	0.043
Wbca	1.000	0.219	0.274	0.276	0.268
Wcdc	0.621	0.621	0.221	0.205	0.195
Wdcd	0.090	0.090	0.251	0.268	0.266
Wabc	0.003	0.003	0.177	0.200	0.190
Wabd	0.003	0.003	0.144	0.125	0.123
Wacc	0.003	0.003	0.105	0.111	0.115
Wacd	0.003	0.003	0.001	0.003	0.001
Wadc	0.005	0.005	0.001	0.006	0.014
Wbaa	0.001	0.001	0.009	0.002	0.002
Wbcd	0.004	0.004	0.089	0.091	0.087
Wbda	0.001	0.001	0.001	0.001	0.001
Wbdc	0.004	0.004	0.197	0.194	0.190
Wbdd	0.003	0.003	0.172	0.165	0.161
Wcac	0.003	0.003	0.137	0.137	0.142
Wcbd	0.003	0.003	0.172	0.158	0.157
Wdad	0.002	0.002	0.008	0.002	0.002
Yccc	0.915	0.915	0.456	0.438	0.462
Yddd	0.038	0.038	0.110	0.110	0.111
Yabd	0.026	0.026	0.348	0.391	0.346
Yaab	0.001	0.001	0.129	0.177	0.153
Yabc	0.001	0.001	0.126	0.127	0.101
Yacc	0.002	0.002	0.164	0.155	0.163
Yacd	0.004	0.004	0.068	0.062	0.067
Yadd	0.005	0.005	0.055	0.050	0.064

Table C.3: Constants for Relaxation Labelling—3-Edge Junctions

Label	Shape Pair	Statistical	Rel-6	Rel-10	Rel-20
Kabcd	1.000	0.013	0.262	0.258	0.274
Kcccd	1.000	0.316	0.453	0.460	0.440
Kcdcd	0.105	0.105	0.301	0.308	0.313
Kddcd	0.066	0.066	0.275	0.280	0.132
Mbcca	1.000	0.178	0.139	0.141	0.158
Mbcda	0.014	0.014	0.111	0.109	0.126
Mccdc	0.123	0.123	0.074	0.074	0.091
Mcdcd	0.096	0.096	0.019	0.021	0.038
Mcddc	0.068	0.068	0.146	0.148	0.165
Mdccd	0.068	0.068	0.182	0.177	0.194
Mdcdd	0.110	0.110	0.007	0.002	0.019
Xabcd	1.000	0.001	0.308	0.246	0.236
Xabdd	0.001	0.001	0.002	0.018	0.003
Xcbda	0.013	0.013	0.069	0.108	0.117
Xcccc	0.642	0.642	0.403	0.385	0.426
Xcccd	0.254	0.254	0.518	0.455	0.495
Xcdcd	0.039	0.039	0.300	0.276	0.282
Xcddd	0.022	0.022	0.001	0.006	0.002
Xdddd	0.022	0.022	0.001	0.013	0.018

Table C.4: Constants for Relaxation Labelling—4-Edge Junctions

Label	Shape Pair	Statistical	Rel-6	Rel-10	Rel-20
Xccccc	1.000	0.972	0.242	0.242	0.242
Xddddd	0.014	0.014	0.001	0.001	0.001
Xabccc	0.014	0.014	0.136	0.136	0.136
Xccccc	1.000	0.556	1.000	1.000	1.000
Xcdcdcd	0.333	0.333	1.000	1.000	1.000
Xdddddd	0.111	0.111	1.000	1.000	1.000

Table C.5: Constants for Relaxation Labelling—Other Junctions

most plausible way of extending the topology of this object?") are generally calculated from figures of merit for lower-level concepts which the hypotheses embody (e.g. "how likely is it that this vertex is trihedral?", "is this extension predicted by rotational symmetry?"). The figures of merit for these lower-level concepts may themselves be derived from concepts at a still lower level ("are these lines likely to be parallel?"), but at some point actual numbers are used, and the choice of these numbers is arbitrary.

It seems likely that the performance of the system could be improved by a more astute choice of these numerical values. The more of them there are, the more likely it is that a good choice will produce benefits. In many cases, there are so many of them that it is effectively certain that the initial arbitrary choices will not be the set which produces the best results. This applies to line labelling, which is discussed further here. It also applies to object classification and topological reconstruction; the methods described here were also applied to optimise those.

C.3 Tuning for the Labelling Problem

As described in Chapter 4, labelling of non-trihedral sketches is rarely unambiguous. Even for simple sketches, there may be hundreds of interpretations. In order to progress, it is necessary to identify a small number (ideally, one) of preferred interpretations. "Preference" is by definition heuristic, not algorithmic, so heuristics are required for assessing the merits of competing interpretations.

Worse, the number of possible interpretations increases exponentially with the number of junctions. Quite "sketchable" sketches may have millions of interpretations, and some test case line drawings (e.g. the most complex Archimedean solids) have absurdly large numbers of interpretations. It is not possible either to store each competing interpretation nor even to generate and assess each one in a reasonable time. Methods of pruning the tree of interpretations are required; these methods too must be heuristic. Ideally, they should also be very quick, and capable of pruning off entire branches of the search tree.

To ensure that the method terminates in a reasonable time, it has also been found necessary to limit the number of labellings assessed by the slower, more thorough heuristic to a fixed maximum number.

The heuristics can thus be divided into three groups:

- A: heuristics which contribute to the assessment and which are based on the labelling as a whole; these heuristics cannot be used to prune the labelling tree.
- B: heuristics which contribute to the assessment and which are based on individual junction or line labels; these heuristics can sometimes be used to prune the labelling tree.
- C: heuristics for determining which branch of the labelling tree to investigate first.

The resulting algorithm is thus:

- Start Here
- If there are already too many labellings, do nothing and drop through to the end.
- Label as far as possible using the Clowes-Huffman line-labeller. Whenever a junction or line is labelled unambiguously, assess it using heuristics (B). If the merit drops below the acceptable threshold, discard it and drop through to the end.
- If the merit is above the acceptable threshold:
 - No valid labellings: discard the current labelling.
 - Unambiguous labelling: complete the merit assessment using heuristics (A). If the merit is still above the acceptable threshold, store the current labelling, otherwise discard it. If the labelling is the best so far, re-assess the acceptable threshold.
 - Labelling still ambiguous: Create a duplicate of the current labelling. Use heuristics (C) to identify a preferred labelling of a chosen ambiguous edge or vertex. Set the current labelling of this edge or vertex to the preferred labelling and the duplicate labelling of the same edge or vertex to all previous possibilities except the preferred labelling. Divide

the remaining number of allowed labellings between the current and the

duplicate. Follow the procedure (from "Start Here" to "End") recursively

for both the current and duplicate.

• End or Pop to previous recursion level

C.3.1**Heuristics Considered**

For group A heuristics (assessing the object as a whole) see Chapter 4.4.1.

Currently group B heuristics are implemented only for junctions. Each of the

common (trihedral) junction labels has an associated merit figure; there is also a

collective merit figure for non-trihedral labellings. Whenever a junction is labelled

unambiguously, the merit for the labelling is multiplied by the junction label merit

figure.

Currently the group C heuristic, identifying preferred branches, is implemented

in two stages.

• list the possible common interpretations of all ambiguous junctions, and choose

as the preferred interpretation the one with the highest merit

• if no ambiguous junctions have common interpretations, choose an arbitrary

ambiguous edge, and choose as the preferred interpretation: convex (if this is

possible), otherwise concave (if this is possible), otherwise an arbitrary direc-

tion.

The "tuning constants" for group C heuristics are not merit figures but merely

provide a preference order for the various common vertex types. The labellings

considered are:

• Boundary L: Lba

• Non-boundary L: Lba, Lac, Lcb

• Any T: Tbaa, Tbab

• Boundary W: Wbca

• Non-boundary W: Wcdc, Wdcd, Wbca

341

• Boundary Y: Yabd

• Non-boundary Y: Yccc, Yddd

• Non-boundary X: Xcccc

The proportion in which allowed labellings are split between the preferred and alternative branches of the tree is also a tuning constant.

Currently, the number of stored labellings and the maximum number of labellings which will be assessed are fixed, at 20 and 2000 respectively. The threshold below which partial labellings will automatically be rejected is $(2 \times M_1) - 1$ where M_1 is the merit of the best labelling so far.

C.3.2 Methodology

The "correct" labellings for each of a set of test drawings (numbering 444 at the time) was determined by hand.

A set of tuning constants is assessed by running the labelling part of the RIBALD program and then comparing the results with the "correct" interpretation. If they are identical, this scores zero; discrepancies result in positive scores (see below). The objective function is the sum of the scores achieved for each test drawing.

An optimal set of tuning constants (for this set of test drawings) is determined by using a standard downhill optimisation routine [117] to minimise the objective function.

