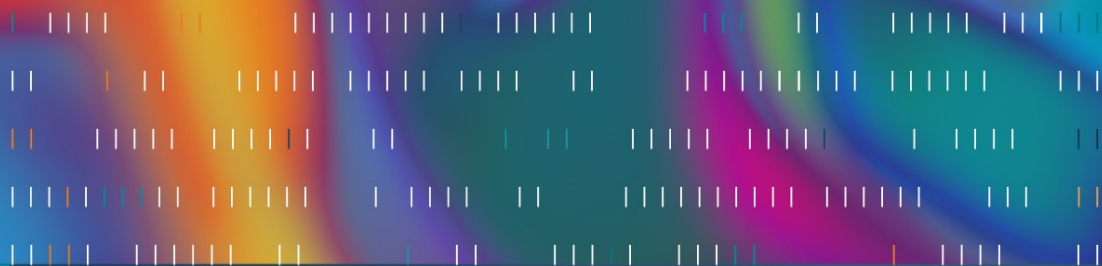


Martin Cooper



Line Drawing Interpretation

 Springer

Line Drawing Interpretation

Martin Cooper

Line Drawing Interpretation

 Springer

Martin Cooper, MA, PhD
University of Toulouse III
France
cooper@irit.fr

ISBN 978-1-84800-228-9 e-ISBN 978-1-84800-229-6
DOI 10.1007/978-1-84800-229-6

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2008928522

© Springer-Verlag London Limited 2008

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licenses issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.

The use of registered names, trademarks, etc., in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant laws and regulations and therefore free for general use.

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Printed on acid-free paper

Springer Science+Business Media
springer.com

Preface

The computer interpretation of line drawings is a classic problem in artificial intelligence (AI) which has inspired the development of some fundamental AI tools, including constraint propagation, probabilistic relaxation, the characterization of tractable constraint classes and, most recently, the propagation of soft constraints in finite-domain optimization problems. Line drawing interpretation has many distinct applications on the borderline of computer vision and computer graphics, including sketch interpretation, the input of 3D object models and the creation of $2\frac{1}{2}$ D illustrations in electronic documents.

I hope I have made this fascinating topic accessible not only to computer scientists but also to mathematicians, psychologists and cognitive scientists and, indeed, to anyone who is intrigued by optical illusions and impossible or ambiguous figures.

This book could not have been written without the support of the CNRS, the French *Centre National de Recherche Scientifique*, who financed my one-year break from teaching at the University of Toulouse III. The UK Engineering and Physical Sciences Research Council also financed several extended visits to the Oxford University Computing Laboratory. Section 9.1 is just a brief summary of the results on tractable constraints that have come out of this very productive joint research programme with David Cohen, Peter Jeavons and Andrei Krokhin. The various soft arc consistency techniques described in Chapter 8 were developed in collaboration with Thomas Schiex and Simon de Givry at INRA, Toulouse. I am also grateful to Ralph Martin and Peter Varley for their comments on the line-labelling constraints presented in Chapter 3.

Last but not least, no words can express my gratitude to Catherine and our two daughters, Ashley and Diana, for everything else.

Contents

Preface	v
1 Introduction	1
2 Impossible Pictures	5
2.1 Losing the Third Dimension	7
2.2 Two Planar Surfaces	8
2.3 Depth Order of Surfaces at a Point	10
2.4 Cubic Corners and Improbable Objects	12
2.5 Impossible Intersections	14
2.6 Impossible Wireframe Projections	17
3 Labeling Line Drawings of Polyhedra	21
3.1 Historical Background	21
3.2 Line Drawing Labelling as Optimization	25
3.3 Parallel-Lines Constraint	26
3.4 A Universal Constraint for Simple Junctions	32
3.5 Lines Sharing the Same Two Regions	38
3.6 Cyclic-Path Constraint	41
3.7 Parallel Junctions on Distinct Faces	44
3.8 Encoding of Soft Constraints	50
3.9 Non-Manifold Scenes	52
3.10 Discussion	52
4 Discrete Inflation Using Cubic Corners	55
4.1 Computer-Enhanced Perception	55
4.2 Machine Interpretation of Line Drawings	56
4.3 Depth Labels	57
4.4 Depth Labels and Impossible Figures	61
4.5 Propagation of Depth Labels	65
4.6 Orthogonality Constraints on Cubic Corners	67
4.7 Experimental Trials	70
4.8 Adding Depth Information to Line Drawings	72
4.9 Vertices Which Are Not Cubic Corners	73