Originally, each discrepancy (junction or line label not as expected) was scored as 1.

Implementation problems with other parts of the system indicate that it is important that the preferred interpretation identifies occluding and non-occluding T-junctions correctly. With this exception, correct identification of individual junction and line labels is not vital—the correct labelling need not be the first-choice labelling, but should appear somewhere in the list of stored labellings.

Therefore, the score for discrepancies was modified as follows:

• a discrepancy in the number of edges (such as would result from an identification of an occluding T-junction as non-occluding or vice versa) scores 10;

- a discrepancy in the vertices which an edge joins (such as would result from a double misidentification of *T*-junctions) also scores 10;
- a mislabelling of a junction or edge scores 3.

These constants are arbitrary: it is hoped that their exact values are comparatively unimportant. However, in view of the indication that there were two minima, with choice alternating between them as new drawings were added to the test data, it is possible that the values of scoring constants also makes a difference.

C.3.3 Results

Experimentation suggests that there is no "best" set of tuning constants: some work well with some types of drawing, others with others. It is not possible to claim that global optimum values for the various tuning constants have been found, but on the basis of the (possibly only local) minima found so far it is possible to make some comments.

Heuristics A are described in Chapter 4.4.1.

Heuristics B: the output values from the optimisation process do not differ substantially from the original guesses. This suggests that either the original guesses were implausibly good or that any reasonably sensible values which give priority to trihedral interpretations are adequate.

Heuristics C: optimising these has reduced the number of mislabellings by about 60%. Unfortunately, it seems that there are (at least) two minima of roughly equal depth. Small changes in the objective function or the set of test sketches are enough to flip the optimum from one minimum to the other. There was not time to investigate which groups of drawings "pulled" the overall minimum towards one or other local minimum.

It is possible that a more flexible set of heuristics for C is required. It is also likely that extra test drawings will be required in order to bias the results towards "typical engineering objects".

Appendix D

Figures of Merit

Figures of merit are in the range 0–1. Standard figures of merit for certain operations, used repeatedly, are defined here.

Combinations

Figure of merit for two hypotheses A and B both being true:

$$F_{A\cap B} = F_A F_B$$

Figure of merit for at least one of hypotheses A and B being true:

$$F_{A \cup B} = 1 - (1 - F_A)(1 - F_B)$$

Parallelism and Perpendicularity

Figure of merit for parallelism between two lines or edges A and B or vectors or face normals $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$:

$$F(A \parallel B) = (\hat{\mathbf{a}} \cdot \hat{\mathbf{b}})^{M_p}$$

The figure of merit for perpendicularity of two 3D lines A and B is calculated as the figure of merit for parallelism of lines B and C where line C is perpendicular to line A and in the plane formed by lines A and B

$$F(A \perp B) = F(((\hat{\mathbf{A}} \times \hat{\mathbf{B}}) \times \hat{\mathbf{A}}) \parallel B).$$

For 2D lines, this simplifies to

$$F(A \perp B) = F((A + 90^{\circ}) \parallel B).$$

Ratios

The figure of merit for the ratio of any two commensurable values A and B being equal is

$$F(A/B) = (min(|\mathbf{A}|, |\mathbf{B}|)/max(|\mathbf{A}|, |\mathbf{B}|))^{M_r}$$

This is specifically used for length ratios of two lines or edges A and B where the lines or edges are hypothesised to be of equal length

Distances

The figure of merit for any two points A and B being in the same location when the actual distance between them is D is

$$F(A=B) = M_d^D$$

Collinearity

Figure of merit for two lines A and B being collinear in 2D is the figure of merit for the distance between the starting-point of A and their crossing-point C being zero. Note that this is arbitrary—where the lines are not collinear, different numerical values will usually be obtained for collinearity of A and B, and collinearity of B and A.

Figure of merit for two lines A and B being collinear in 3D is the figure of merit for the shortest distance between line B and the starting-point of line A. Note that this is arbitrary—where the lines are not collinear, different numerical values will usually be obtained for collinearity of A and B, and collinearity of B and A.

Coplanarity

Figure of merit for a vertex V being coplanar with a face F is the figure of merit for the shortest distance between V and the plane of face F being zero.

Figure of merit for four vertices A, B, C and D being coplanar is the figure of merit for the vector BA being perpendicular to the normal of the plane through BCD. Note that this is arbitrary—when the vertices are not coplanar, different numerical values will usually be obtained for different orderings of the parameters.

Crossing

Figure of merit for two 3D lines A and B crossing is the figure of merit for the shortest 3D distance between the two lines being zero.

Constraints

2-Way Perpendicularity

Figure of merit for a two-way perpendicularity constraint (faces M and N are perpendicular):

$$F(M \perp N) = F(\hat{\mathbf{n}}_{\mathbf{M}} \perp \hat{\mathbf{n}}_{\mathbf{N}}).$$

3-Way Perpendicularity

Figure of merit for a three-way perpendicularity constraint (faces M, N and O are mutually perpendicular):

$$F \perp (M, N, O) = (F(\mathbf{\hat{n}_N} \times \mathbf{\hat{n}_M} \parallel \mathbf{\hat{n}_O}) + F(\mathbf{\hat{n}_O} \times \mathbf{\hat{n}_M} \parallel \mathbf{\hat{n}_N}) + F(\mathbf{\hat{n}_O} \times \mathbf{\hat{n}_N} \parallel \mathbf{\hat{n}_M}))/3.$$

Angle

Figure of merit for an angle constraint (angle between faces M and N is θ):

- $\bullet~v=$ vector in plane of $\mathbf{\hat{n}_N}$ and $\mathbf{\hat{n}_M}$, with angle between $\mathbf{\hat{n}_N}$ and $v=\theta$,
- $F(ang(N, M, \theta)) = F(\mathbf{v} \parallel \hat{\mathbf{n}}_{\mathbf{M}}).$

Mirror

Figure of merit for a mirror constraint (reflection through mirror chain C moves face N to the current location of face M):

- $ullet \mathbf{v} = \hat{\mathbf{n}}_{\mathbf{N}}$,
- reflect \mathbf{v} through mirror chain C,
- $F(ref(C, N, M)) = F(\mathbf{v} \parallel \hat{\mathbf{n}}_{\mathbf{M}}).$

Rotation

Figure of merit for a rotation constraint (rotation of angle ρ about a perpendicular axis through the centre of face R rotates face N to the current location of face M):

- $\mathbf{v} = \hat{\mathbf{n}}_{\mathbf{N}}$
- ullet rotate ${f v}$ by angle ho about an axis through the centre of R,
- $F(rot(R, N, M)) = F(\mathbf{v} \parallel \hat{\mathbf{n}}_{\mathbf{M}}).$

Labelling

Figure of merit for line labels in a labelling (see Page 74):

 $(1 - \frac{E_o}{E_t})^{k_E}$, where E_o is the number of occluding lines, E_t the total number of lines, and k_E a tuning constant.

Figure of merit for a vertex being complete:

$$F(complete(v)) = (e+1-n)/(x+1-n)$$

where e is the current number of edges meeting at the vertex, x is the maximum number of edges possible at the vertex, and n is the minimum number of edges possible at the vertex.

Note that this could be improved: it should (but does not) take account of the frequency of different underlying vertex types.

Completeness

Figure of merit for the hypothesis that an object is complete

$$F(complete) = (1/max(1,A)) * \Pi_0^{i=n_V}(F(complete)(i))$$

Figure of merit for vertex completeness of a vertex with E edges, where the set of underlying vertex types suggests a range of edges E_{min} – E_{max} :

- 0 if $E < E_{min}$
- $1 T_e(E_{max} E)$, if this is greater than 0
- 0 otherwise

Y-junction Obtuse

Figure of merit for a Y-junction being obtuse

$$F(Yobtuse) = \prod_{\text{line } 1}^{\text{line } 3} (1.0 \text{ if angle is obtuse}, F(A \perp B) \text{ otherwise}).$$

Subgraph Connection

Figures of merit for adding topology to connect two vertices A and B should be multiplied by a figure of merit S_{AB} for them being in the same subgraph, as follows:

- set $S_{AB} = 1$
- if there is more than one subgraph in the object and A and B are in different subgraphs
 - multiply S_{AB} by S_y
 - if the subgraph types are different (e.g. one is a pocket, the other is a boss) multiply S_{AB} by S_z

New Edge of Given Length

Given N, the length of the new edge, S, the length of the shortest edge in the object, and L, the length of the longest edge in the object,

- if N < S, merit is $(N/S)^{T_s}$
- if N > L, merit is $T_r(L/N)^{T_s}$
- otherwise, merit is $(N(1-T_r)+ST_r-L)/(S-L)$

Appendix E

Junction Catalogue Illustrations

This Appendix illustrates the trihedral and tetrahedral junction labels recognised by RIBALD, and the relationship between junction label and underlying vertex type. Each section shows differing views of the same object, with correspondingly different junction labels for a chosen vertex.

The trihedral catalogue [14, 56] is well-established. It can be seen from the illustrations that all entries in the tetrahedral catalogue are correct; both the methodology by which it was produced (see Chapter 4.3) and practical experience suggest that it is also complete.

The titles refer to the underlying vertex type, so (for example) "All Convex" means that all edges at the vertex are convex; the lines at the junction may be convex, occluding or even invisible, depending on viewpoint.

E.1 Trihedral Catalogue

E.1.1 Yccc, Wbca, Lba

Trihedral: All Edges Convex. The illustrative solid is a single cube.

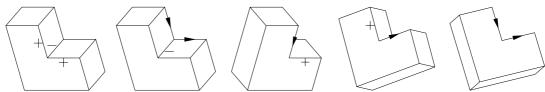






E.1.2 Wcdc, Yabd, Lac, Lcb, Lab

Trihedral: Two Edges Convex, One Concave. The illustrative solid is built from three cubes.



E.1.3 Wdcd,Lbd,Lda

Trihedral: One Edge Convex, Two Concave. The illustrative solid is built from five cubes in two layers.







E.1.4 Yddd

Trihedral: All Edges Concave. The illustrative solid is built from seven cubes in two layers.

E.2 Tetrahedral Catalogue

$E.2.1 \quad Xcccc, Mbcca, Wbca, Lba$

X-Type Tetrahedral: All Edges Convex. These are illustrated by a single oojit.





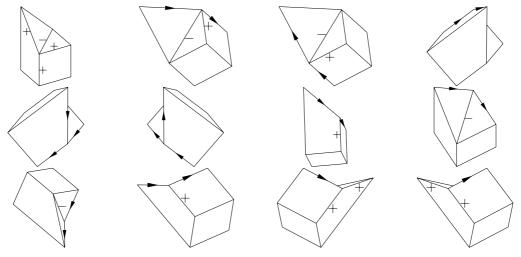




E.2.2 Xcccd etc

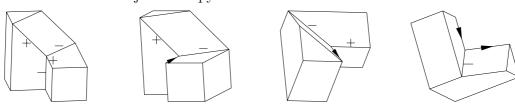
Xcccd, Mbcda, Mbdca, Lba, Wbaa, Wbba, Wbca, Wbda, Yabd, Yabc, Yacc, Ybcc

X-Type Tetrahedral: Three Edges Convex, One Concave. The illustrative solid is built from a cube and a pyramid.



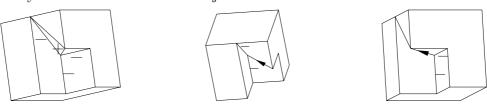
E.2.3 Xcdcd, Yacd, Ybdc, Yabd

X-Type Tetrahedral: Two Edges Convex, Two Concave, Alternating. The illustrative solid is built from a base layer of three cubes forming an L-shape, to which are added an oojit and a pyramid.



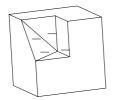
E.2.4 Xcddd,Yadd,Ybdd

X-Type Tetrahedral: One Edge Convex, Three Concave. The illustrative solid is built from a base layer of four cubes forming a square, to which are added a second layer of two cubes and an oojit.



E.2.5 Xdddd

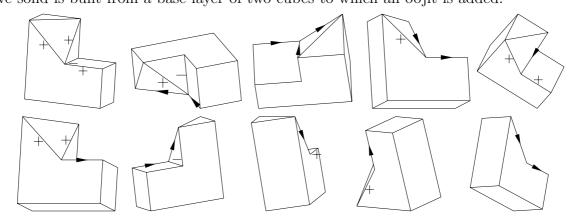
X-Type Tetrahedral: All Edges Concave. The illustrative solid is built from a base layer of four cubes forming a square, to which are added a second layer of three cubes and a triangular pyramid.



E.2.6 Mccdc etc

Mccdc, Xabcd, Yaab, Wcab, Wcac, Wccb, Yabd, Lac, Lcb, Lab

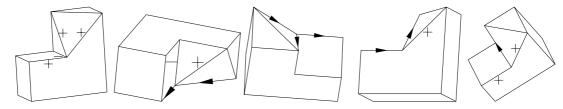
M-Type Tetrahedral: One Edge Concave, Three Convex. The illustrative solid is built from a base layer of two cubes to which an oojit is added.

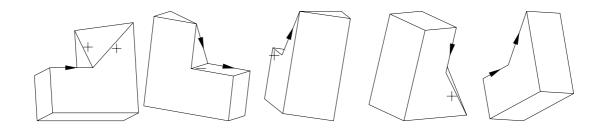


E.2.7 Mcdcc etc

Mcdcc, Xabdc, Yabb, Wabc, Wcbc, Wacc, Yabd, Lcb, Lac, Lab

M-Type Tetrahedral: One Edge Concave, Three Convex, Mirrored. The illustrative solid is the mirror image of the previous one—the resulting junction labels are different.

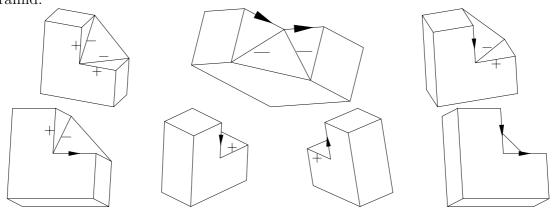




E.2.8 Mcddc, Xabdd, Wadc, Wcdb, Lac, Lcb, Lab

M-Type Tetrahedral: Two Concave Edges Between Two Convex Edges.

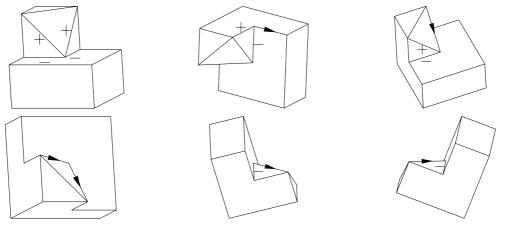
The illustrative solid is built from a single layer of three cubes and a triangular pyramid.



E.2.9 Mdccd, Wbcd, Wdca, Lba, Lbd, Lda

M-Type Tetrahedral: Two Convex Edges Between Two Concave Edges.

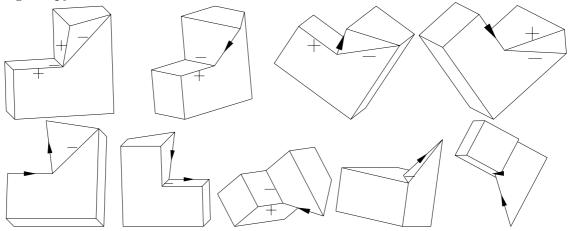
The illustrative solid is built from a layer of four cubes to which one oojit is added.



E.2.10 Mcdcd etc

Mcdcd, Wcda, Wcbd, Wacd, Wabd, Yabd, Yacd, Lbd, Laa

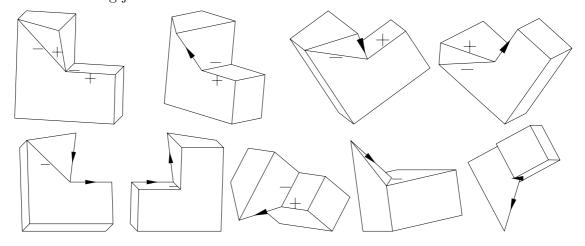
M-Type Tetrahedral: Two Convex Edges, Two Concave, Alternating Convexity. The illustrative solid is built from a layer of three cubes to which one triangular pyramid is added.



E.2.11 Mdcdc etc

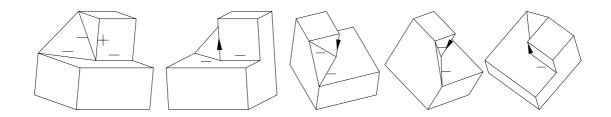
Mdcdc, Wbdc, Wdac, Wdcb, Wdab, Yabd, Ybdc, Lda, Lbb

M-Type Tetrahedral: Two Convex Edges, Two Concave, Alternating Convexity, Mirrored. The illustrative solid is the mirror image of the previous one—the resulting junction labels are different.



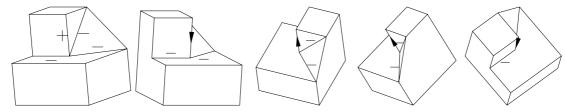
E.2.12 Mddcd, Wdbd, Wdda, Yadd, Lbd

M-Type Tetrahedral: One Edge Convex, Three Concave. The illustrative solid is built from a layer of four cubes to which one cube and a triangular pyramid are added.