4.10	Discussion	75
4.11	Conclusion	76
5	A Rich Labeling Scheme for Curved Objects	79
5.1	Labeling Line Drawings of Curved Objects	79
5.2	Regularities in Man-Made Objects	82
5.3	Planarity Constraints	84
5.4	Constraints from Orthogonal Edges	87
5.5	Examples of Drawing Interpretation	91
5.6	Complete 3D Reconstruction	94
5.7	Discussion	95
6	Depth Recovery Through Linear Algebra	97
6.1	Gradient Space and Gradient Directions	97
6.2	Linear Constraints and Curved Objects	100
6.3	Formulation of Linear Constraints	102
6.4	Deriving Linear Constraints from a Drawing	105
6.4.1	Vanishing Point Constraint	105
6.4.2	Constraints from Collinearity or Intersections	105
6.4.3	T-junction Constraint	106
6.4.4	Convex/Concave Edge Constraints	107
6.4.5	Coplanarity Constraints	112
6.4.6	Hidden-Surface Coplanarity Constraints	114
6.5	Orthographic Projection	117
6.6	Physical Realizability of Drawings	118
6.7	The Computational Problem	119
6.8	Conclusion	122
7	Wireframe Projections	125
7.1	Introduction	125
7.2	Semantic and Numerical Line Labels	127
7.3	Realizability	130
7.4	All Wireframes Are Ambiguous	141
7.5	Identifying Faces	142
7.6	Common-Surface Constraints	146
7.7	Coplanarity Constraints	147
7.8	Unambiguous Wireframes	150
7.9	Residual Ambiguity	153
7.10	Constraints Between Distant Lines	155
7.11	Tetrahedral Vertices	163
7.12	Tangential Edges and Surfaces	169
7.13	Rich Labelling Scheme	173
7.14	Discussion	178
7.15	Conclusion	180

- 8 Simplification of Combinatorial Problems** **183**
- 8.1 Transformations of Combinatorial Problems 183
- 8.2 When Local Reductions Suffice 185
- 8.3 Arc Consistency 187
- 8.4 Neighbourhood Substitution 189
- 8.5 Simplification of Soft Constraint Problems 191
- 8.6 Valuation Structures 192
- 8.7 Valued Constraint Satisfaction 194
- 8.8 Soft Arc Consistency Techniques 195
- 8.9 Optimal Soft Arc Consistency 198
- 8.10 Virtual Arc Consistency 203
- 8.11 VAC Decomposition 209
- 8.12 Soft Neighbourhood Substitution 212
- 8.13 Discussion 214

- 9 Tractability of Drawing Interpretation** **217**
- 9.1 Tractable Constraint Classes 217
 - 9.1.1 Zero/One/All Constraints 217
 - 9.1.2 Max-Closed Constraints 219
 - 9.1.3 Characterization of Tractable Boolean Constraints 220
 - 9.1.4 Characterization of Tractable Boolean Valued Constraints 221
- 9.2 Complexity of Line Drawing Interpretation 224

- 10 3D Reconstruction of Ambiguous Pictures** **231**
- 10.1 Reconstruction of Frontal Geometry 231
- 10.2 Hidden-Part Reconstruction 233

- Bibliography** **237**

- Index** **251**

Chapter 1

Introduction

Human vision appears both effortless and universal. No conscious thought is required to interpret a two-dimensional (2D) image as a three-dimensional (3D) scene. Although there is undoubtedly a learning period during early childhood, we have no recollection of it, which leaves us with the impression that our ability to see is innate and hence inherently mysterious. This impression is comforted by the fact that everyone seems to see the same objects when presented with emotionally neutral images, such as line drawings of polyhedral objects. Can we imagine seeing anything else other than a cube in Figure 1.1(a) or a house in Figure 1.1(b)? Is drawing, then, a universal natural language accessible to everyone without the need to learn tedious lists of vocabulary and rules of grammar? Artists, illustrators, draughtsmen, cartographers and photographers spend their professional lives producing images. Diagrams are an essential part of many technical reports and presentations. Images of all kinds are important pedagogical tools at all levels of teaching.

Of course, it is clear that pictures alone cannot represent the same range of abstract ideas that can be represented in words. For example, using the picture in Figure 1.1(b) to represent *all* houses (from a fisherman's cottage to a stately home) would be the definition of a pictorial symbol which, in the end, would be no less arbitrary than the word 'house'.

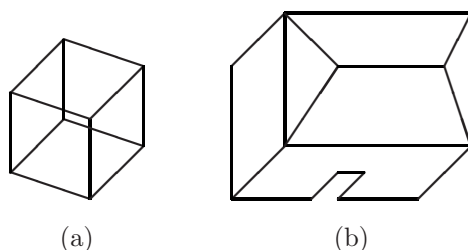


Figure 1.1: Simple drawings of polyhedral objects.

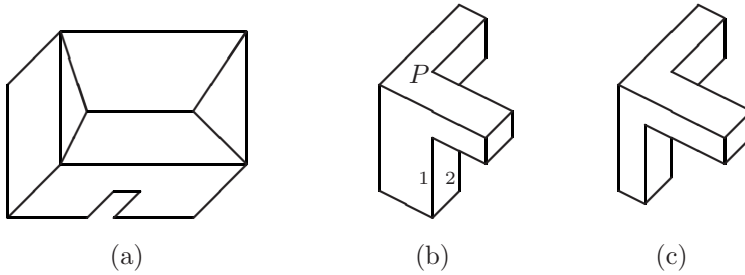


Figure 1.2: (a) A distorted house; (b) an incorrect drawing of an object; (c) the correct drawing of the same object.