E.2.13 Mdcdd, Wdad, Wbdd, Ybdd, Lda

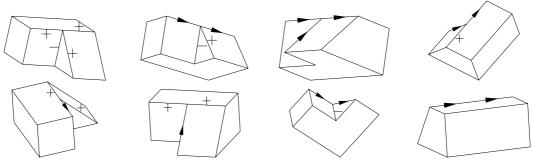
M-Type Tetrahedral: One Edge Convex, Three Concave, Mirrored. The illustrative solid is the mirror image of the previous one—the resulting junction labels are different.



E.2.14 Kcccd etc

Kcccd, Kabcd, Taba, Tbca, Tbcc, Tcca, Yabd, Iab

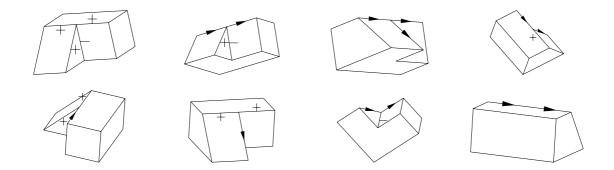
K-Type Tetrahedral: One Edge Concave, Three Convex. The illustrative solid is built from two cubes and a wedge.



E.2.15 Kccdc

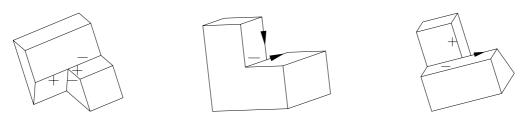
Kccdc, Kabdc, Tabb, Tcab, Tcac, Tccb, Yabd, Iab

K-Type Tetrahedral: One Edge Concave, Three Convex, Mirrored. The illustrative solid is the mirror image of the previous one—the resulting junction labels are different.



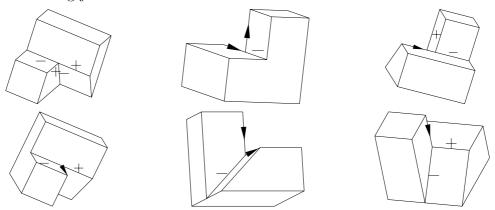
E.2.16 Kcdcd, Tbda, Tbdc, Tcda, Yabd, Ybdc

K-Type Tetrahedral: Two Convex, Two Concave, Alternating. The illustrative solid is built from a base layer of three cubes and a triangular prism, to which is added a single cube.



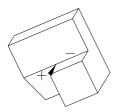
E.2.17 Kdcdc, Tdab, Tdac, Tdcb, Yabd, Yacd

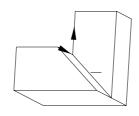
K-Type Tetrahedral: Two Edges Convex, Two Concave, Alternating Convexity, Mirrored. The illustrative solid is the mirror image of the previous one—the resulting junction labels are different.

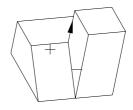


E.2.18 Kcdcd*

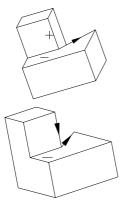
Wdcb, Tbda, Tbdc, Wdab, Lda, Lbb

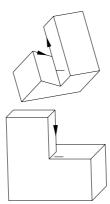


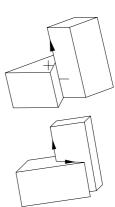




K-Type Tetrahedral: Two Edges Convex, Two Concave, Some Occluded. The illustrative solid is again built from a base layer of three cubes and a wedge, to which is added a single cube. The difference is the orientation of the wedge, which is such that it is impossible to see all four edges meeting at the central vertex whatever the viewpoint.



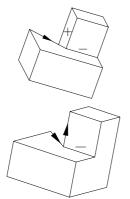


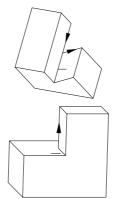


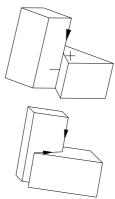
E.2.19 Kdcdc*

Wacd, Tdab, Tdac, Wabd, Lbd, Laa

K-Type Tetrahedral—Two Edges Convex, Two Concave, Some Occluded, Mirrored. The illustrative solid is the mirror image of the previous one—the resulting junction labels are different.

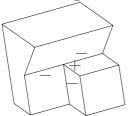


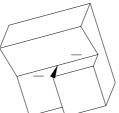


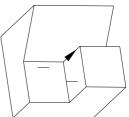


E.2.20 Kddcd, Tdda, Ybdd

K-Type Tetrahedral: One Edge Convex, Three Concave. The illustrative solid is built from a base layer of four cubes, to which is added a single cube and two collinear wedges.

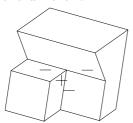


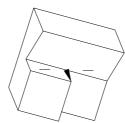


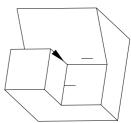


E.2.21 Kdcdd, Tddb, Yadd

K-Type Tetrahedral: One Edge Convex, Three Concave, Mirrored. The illustrative solid is the mirror image of the previous one—the resulting junction labels are different.







Appendix F

Geometric Analysis

F.1 Rotation Axis from Start and End Points and Angle

The method presented in Chapter 11.6 requires iterative estimation of face normals given a constraint and the face normal values after the previous iteration. In most cases, geometric methods for making these estimates are either straightforward or available in the literature. However, the problem of obtaining an estimate for a rotation axis given a constraint which rotates one face to another, and the centre points of the start and end rotating faces, is less straightforward.

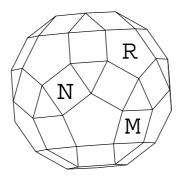


Figure F.1: Rotation about Unknown Axis

Formally, the problem is: given a rotation constraint $C_r(R, N, M, \rho)$ which states that a rotation of an angle ρ about an axis normal to and through the centre of face R moves a vector normal to and through the centre of face N to the position which (prior to the rotation) was that of a vector normal to and through the centre of face M, and given values of face normals $\hat{\mathbf{n}}_{\mathbf{M}}$ and $\hat{\mathbf{n}}_{\mathbf{N}}$, estimate the value of face normal $\hat{\mathbf{n}}_{\mathbf{R}}$.

Consider R, M and N as points on the Gaussian sphere, and add point D, a point on the sphere mid-way along the shortest curve between M and N, and point 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 $\hat{\mathbf{n}}_{\mathbf{R}} = \hat{\mathbf{G}} \sin \sigma \pm \hat{\mathbf{D}} \cos \sigma$ (rotation about either axis will move N to M).

To find σ , take the cosine rule for spherical triangles [95, 186]: 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 location 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

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

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

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.

RIBALD therefore estimates $(\hat{\mathbf{n}}_{\mathbf{R}})_{i+1}$ as follows:

- Set $\hat{\mathbf{D}} = \hat{\mathbf{((\hat{n}_M)}_i + (\hat{n}_N)_i)}$;
- ullet Set $\mathbf{\hat{G}}=\hat{\ }(\left(\mathbf{\hat{n}_{M}}\right)_{i} imes\mathbf{\hat{D}})$;
- Set $\cos \phi = (\hat{\mathbf{n}}_{\mathbf{M}})_i \cdot (\hat{\mathbf{n}}_{\mathbf{N}})_i$;
- Set $\cos^2 \sigma = 2(\cos \rho \cos \phi)/((\cos \phi 1)(\cos \rho 1));$

- Estimate the two values $(\hat{\mathbf{n}}_{\mathbf{R}})_{i+1} = \hat{\mathbf{G}} \sin \sigma \pm \hat{\mathbf{D}} \cos \sigma$;
- \bullet Choose the nearer estimate of $\left(\hat{\mathbf{n}}_{\mathbf{R}}\right)_{i+1}$ to $\left(\hat{\mathbf{n}}_{\mathbf{R}}\right)_{i}$

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

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

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 $\hat{\mathbf{n}}_{\mathbf{R}}$, 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 $(\hat{\mathbf{n}}_{\mathbf{R}})_{i+1}$. In some test cases, $\hat{\mathbf{n}}_{\mathbf{N}}$ and $\hat{\mathbf{n}}_{\mathbf{M}}$ are accurate and both perpendicular to the true $\hat{\mathbf{n}}_{\mathbf{R}}$, and $(\hat{\mathbf{n}}_{\mathbf{R}})_i$ is inaccurate by a predetermined angle in the range 5° to 30°, and measured the resulting inaccuracy in $(\hat{\mathbf{n}}_{\mathbf{R}})_{i+1}$ 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, the geometric approach is preferred.

Angle ρ	Error in $(\hat{\mathbf{n}}_{\mathbf{R}})_i$	Geometric Method	Quaternion Method
60°	5°	0°	2.50°
60°	10°	0°	4.98°
60°	15°	0°	7.44°
60°	20°	0°	9.85°
60°	25°	0°	12.21°
60°	30°	0°	14.50°
72°	5°	0°	3.09°
72°	10°	0°	6.14°
72°	15°	0°	9.15°
72°	20°	0°	12.07°
72°	25°	0°	14.89°
72°	30°	0°	17.60°
90°	5°	0°	4.98°
90°	10°	0°	9.85°
90°	15°	0°	14.51°
90°	20°	0°	18.88°
90°	25°	0°	22.91°
90°	30°	0°	26.57°
120°	5°	0°	4.99°
120°	10°	0°	9.93°
120°	15°	0°	14.75°
120°	20°	0°	19.43°
120°	25°	0°	23.91°
120°	30°	0°	28.19°
180°	5°	0°	5.00°
180°	10°	0°	10.00°
180°	15°	0°	15.00°
180°	20°	0°	20.00°
180°	25°	0°	25.00°
180°	30°	0°	30.00°

Table F.1: Errors for Rotation Axis Prediction

Bibliography

- [1] S.Bagali and W.N.Waggenspack Jr. A Shortest Path Approach to Wireframe to Solid Model Conversion. In ed. C.M.Hoffmann and J.Rossignac, SMA 95: Proceedings of the Third Symposium on Solid Modeling and Applications, 339– 350, ACM Press, 1995.
- [2] H.G.Barrow and J.M.Tenenbaum. Interpreting Line Drawings as Three-Dimensional Surfaces. *Artificial Intelligence* 17(1–3), 75–116, 1981.
- [3] L.Bauer. Elimination with Weighted Row Combinations for Solving Linear Equations and Least Squares Problems. Contribution I/9 in ed. J.H.Wilkinson and C.Reinsch, *Handbook for Automatic Computation*, volume II of *Linear Algebra*, 119-133, Springer-Verlag, 1971
- [4] W.Bouma, I.Fudos, C.M.Hoffmann, J.Cai and R.Paige. Geometric Constraint Solver. *Computer-Aided Design* 27(6), 487–501, 1995.
- [5] A.Bowyer and J.Woodwark. *Introduction to Computing with Geometry*, Information Geometers, 1993.
- [6] R.P.Brent. Algorithms for Minimization without Derivatives, Prentice-Hall, 1973.
- [7] K.W.Brodlie. Chapter III.1. in ed. D.A.H.Jacobs, *The State of the Art in Numerical Analysis*, 3–6, London Academic Press, 1977.
- [8] C.G.Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of Computation* 19, 577–593, 1965.

- [9] B.Buchberger. Gröbner Bases: an Algorithmic Method in Polynomial Ideal Theory. In ed. N.K.Bose, Recent Trends in Multidimensional System Theory, 184–232, D Reidel, 1985.
- [10] G.Butlin and C.Stops. CAD Data Repair. In *Proceedings of 5th International Meshing Roundtable*, Sandia National Laboratories, 7–12, 1996.
- [11] T.By. Line Labelling by Meta-programming. Technical Note CS-97-07, Sheffield University, 1997.
- [12] Capalex. Catalogue of Standard Profiles, Capital Aluminium Extrusions Ltd, 2000.
- [13] I.Chakravarty. A Generalised Line and Junction Labelling Scheme With Applications To Scene Analysis. *IEEE Pattern Analysis and Machine Intelligence* 1(2), 202–205, 1979.
- [14] M.B.Clowes. On Seeing Things. Artificial Intelligence 2, 79–116, 1970.
- [15] P.Company, J.M.Gomis and M.Contero. Geometrical Reconstruction from Single Line Drawings using Optimization-based Approaches. In ed. V.Skala, Proceedings of WSCG99 7th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media'99, University of West Bohemia, Plzen, Vol. II, 361–368, 1999.
- [16] P.Company, J.M.Gomis and M.Contero. An Optimization-Based Algorithm to Reconstruct 3D Models from Single Line Drawings. In *II Seminario Italo-Spagnolo*, Vol. 2, 952–958, 1998.
- [17] 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.
- [18] J.Corney. 3D Modeling with the ACIS Kernel and Toolkit, Wiley, 1997.
- [19] P.R.Cromwell. *Polyhedra*, Cambridge University Press, 1997.
- [20] H.T.Davey and R.J.Wilkins. Engineering Drawing, MacDonald and Co., 1952.

- [21] E.W.Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* I, 269–271, 1959.
- [22] M.Dorigo, V.Maniezzo and A.Colorni. Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man and Cybernetics* 26B(1), 29–41, 1996.
- [23] S.W.Draper. Reasoning about Depth in Line-Drawing Interpretation. PhD Thesis, Sussex University, 1980.
- [24] S.W.Draper. The Use of Gradient and Dual Space in Line-Drawing Interpretation. *Artificial Intelligence* 17, 461–508, 1981.
- [25] L.Eggli, C-Y.Hsu, B.Brüderlin and G.Elber. Inferring 3D Models from Freehand Sketches and Constraints. *Computer-Aided Design* 29(2), 101–112, 1997.
- [26] G.Falk. Interpretation of Imperfect Line Data as a Three-Dimensional Scene. Artificial Intelligence 3, 101–144, 1972.
- [27] G.E.Farin and D.Hansford. The Geometry Toolbox for Graphics and Modeling, A.K.Peters, 1998.
- [28] S.Fortin. The Graph Isomorphism Problem. Technical Report 96-20, University of Alberta, 1996.
- [29] X.S.Gao and S.C.Chou. Solving Geometric Constraint Systems I: A Global Propagation Approach. *Computer-Aided Design* 30(1), 47–54, 1998.
- [30] X.S.Gao and S.C.Chou. Solving Geometric Constraint Systems II: A Symbolic Approach and Decision of Reconstructability. *Computer-Aided Design* 30(2), 115–122, 1998.
- [31] M.R.Garey and D.S.Johnson. *Computers and Intractability*, Freeman, San Francisco, 1979.
- [32] J.X.Ge, S.C.Chou and X.S.Gao. Geometric Constraint Satisfaction using Optimization Methods. *Computer-Aided Design* 31(14), 867–879, 1999.
- [33] A.Gibbons. Algorithmic Graph Theory, Cambridge University Press, 1985.

- [34] J.W.Gibbs. Elements of Vector Analysis. Technical Report, Yale University, 1881.
- [35] D.E.Goldberg. Computer-Aided Gas Pipeline Operation using Genetic Algorithm and Rule Learning. PhD Thesis, University of Michigan, 1989.
- [36] R.Goldman. Intersection of Three Planes. In ed. A.S.Glassner, Graphics Gems I, 305, Academic Press, 1990.
- [37] M.Gondran and M.Minoux. Graphs and Algorithms, Wiley-Interscience, 1986.
- [38] I.J.Grimstead. Interactive Sketch Input of Boundary Representation Solid Models. PhD Thesis, Cardiff University, 1997.
- [39] I.J.Grimstead and R.R.Martin. Incremental Line Labelling for Sketch Input of Solid Models. *Computer Graphics Forum* 15(2), 155–166, 1996.
- [40] J.Grosjean, T.Stein and S.Coquillart. Incremental Conversion of 3D Wire Frame Models to Polygonal Surface Models. In *Proceedings of the 8th International Conference in Central Europe on Computer Graphics, Visualisation and Interactive Digital Media 2000*, 2000.
- [41] S.K.Gupta, W.C.Regli and D.S.Nau. Manufacturing Feature Instances: Which Ones to Recognize? In ed. C.M.Hoffmann and J.Rossignac, SMA 95: Proceedings of the Third Symposium on Solid Modeling and Applications, 141–152, ACM Press, 1995.
- [42] A.Guzman. Decomposition of a Visual Scene into Three-Dimensional Bodies. In ed. A.Grasselli, AFIPS (American Federation of Information Processing Societies) Proceedings of Fall Joint Computer Conference, Vol. 33, 291–304, Academic Press, New York, 1968.
- [43] A.Guzman. Computer Recognition of Three-Dimensional Objects in a Visual Scene. Tech Rept MAC-TR-59, M.I.T., Cambridge USA, 1968.
- [44] J.H.Han. Survey of Feature Research. Technical Report IRIS-96-346, University of Southern California Institute for Robotics and Intelligent Systems, 1996.