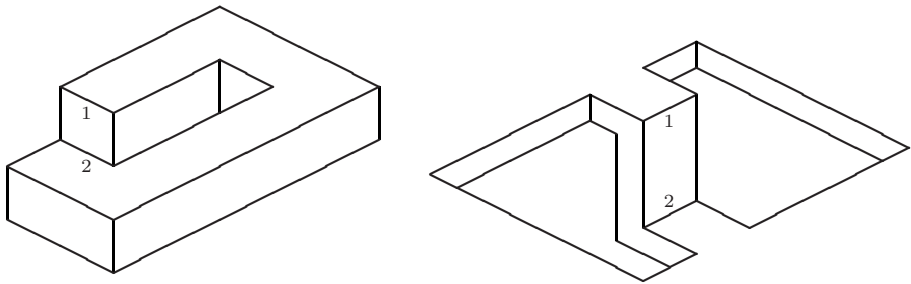


Figure 1.3: Two impossible pictures illustrating the same geometrical constraint.

As with written or spoken language, picture understanding is much easier than picture production. Consider the incorrectly drawn house in Figure 1.2(a), which now appears distorted. Figure 1.2(b) demonstrates a typical but subtle geometric error that is easily made: point P has mistakenly been drawn so as to be collinear with line 1, when it should, in fact, be collinear with line 2, as in the correct projection in Figure 1.2(c). These examples demonstrate that line drawings do have their own rules of syntax and semantics. This is highlighted by the study of impossible pictures and their reasons for impossibility. For example, the two pictures in Figure 1.3 are both impossible because they break the same geometrical constraint that two planes cannot intersect along two non-collinear lines (marked 1 and 2 in the drawings).

The computer analysis of line drawings is a classic problem in Artificial Intelligence (AI). The first major breakthrough was to identify the important subproblem consisting of labelling the lines in a drawing of a polyhedron as convex, concave or occluding [20, 75]. This subproblem is a classic example of how a global solution can be specified by a simple set of local constraints: in this case, the lists of the possible labellings of each type of junction. The line-labelling problem proved to be an inspiration for such widely applicable techniques as arc consistency [173, 108] and probabilistic relaxation [136]. The study of tractable constraint satisfaction problems is also intimately linked with the line-drawing-labelling problem [90, 127, 36].

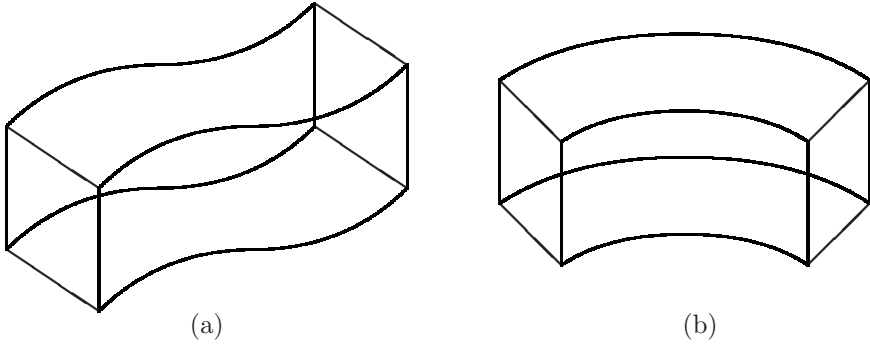


Figure 1.4: The wireframe projection (a) is ambiguous (since it could be either the top or the side of the object, which is planar), whereas (b) does not suffer from the same type of ambiguity.

A landmark result was Sugihara’s necessary and sufficient conditions for a line drawing to be realizable as the projection of a polyhedral scene [154]. This meant that line drawing interpretation was seen as a mixture of constraint satisfaction (over finite domains) and linear algebra (over real domains).

In certain computer vision applications, where high-quality images of uncluttered scenes are available, a raw image may be converted into a line drawing by edge detection and linking. Alternatively, the drawings to be analysed may have been produced by hand: obtained via a graphic pen or a mouse, for example, or by digitizing a traditional pen-and-paper drawing. However, line drawings derived from raw images and hand-produced drawings have different characteristics, as do technical drawings and sketches. For example, an occlusion edge cannot be detected in an image if there is insufficient contrast between the occluding and occluded surfaces. Hand-drawn sketches, on the other hand, are often far from being true projections of physically possible 3D scenes, as the examples in Figure 1.2 amply illustrate.

To cope with imperfections in the input and to overcome the inevitable ambiguity of a 2D projection of a 3D scene, we have to express line drawing interpretation as an optimization problem. This has led many workers to apply standard but incomplete optimization methods (such as hill climbing, simulated annealing and genetic algorithms) to the real-valued variables of the problem. A complementary approach is to perform a complete search over finite-domain variables in an appropriately chosen subproblem. This approach can profit from recent progress on optimization problems over finite domains, including various soft consistency techniques and the characterization of tractable classes of soft constraints [115]. For example, we show in Chapter 7 how soft constraint satisfaction allows us to automatically recognize that the wireframe projection in Figure 1.4(a) is ambiguous but that the similar drawing in Figure 1.4(b) is not.

Computer vision applications in which some form of line drawing interpretation

has been employed include scene analysis, robot navigation and object recognition. Matching high-level primitives such as vertices, faces or parts of objects, rather than pixels, increases the accuracy of parameter estimation in such applications as stereo matching or motion analysis. The detection of these primitives may use some local form of line drawing interpretation. The correspondence problem between two line drawings of curved objects is discussed in detail in [31].

The interpretation of hand-made line drawings (whether produced by scanning a pen-and-paper drawing or entered via a mouse or data tablet) has applications in the creation of 3D object models such as boundary representation or constructive solid geometry models. Company et al. [27] provide a recent survey of work on the 3D reconstruction of line drawings over the last 30 years. They point out the gradual shift of research effort from the analysis of engineering drawings [2] to the interpretation of freehand sketches. A typical application is the addition of new elements to an existing view of the interior or the exterior of a building to create a virtual image of the renovated building [162]. An intriguing new application area is in 3D object retrieval from large databases or the web [124].

The obvious input data for complete 3D object reconstruction is a wireframe projection (a line drawing in which all edges are represented, including those facing away from the viewer or occluded by a nearer surface). The labelling of wireframe projections was first studied by Huffman [75]. In Chapter 7 we give necessary and sufficient conditions for a line drawing to be a legal wireframe projection of a curved object. If the input is a standard line drawing of an opaque object in which only visible edges are shown, then complete object reconstruction is highly under-constrained. Systems for hidden face recovery [57, 18] use both strict geometrical constraints and Gestalt principles of human perception such as symmetry, simplicity and closure in order to choose the most likely 3D object.

Another range of applications is the computer enhancement of drawings. For example, estimating the gradients of surfaces allows the computer to automatically add shading information to a drawing. Furthermore, electronic publishing opens up the possibility of creating documents containing virtual 3D figures. Even simple histograms, tree diagrams or maps can potentially convey much more information if given a third dimension which the reader is allowed to explore by changing the viewpoint.

Chapter 2

Impossible Pictures

Impossible pictures have proved to be an important source of geometric constraints which can be applied by a computer program when interpreting line drawings. Although this is our main motivation, there are several other reasons why impossible pictures have been studied. M.C. Escher raised the study of impossible pictures to an art form by incorporating them into his etchings. Psychologists and cognitive scientists have used pictures of impossible objects to investigate human visual processing and memory. Figure 2.1 shows the kind of impossible pictures used in psychological experiments to determine how shapes are represented in the brain [113, 172]. Certain odd-looking pictures appear to represent impossible objects but can, in fact, be realized as the projection of a polyhedron. Sugihara [156] suggested using the unfolded surface of such a polyhedron as a toy so that children could play at constructing the ‘impossible’ object.

Sugihara distinguished between different classes of impossible drawings [153]. An important subclass of impossible drawings (called correctable) are those that can be rendered realizable by adjusting the positions of junctions without

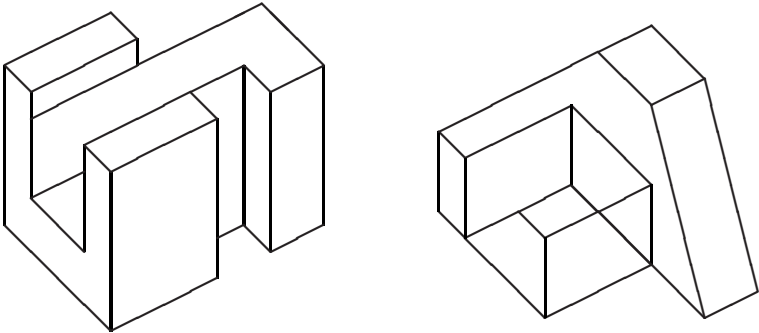


Figure 2.1: Pictures of impossible objects used in psychological experiments.