- [45] J.H.Han. On Multiple Interpretations. In ed. C.Hoffmann and W.Bronsvort, Proceedings: Fourth Symposium on Solid Modeling and Applications, 311–321, ACM Press, 1997.
- [46] J.H.Han and M.J.Kang. Geometric Reasoning and Search Algorithms for Manufacturing Cost Optimization. In ed. H.I.Choi, M.S.Kim, K.W.Lee and R.R.Martin, 1st Korea-UK Joint Workshop on Geometric Modeling and Computer Graphics, 47–54, Kyung Moon Publishers, 2000.
- [47] J.H.Han and A.A.G.Requicha. Integration of Feature Based Design and Feature Recognition. *Computer Aided Design* 20(5), 393–403, 1997.
- [48] M.Herman and T.Kanade. The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images. In ed. M.A.Fischler and O.Firschein, Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, 471–482, Kaufmann, 1987.
- [49] G.E.Hinton. *Relaxation and its Role in Vision*. PhD Thesis, Edinburgh University, 1978.
- [50] C.M.Hoffman and K.J.Kim. Towards Valid Parametric CAD Models. Computer-Aided Design 33(1), 81–90, 2001.
- [51] D.D.Hoffman and W.A.Richards. Parts of Recognition. Cognition 18, 65–96, 1984.
- [52] J.H.Holland. Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
- [53] J.E.Hopcroft and R.E.Tarjan. Dividing a Graph into Triconnected Components. SIAM Journal of Computing 2(3), 135–158, 1973.
- [54] J.E.Hopcroft and R.E.Tarjan. A V log V Algorithm for Isomorphism of Triconnected Planar Graphs. Journal of Computer and System Sciences 7, 323–331, 1973.
- [55] J.E.Hopcroft and J.K.Wong. Linear Time Algorithm for Isomorphism of Planar Graphs. In Proceedings, 6th Annual ACM Symposium on Theory of Computing, 172–184, 1974.

- [56] D.A.Huffman. Impossible Objects as Nonsense Sentences. In ed. B.Meltzer and D.Michie, *Machine Intelligence*, Vol. 6, 295–323, Edinburgh University Press, 1971.
- [57] D.A.Huffman. A Duality Concept for the Analysis of Polyhedral Scenes. In ed. E.W.Elcock and D.Michie, *Machine Intelligence*, Vol. 8, 475–492, Ellis Horwood, 1977.
- [58] D.A.Huffman. Realizable Configurations of Lines in Pictures of Polyhedra. In ed. E.W.Elcock and D.Michie, *Machine Intelligence*, Vol. 8, 493–509, Ellis Horwood, 1977.
- [59] D.L.Jenkins. The Automatic Interpretation of Two-Dimensional Freehand Sketches. PhD Thesis, University of Wales College of Cardiff, 1992.
- [60] X.Y.Jiang and H.Bunke. A Simple and Efficient Algorithm for Determining the Symmetries of Polyhedra. CVGIP Graphics Models and Image Processing 54(1), 91–95, 1992.
- [61] X.Jiang, K.Yu and H.Bunke. Detection of Rotational and Involutional Symmetries and Congruity of Polyhedra. *Visual Computing* 12(4), 193–201, 1996.
- [62] T.Kanade. A Theory of Origami World. Artificial Intelligence 13, 279–311, 1980.
- [63] T.Kanade. Recovery of the Three-Dimensional Shape of an Object from a Single View. *Artificial Intelligence* 17, 409–460, 1981.
- [64] K.Kanatani. Group-Theoretical Methods in Image Understanding, Number 20 in Springer Series in Information Sciences, Springer-Verlag, 1990.
- [65] D.Kapur, T.Saxena and L.Yang. Algebraic and Geometric Reasoning using Dixon Resultants. In Proc. ACM International Symposium on Symbolic and Algebraic Computation, 99–107, 1994.
- [66] S.F.A.Kettle. Symmetry and Structure, Wiley, 1985.
- [67] S.Kirkpatrick, C.D.Gelatt and M.P.Vecchi. Optimization by Simulated Annealing. Science 220, 671–680, 1983.

- [68] L.M.Kirousis. Effectively Labelling Planar Projections of Polyhedra. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(2), 123–130, 1990.
- [69] L.M.Kirousis and C.H.Papadimitriou. The Complexity of Recognising Polyhedral Scenes. *Journal of Computer Systems Science* 37, 14–38, 1988.
- [70] J.Kobler, U.Schoning and J.Toran. *The Graph Isomorphism Problem: Its Structural Complexity*, Birkhauser Boston Inc, Boston MA, 1993.
- [71] K.Kondo. PIGMOD: Parametric and Interactive Geometric Modeller for Mechanical Design. *Computer-Aided Design* 22(10), 633–644, 1990.
- [72] K.Kondo. Algebraic Method for Manipulation of Dimensional Relationship in Geometric Models. *Computer-Aided Design* 24(3), 141–147, 1992.
- [73] G.A.Kramer. Using Degrees of Freedom Analysis to Solve Geometric Constraint Systems. In ed. J.Rossignac and J.Turner, Proc. Symposium on Solid Modeling Foundations and CAD/CAM Applications, 371–378, ACM Press, 1991.
- [74] G.A.Kramer. Solving Geometric Constraint Systems, MIT Press, 1992.
- [75] A.V.Kumar and L.Yu. Sequential Constraint Imposition for Dimension-Driven Solid Models. *Computer-Aided Design* 33, 475–486, 2001.
- [76] V.Kumar. Algorithms for Constraint Satisfaction Problems: A Survey. AI Magazine 13(1), 32–44, 1992.
- [77] D.Lamb and A.Bandopadhay. Interpreting a 3D Object From a Rough 2D Line Drawing. In ed. A.E.Kaufman, Proceedings of the First IEEE Conference on Visualization '90, 59–66, IEEE, 1990.
- [78] F.C.Langbein, B.I.Mills, A.D.Marshall and R.R.Martin. Recognizing Geometric Patterns for Beautification of Reconstructed Solid Models. In *Proc. International Conference on Shape Modelling and Applications, Genova, Italy*, 10–19, IEEE Computer Society, Los Alamitos, CA, USA, 2001.

- [79] F.C.Langbein, B.I.Mills, A.D.Marshall and R.R.Martin. Finding Approximate Shape Regularities in Reverse Engineered Solid Models Bounded by Simple Surfaces. In ed. D.C.Anderson and K.Lee, Proc. 6th ACM Symp. Solid Modelling and Applications, Ann Arbor, MI, USA, 206–215, 2001.
- [80] R.S.Latham. Combinatorial Algorithms for the Analysis and Satisfaction of Geometric Constraints. PhD Thesis, Department of Computer Science, Brunel University, 1996.
- [81] R.S.Latham and A.E.Middleditch. Connectivity Analysis: A Tool for Processing Geometric Constraints. *Computer-Aided Design* 28(11), 917–928, 1996.
- [82] D.Lazard. Resolution of Polynomial Systems. In ed. X.S.Gao and D.M.Wang, Computer Mathematics: Proceedings of the Fourth Asian Symposium (ASCM 2000), 1–8, World Scientific, 2000.
- [83] Y.G.Leclerc and M.A.Fischler. An Optimization-Based Approach to the Interpretation of Single Line Drawings as 3D Wire Frames. *International Journal of Computer Vision* 9(2), 113–136, 1992.
- [84] K.W.Lee. Principles of CAD/CAM/CAE Systems, Addison Wesley, 1999.
- [85] S.J.Lee, R.M.Haralick and M.C.Zhang. Understanding Objects with Curved Surfaces from a Single Perspective View of Boundaries. *Artificial Intelligence* 26, 145–169, 1985.
- [86] W.Lee, P.Kalra and N.Magnenat-Thalmann. Model Based Face Reconstruction for Animation. In *Proc. Multimedia Modeling (MMM'97)*, 323–338, World Scientific, Singapore, 1997.
- [87] D.Levy. The Chess Computer Handbook, Batsford, 1984.
- [88] Y.T.Li, S.M.Hu and J.G.Sun. A Constructive Approach to Solving 3D Geometric Constraint Systems using Dependence Analysis. *Computer-Aided Design* 34(2), 97–108, 2002.

- [89] Y.T.Li, S.M.Hu and J.G.Sun. On the Numerical Redundancies of Geometric Constraint Systems. In ed. H.Suzuki, A.Rockwood and L.P.Kobbelt, *Ninth Pacific Conference on Computer Graphics and Applications (PG'01)*, 118–123, IEEE Comp Soc Press, 2001.
- [90] H.Lipson. Computer Aided 3D Sketching for Conceptual Design. PhD Thesis, Technion-Israel Institute for Technology, Haifa, 1998.
- [91] 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.
- [92] H.Lipson and M.Shpitalni. An Interface for 3D Conceptual Design Based on Freehand Sketching. In *IFIP WG5.2 Workshop on Geometric Modeling in Computer-Aided Design*, 139–148, 1996.
- [93] J.Z.Liu and Y.T.Lee. A Graph-Based Method for Face Identification from a Single 2D Wireframe Object. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(10), 1106–1119, 2001.
- [94] E.H.Lockwood and R.H.Macmillan. *Geometric Symmetry*, Cambridge University Press, 1978.
- [95] W.H.Macaulay. Solid Geometry, Cambridge University Press, 1930.
- [96] A.K.Mackworth. Interpreting Pictures of Polyhedral Scenes. *Artificial Intelligence* 4, 121–137, 1973.
- [97] A.K.Mackworth. On the Interpretation of Drawings as Three-Dimensional Scenes. D.Phil Thesis, University of Sussex, 1974.
- [98] A.K.Mackworth. Consistency in Networks of Relations. *Artificial Intelligence* 8, 99–118, 1977.
- [99] J.Malik. Interpreting Line Drawings of Curved Objects. Technical Report STAN-CS-86-1099, Computer Science Department, Stanford University, 1985.
- [100] J.Malik. Interpreting Line Drawings of Curved Objects. *International Journal of Computer Vision* 2(1), 73–103, 1987.