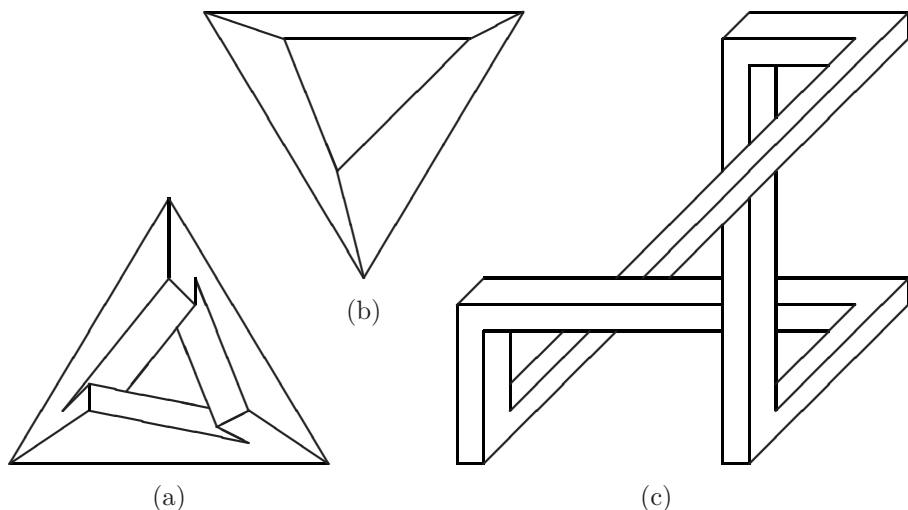


Figure 2.2: (a) A physically realizable drawing; (b) an impossible but correctable drawing; (c) an impossible and uncorrectable drawing (adapted from [144]).

altering the basic configuration of the drawing (i.e. the planar graph representing the adjacency of junctions and lines together with the set of pairs of lines meeting at an angle less than π). Figure 2.2 shows examples of (a) a possible drawing, (b) an impossible but correctable drawing, (c) an impossible and uncorrectable drawing. The three edges of the pyramid in Figure 2.2(b) should meet at a point when extended: this can easily be fixed by changing the position of a single Y junction. In this section we provide an informal but finer classification of impossible drawings by enumerating some basic reasons for their impossibility. This discussion was inspired by classic examples of impossible figures [61, 134, 144]. The most aesthetically pleasing impossible pictures are those that have been specifically designed to be locally coherent but for which no coherent global interpretation exists. We begin by studying perhaps the most famous example of an impossible figure, the impossible tribar, discovered by Oscar Reutersvärd in 1934 and independently by Roger Penrose [128] in 1958.

In all the examples given in this section we assume an orthographic projection of polyhedral opaque objects from a general viewpoint. Under orthographic projection, parallel 3D edges project into parallel lines in a drawing. Our assumption of a general viewpoint means that collinear lines in a drawing are projections of collinear lines in 3D, that parallel lines in a drawing are projections of parallel lines in 3D, that straight lines in a drawing are projections of straight lines in 3D, etc.

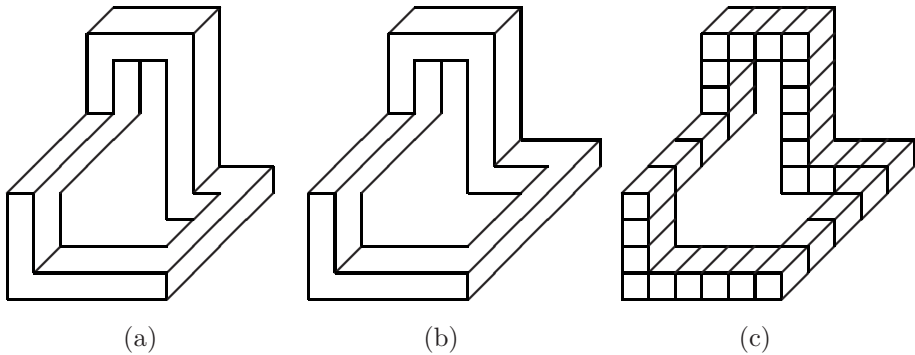


Figure 2.3: (a) A possible figure; (b),(c) an impossible figure.

2.1 Losing the Third Dimension

Consider a simple class of objects consisting of polyhedra with faces parallel to one of three orthogonal directions, with a square cross-section and topologically equivalent to a torus. Figure 2.3(a) is a physically realizable drawing of one such object. However, the drawing in Figure 2.3(b) is not physically realizable, as can be verified by counting bricks in Figure 2.3(c). Let the X , Y and Z axes represent respectively the horizontal, vertical and depth axes in 3D. Consider a 3D object composed of m segments parallel to the X axis (of lengths x_1, \dots, x_m), n segments parallel to the Y axis (of lengths y_1, \dots, y_n) and p segments parallel to the Z axis (of lengths z_1, \dots, z_p). Since the object is equivalent to a closed curve, we can assign an arbitrary order to these segments corresponding to a circuit C . Segment lengths x_i, y_j, z_k are considered to be negative if the value of x, y, z decreases as we follow the circuit C . A necessary condition for the object to be realizable is that the net displacement along the circuit C must be zero, i.e.

$$\sum_{i=1}^m x_i = \sum_{j=1}^n y_j = \sum_{k=1}^p z_k = 0. \quad (2.1)$$

We can easily verify that Equation (2.1) is not satisfied by the object depicted in Figure 2.3(b),(c), which proves that it is not physically realizable.

For the lines to join up in 2D we only require that (under an appropriate choice of units in the x, y and z directions)

$$\sum_{i=1}^m x_i + \sum_{k=1}^p z_k = \sum_{j=1}^n y_j + \sum_{k=1}^p z_k = 0. \quad (2.2)$$

Figure 2.4 shows five examples of impossible figures, including the famous Penrose triangle and the most elementary impossible staircase [128]. In each case, the figure is impossible since Equation (2.2) is satisfied but Equation (2.1) is not. We call this the impossible closed curve class of impossible drawings.