- [101] T.Marill. Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects. International Journal of Computer Vision 6(2), 147–161, 1991.
- [102] D.W.Marquardt. An Algorithm for Least-squares Estimation of Nonlinear Parameters. SIAM Journal on Applied Mathematics 11, 431–441, 1963.
- [103] D.Marr. Representing Visual Information. In ed. A.Hanson and E.M.Riseman, Computer Vision Systems, 61–80, Academic Press NY, 1978.
- [104] R.R.Martin and D.Dutta. Tools for Asymmetry Rectification in Shape Design.

 Journal of Systems Engineering 6, 98–112, 1996.
- [105] H.Mayer. Automatic Object Extraction from Aerial Imagery—A Survey Focusing on Buildings. *Computer Vision and Image Understanding* 74(2), 138–149, 1999.
- [106] J.A.McHugh. Algorithmic Graph Theory, Prentice-Hall, 1990.
- [107] S.Meeran and J.M.Taib. A Generic Approach to Recognising Isolated, Nested and Interacting Features from 2D Drawings. Computer-Aided Design 31(14), 891–910, 1999.
- [108] N.Metropolis, A.Rosenbluth, M.Rosenbluth, A.Teller and E.Teller. Simulated Annealing. *Journal of Chemical Physics* 21, 1087–1092, 1953.
- [109] J.S.Mill. An Examination of Sir William Hamilton's Philosophy and of The Principal Philosophical Questions Discussed in his Writings. Volume IX of Collected Works of John Stuart Mill, University.of.Toronto.Press, 1979.
- [110] B.I.Mills. Private Communication. 2001.
- [111] B.I.Mills, F.C.Langbein, A.D.Marshall and R.R.Martin. Approximate Symmetry Detection for Reverse Engineering. In ed. D.C.Anderson and K.Lee, Proc. 6th ACM Symp. Solid Modelling and Applications, Ann Arbor, MI, USA, 241–248, 2001.
- [112] J.Mitani, H.Suzuki and F.Kimura. 3D SKETCH: Sketch Based Model Reconstruction and Rendering. In ed. U.Cugini and M.J.Wozny, *From Geometric*

- Modeling to Shape Modeling, IFIP TC5 WG5.2 Seventh Workshop on Geometric Modeling: Fundamentals and Applications, October 2-4, 2000, Parma, Italy, 85–98, Kluwer, 2001.
- [113] R.Mohan and R.Nevatia. Using Perceptual Organisation to Extract 3D Structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11(11), 1121–1139, 1989.
- [114] R.O.Myers. Genetic Algorithms for Ambiguous Labelling Problems. PhD Thesis, University of York, 1999.
- [115] I.V.Nagendra and U.G.Gujar. 3D Objects From 2D Orthographic Views. Computers and Graphics 12(1), 111–114, 1988.
- [116] L.A.Necker. Observations on Some Remarkable Optical Phenomena Seen in Switzerland. *Philos. Mag. Third Series* 1(5), 329–337, 1832.
- [117] J.A.Nelder and R.Mead. A Simplex Method for Function Minimization. Computer Journal 7, 308–313, 1965.
- [118] P.Norvig. Paradigms of Artificial Intelligence Programming, Morgan Kaufmann, 1992.
- [119] J.C.Owen. Algebraic Solution for Geometry from Dimensional Constraints. In ed. J.Rossignac and J.Turner, Proc. Symposium on Solid Modeling Foundations and CAD/CAM Applications, 397–407, ACM Press, 1991.
- [120] 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.
- [121] S.Parry-Barwick and A.Bowyer. Symmetry Analysis and Geometric Modelling. In ed. K.K.Fung and A.Ginige, Dicta 93 Proc. Digital Image Computing Conference, Vol. 1, 39–46, Australian Pattern Recognition Society, 1993.
- [122] T.Pavlidis and C.J.van Wyk. An Automatic Beautifier for Drawings and Illustrations. *Computer Graphics* 19(3), 225–234, 1985.

- [123] J.Pearl. Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley, 1984.
- [124] L.S.Penrose and R.Penrose. Impossible Objects: A Special Type of Visual Illusion. *British Journal of Psychology* 49, 31–33, 1958.
- [125] D.N.Perkins. Cubic Corners. Quarterly Progress Report 89, 207–214, MIT Research Laboratory of Electronics, 1968.
- [126] D.N.Perkins. How Good a Bet is Good Form? Perception 5, 393–406, 1976.
- [127] B.T.Phong. Illumination for Computer-Generated Images. *CACM* 18(6), 311–317, 1975.
- [128] F.Pickup and M.A.Parker. Engineering Drawing with Worked Examples, Vol. 1, 3rd Edition, Hutchison and Co, 1979.
- [129] F.Pickup and M.A.Parker. Engineering Drawing with Worked Examples, Vol.2, 2nd Edition, Hutchison and Co, 1980.
- [130] A.Piquer. Private Communication. 2001.
- [131] W.H.Press, S.A.Teukolsky, W.T.Vettering and B.P.Flannery. *Numerical Recipes in C, The Art of Scientific Computing*, Second Edition, Cambridge University Press, 1994.
- [132] D.Pugh. Interactive Sketch Interpretation using Arc-labelling and Geometric Constraint Satisfaction. Technical Report CMU-CS-91-181, Carnegie Mellon University, 1991.
- [133] D.Pugh. Designing Solid Objects Using Interactive Sketch Interpretation.

 ACM/SIGGRAPH Computer Graphics—Special Issue: 1992 Symposium on Interactive Computer Graphics 23(2), 117–126, 1992.
- [134] E.Puppo. On the Topological Representation of Line Drawings. *Pattern Recognition Letters* 18, 575–582, 1997.
- [135] A.Z.Qamhiyah, R.D.Venter and B.Benhabib. Geometric Reasoning for the Extraction of Form Features. *Computer-Aided Design* 28(11), 887–903, 1996.

- [136] C.Qin and J.Y.S.Luh. Ambiguity Reduction by Relaxation Labelling. *Pattern Recognition* 27, 165–180, 1994.
- [137] S.F.Qin, D.K.Wright and I.N.Jordanov. From On-Line Sketching to 2D and 3D Geometry: a System Based on Fuzzy Knowledge. *Computer-Aided Design* 32(12), 851–866, 2000.
- [138] S.F.Qin, D.K.Wright and I.N.Jordanov. A Conceptual Design Tool: A Sketch and Fuzzy Logic Based System. *Proceedings of the I MECH E Part B: Journal of Engineering Manufacture* 215, 111–116, 2001.
- [139] L.G.Roberts. Machine Perception of Three-Dimensional Solids. PhD Thesis, MIT, 1963.
- [140] A.Rosenfeld, R.A.Hummel and S.W.Zucker. Scene Labelling by Relaxation Operations. *IEEE Transactions on Systems, Man and Cybernetics* 6, 420–433, 1976.
- [141] P.J.Rousseeuw and A.M.Leroy. Robust Regression and Outlier Detection, Wiley, 1987.
- [142] P.J.Rousseeuw and S.van Aelst. Positive-Breakdown Robust Methods in Computer Vision. *Computing Science and Statistics* 31, 451–460, 1999.
- [143] 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 USA, 1976.
- [144] T.Schiex, H.Fargier and G.Verfaillie. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In ed. C.Mellish, Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95), 631–637, 1995.
- [145] O.G.Selfridge. Pandemonium: A Paradigm for Learning. In Mechanisation of Thought Processes: Proceedings of a symposium held at the National Physical Laboratory, November 1958, Vol. 1, 511–527, HMSO, 1959.
- [146] A.Selinger and R.C.Nelson. A Perceptual Grouping Hierarchy for Appearance-Based 3D Object Recognition. *Computer Vision and Image Understanding* 76(1), 83–92, 1999.