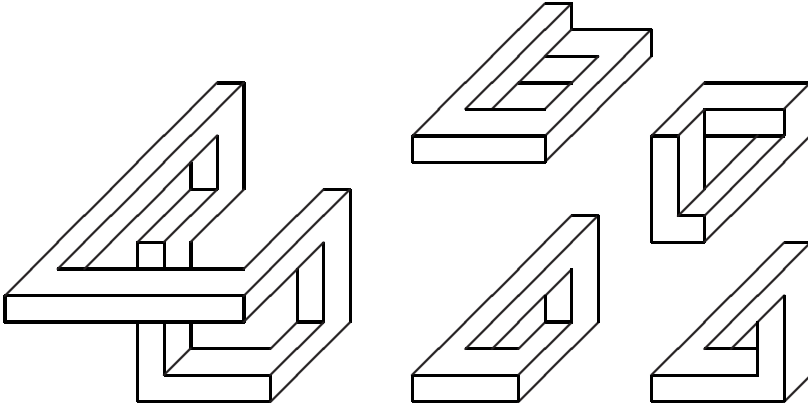


Figure 2.4: Some examples of impossible figures.

2.2 Two Planar Surfaces

In the classic machine-vision approach of Huffman [75] and Clowes [20], the aim is to find an incomplete but nevertheless very informative representation of a 3D scene by labelling each line segment in the drawing as the projection of an occluding, a convex or a concave edge. These are known as semantic labels. The only constraints are provided by a catalogue of junction labellings, derived by imagining all possible projections of physically possible vertices. However, there is also an unstated but inherent geometrical constraint which is used in this approach, namely that the semantic label of a straight line which is the projection of the intersection of two planar faces is invariant between junctions.

The absence of a semantic labelling proves that a drawing is physically impossible. The most famous example is the impossible fork: Figures 2.5(a) and (b) show two variations. A more subtle example is given in Figure 2.6 (based on an example by Penrose [129]). Line i in Figure 2.6 is concave if and only if line $(i + 1) \bmod 5$ is convex (for $i = 0, \dots, 4$). There can clearly be no solution when the number of such lines (in this case 5) is an odd number.

Unfortunately, many other examples of impossible figures cannot be detected by semantic labelling alone, since they have a global legal labelling, notably drawings from the impossible closed curve class (Figure 2.4)

Figure 2.7(a) shows a classic example of an impossible figure which is the basis for M.C. Escher's famous etching 'Belvedere'. To see that this is an impossible projection of a polyhedron, note that no perfectly planar face F can occur simultaneously on both sides of a 3D edge since there can be no gradient or depth discontinuity within F , and yet this is what occurs along the occluding lines 1 and 2 in Figure 2.7(a). We can, in fact, find other reasons to declare this an impossible figure. Consider the reduced version in Figure 2.7(b). In this reduced version, surfaces S and S' appear to be coplanar, but T intersects S and S' along two non-collinear lines, which is impossible if all faces are planar. It is interesting to note that, despite the fact that both of the objects depicted

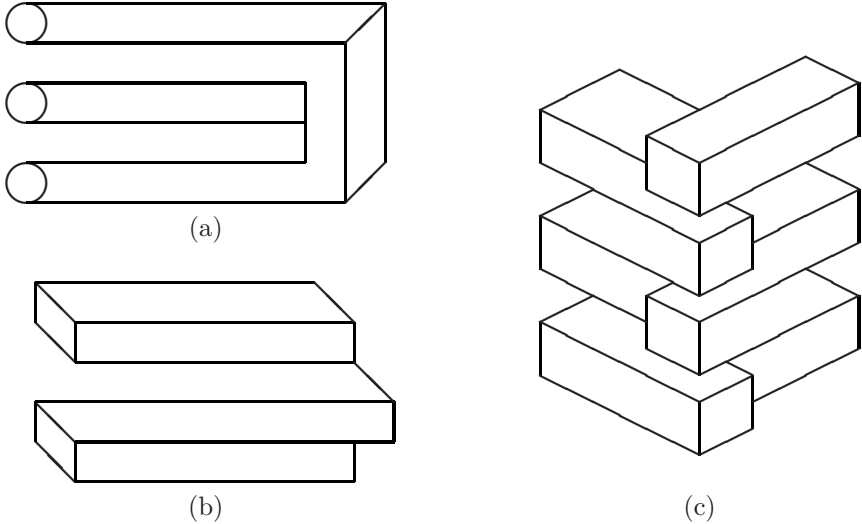


Figure 2.5: (a),(b) Impossible forks; (c) another impossible drawing in which lines appear to have different semantic labels at their two ends.

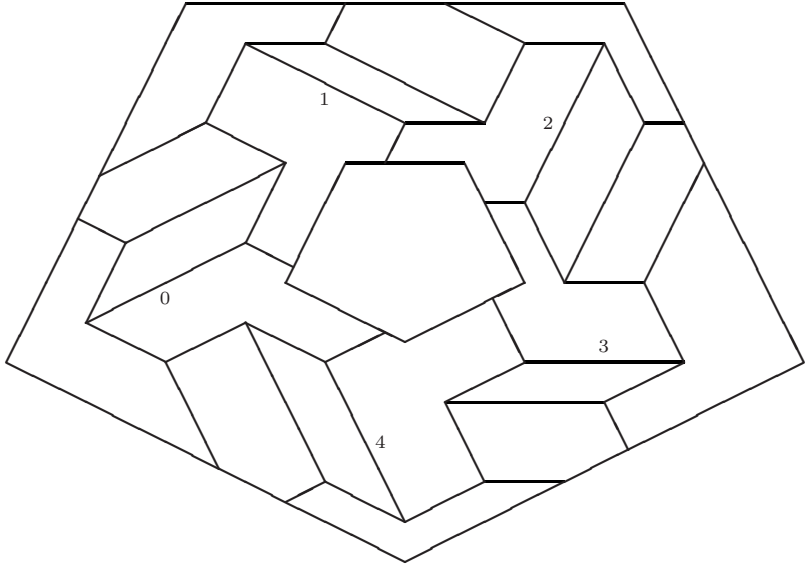


Figure 2.6: A drawing which has no legal labelling in terms of concave and convex edges.

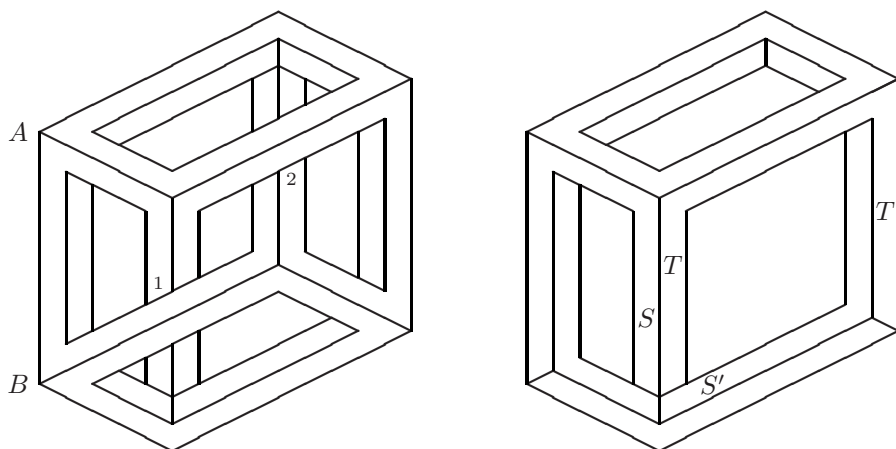


Figure 2.7: (a) An impossible figure resuming the basic structure of M.C. Escher's 1958 etching 'Belvedere'. (b) A reduced version.

in Figure 2.7 are physically impossible, most people would agree that one is a subset of the other.

Another recipe for producing impossible figures is to deliberately produce depth incoherencies. Consider firstly the simple case of two planar faces A , B (not parallel to the line of sight) under orthographic projection. Let $d_A(x, y)$, $d_B(x, y)$ denote the depths of faces A , B at the scene point projecting into image point (x, y) . Then d_A, d_B are linear functions of x and y . Hence $d_B - d_A$ is also a linear function of x and y and can itself be considered as a plane. Such a function is not planar if it contains a ridge, a peak or a saddle point. Figure 2.8 shows examples of impossible figures in which (save for accidental alignment) $d(B) - d(A)$ contains a peak or a saddle point. (The case of a ridge is even easier to produce and can be considered as the limiting case of either of these two cases in which two points marked \circ coincide.) The two drawings in Figure 2.8 also contain many other visual cues, such as lines on face A which are parallel to lines on face B . Figure 2.9 shows an example of an impossible figure which is only impossible by the "no peak in $d(B) - d(A)$ " rule.

2.3 Depth Order of Surfaces at a Point

Consider a vertical line L in a drawing, separating regions S_L and S_R lying (respectively) to the left and right of L . Suppose that P_L , P_R are the planes of the faces visible in regions S_L , S_R (respectively). If L is the projection of a convex edge, then P_L lies in front of (behind) P_R at all points to the left (right) of L . Returning again to the Penrose triangle, illustrated in Figure 2.10(a), we can also deduce its impossibility by reasoning about the relative depth of planes. The Penrose triangle appears to have three planar faces S , T , U . Let P_S , P_T ,