- [147] I.Shimshoni and J.Ponce. Recovering the Shape of Polyhedra using Line-Drawing Analysis and Complex Reflectance Models. In CVPR94, 514–519, IEEE Computer Society Press, 1994.
- [148] Y.Shirai. Three-Dimensional Computer Vision, Springer-Verlag, 1987.
- [149] M.Shpitalni and H.Lipson. Identification of Faces in a 2D Line Drawing Projection of a Wireframe Object. IEEE Trans. Pattern Analysis and Machine Intelligence 18(10), 1000–1012, 1996.
- [150] S.S.P.Shum, W.S.Lau, M.M.F.Yuen and K.M.Yu. Solid Reconstruction from Orthographic Views using 2-Stage Extrusion. Computer-Aided Design 33(1), 91–102, 2001.
- [151] S.S.Sinha and B.G.Schunck. A Two-Stage Algorithm for Discontinuity-Preserving Surface Reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(1), 36–55, 1992.
- [152] M.G.L.Sommerville, D.E.R.Clark and J.R.Corney. Viewer-Centred Feature Recognition. In ed. C.M.Hoffmann and J.Rossignac, Proceedings of the Third ACM Symposium on Solid Modeling and Applications, 125–130, ACM Press, 1995.
- [153] L.A.E.Stevens. Genetic Algorithm to Optimise Line Labelling. MSc Dissertation, Cardiff University, 1994.
- [154] L.Stiller. Multilinear Algebra and Chess Endgames. In ed. R.J.Nowakowski, Games of No Chance: Combinatorial Games at MRSI, 1994, 151–192, Cambridge University Press, 1996.
- [155] B.Stilman. Linguistic Geometry from Search to Construction, Kluwer Academic Publishers, 2000.
- [156] P.F.Sturm and S.J.Maybank. A Method for Interactive 3D Reconstruction of Piecewise Planar Objects from Single Images. In ed. A.Pridmore and D.Elliman, Proc. British Machine Vision Conference, Nottingham, 265–274, 1999.
- [157] K.Sugihara. Picture Language for Skeletal Polyhedra. Computer Graphics and Image Processing 8, 382–405, 1978.

- [158] K.Sugihara. Range Data Analysis Guided by a Junction Dictionary. *Artificial Intelligence* 12(1), 41–69, 1979.
- [159] K.Sugihara. Mathematical Structures of Line Drawings of Polyhedrons— Towards Man-Machine Communication by Means of Line Drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-4(5), 458–469, 1982.
- [160] K.Sugihara. An Algebraic Approach to Shape-from-Image Problems. *Artificial Intelligence* 23, 59–95, 1984.
- [161] K.Sugihara. An $n \log n$ Algorithm for Determining the Congruity of Polyhedra. Journal of Computer and Systems Science 29(1), 36–47, 1984.
- [162] K.Sugihara. A Necessary and Sufficient Condition for a Picture to Represent a Polyhedral Scene. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6(5), 578–586, 1984.
- [163] K.Sugihara. *Machine Interpretation of Line Drawings*, Cambridge, Massachusetts: MIT Press, 1986.
- [164] K.Sugihara. Three-Dimensional Realization of Anomalous Pictures—an Application of Picture Interpretation Theory to Toy Design. *Pattern Recognition* 30(7), 1061–1067, 1997.
- [165] K.Sugihara. Resolvable Representations of Polyhedra. *Discrete and Computational Geometry* 21(2), 243–255, 1999.
- [166] D.A.Turner. The Approximation of Cartesian Co-ordinate Data by Parametric Orthogonal Distance Regression. PhD Thesis, University of Huddersfield, 1999.
- [167] K.Turner. Computer Perception of Curved Objects using a Television Camera. PhD Dissertation, Edinburgh University, 1974.
- [168] UG Solutions, Inc. Parasolid—Powering the Digital Enterprise. http://www.ugsolutions.com/products/parasolid/.
- [169] J.R.Ullman. An Algorithm for Subgraph Isomorphism. *Journal ACM* 23, 504–519, 1976.

- [170] P.A.C. Varley. Computer Modelling of Seasonal Storage of Solar Thermal Energy for District Heating Systems in the UK. MSc Dissertation, Cardiff University 1995.
- [171] P.A.C. Varley and R.R. Martin. A System for Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch. In ed. W. Wang and R.R. Martin, *Proc. GMP 2000*, 13–32, IEEE Press, 2000.
- [172] P.A.C. Varley and R.R. Martin. Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch—Sketch Categorisation and Frontal Geometry. In ed. H.I.Choi, M.S. Kim, K.W. Lee and R.R. Martin, 1st Korea-UK Joint Workshop on Geometric Modeling and Computer Graphics, 113–128, Kyung Moon Publishers, 2000.
- [173] P.A.C. Varley and R.R. Martin. Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch—Topology of Hidden Parts. In ed. H.I. Choi, M.S. Kim, K.W. Lee and R.R. Martin, 1st Korea-UK Joint Workshop on Geometric Modeling and Computer Graphics, 129–144, Kyung Moon Publishers, 2000.
- [174] P.A.C. Varley and R.R. Martin. Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch—Geometric Finishing. In ed. H.I. Choi, M.S. Kim, K.W. Lee and R.R. Martin, 1st Korea-UK Joint Workshop on Geometric Modeling and Computer Graphics, 145–158, Kyung Moon Publishers, 2000.
- [175] P.A.C. Varley and R.R. Martin. The Junction Catalogue for Labelling Line Drawings of Polyhedra with Tetrahedral Vertices. *International Journal of Shape Modelling* 7(1), 23–44, 2001.
- [176] P.A.C. Varley and R.R. Martin. Estimating Depth from Line Drawings. In ed. K.Lee and N.Patrikalakis, Proc. 7th ACM Symposium on Solid Modeling and Applications, SM02, 180–191, ACM Press, 2002.
- [177] P.A.C. Varley and R.R. Martin. Deterministic and Probabilistic Approaches to Labelling Line Drawings of Engineering Objects. Accepted for publication in International Journal of Shape Modelling, 2003.

- [178] P.A.C. Varley. Algorithms supporting Reconstructing Interactive B-Reps to Analyse Line Drawings. Cardiff University, 2002. Download from http://ralph.cs.af.ac.uk/papers/Geometry/varleyphdalgorithms.pdf.
- [179] A. Verroust, F. Schonek and D. Roller. Rule-Oriented Method for Parameterized Computer-Aided Design. *Computer-Aided Design* 24(10), 531–540, 1992.
- [180] T.Vetter and T.Poggio. Symmetric 3D Objects are an Easy Case for 2D Object Recognition. Spatial Vision 8(4), 443–453, 1995.
- [181] D.M.Waltz. Generating Semantic Descriptions from Drawings of Scenes with Shadows. Tech Rept AI-TR-271, M.I.T., Cambridge USA, 1972.
- [182] W.Wang. On the Automatic Reconstruction of a 3D Object's Constructive Solid Geometry Representation from its 2D Projection Line Drawing. DSc Dissertation, University of Massachusetts at Lowell, 1992.
- [183] W.Wang and G.G.Grinstein. A Polyhedral Object's CSG-rep Reconstruction from a Single 2D Line Drawing. In *Proceedings of 1989 SPIE Intelligent Robots and Computer Vision III: Algorithms and Techniques*, Vol. 1192, 230–238, 1989.
- [184] W.Wang and G.G.Grinstein. A Survey of 3D Solid Reconstruction from 2D Projection Line Drawings. *Computer Graphics Forum* 12(2), 137–158, 1993.
- [185] C.E. Weatherburn. Elementary Vector Analysis, Bell, 1921.
- [186] C.E.Weatherburn. *Differential Geometry*, Vol. I, Cambridge University Press, 1927.
- [187] C.E.Weatherburn. *Differential Geometry*, Vol. II, Cambridge University Press, 1927.
- [188] X.Wei. Computer Vision Method for 3D Quantitative Reconstruction from a Single Line Drawing. PhD Thesis, Department of Mathematics, Bejing University, China, 1987. (In Chinese).
- [189] L.Weinberg. A Simple and Efficient Algorithm for Determining Isomorphism of Planar Triply Connected Graphs. *IEEE Transactions on Circuit Theory* CT-13, 142–148, 1966.

- [190] N.Werghi, R.B.Fisher, A.Ashbrook and C.Robertson. Improving Model Shape Acquisition by Incorporating Geometric Constraints. In *Proc. British Machine* Vision Conference BMVC97, 520–529, 1997.
- [191] N.Werghi, R.B.Fisher, C.Robertson and A.Ashbrook. Object Reconstruction by Incorporating Geometric Constraints in Reverse Engineering. Computer-Aided Design 31, 363–399, 1999.
- [192] W.Whiteley. Matroid and Rigid Structures. In ed. N.White, *Matroid Applications*, 1–53, Cambridge University Press, 1991.
- [193] W.T.Wu. Basic Principles of Mechanical Theorem Proving in Geometries, Vol. I, Springer-Verlag, 1993.
- [194] H.W.Yankee. Engineering Graphics, Prindle, Weber and Schmidt, 1985.
- [195] S.W.Zucker, E.Krishnamurthy and R.Harr. Relaxation Processes for Scene Labelling: Convergence, Speed and Stability. *IEEE Transactions on System*, Man and Cybernetics SMC-8(1), 41–48, 1978.