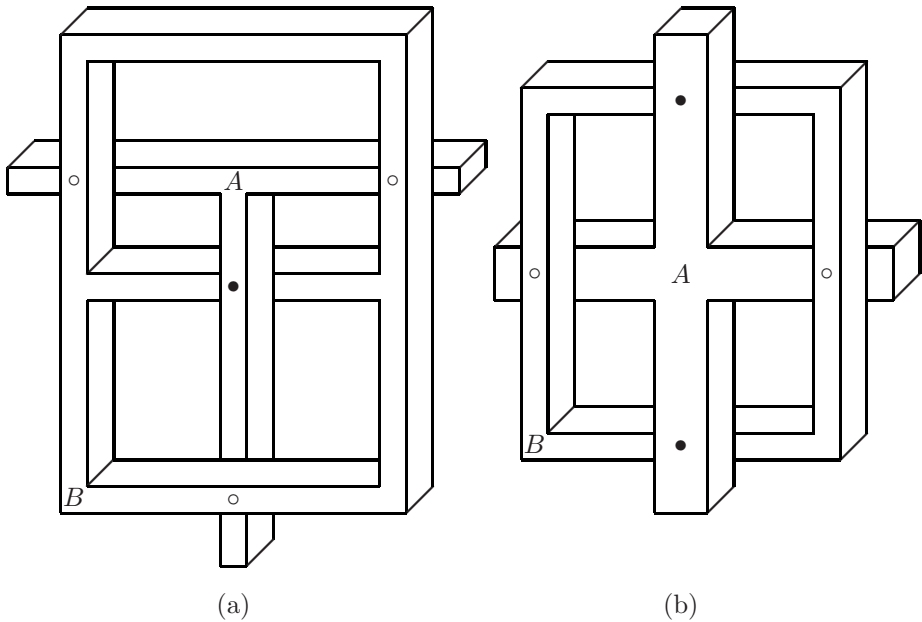


Figure 2.8: (a) A figure which is impossible since there would be a peak in $d(B) - d(A)$; (b) a figure which is impossible since there would be a saddle point in $d(B) - d(A)$. In both cases \bullet represents an image point where A occludes B and \circ represents a point where B occludes A .

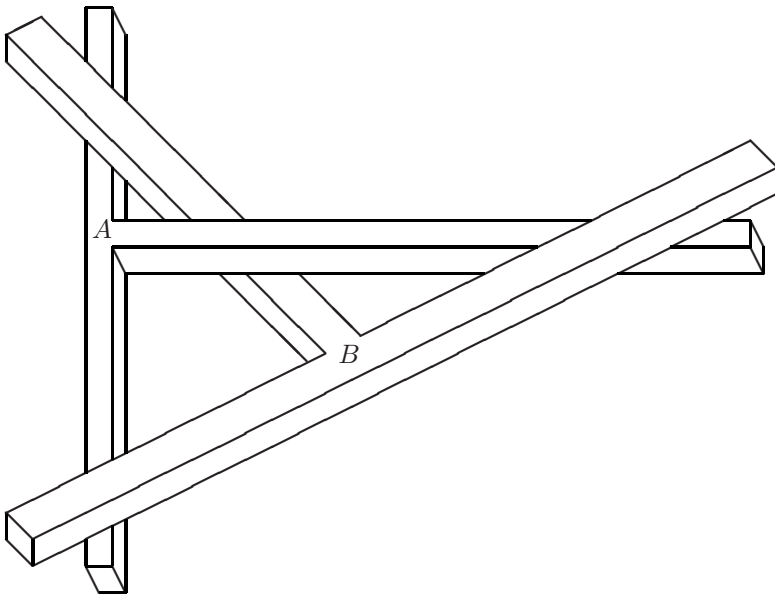


Figure 2.9: A figure which is impossible due to a peak in $d(B) - d(A)$.

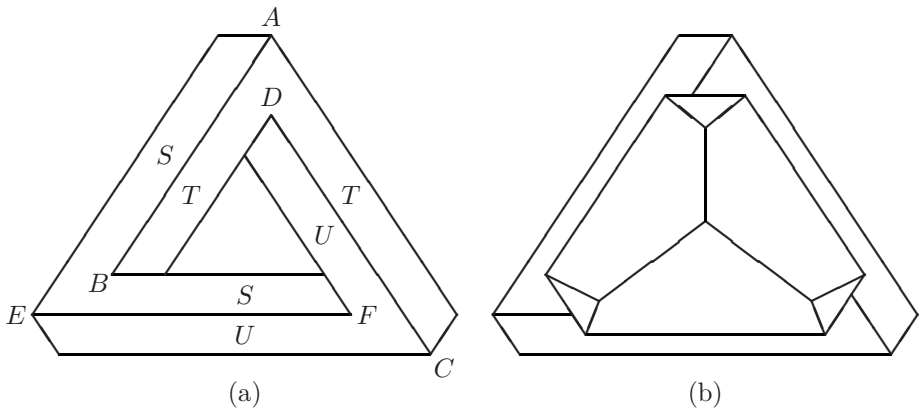


Figure 2.10: (a) The Penrose triangle; (b) a partially occluded Penrose triangle.

P_U be the planes of these three faces. Furthermore, we see AB , CD and EF as convex edges. Consider now a point Q in the very centre of the inner triangle. By the reasoning above, we can deduce that at point Q , P_S is in front of P_T , P_T is in front of P_U , and P_U is in front of P_S . This is clearly impossible. Observe that this argument applies to various versions of the same drawing, such as the partially occluded Penrose triangle shown in Figure 2.10 or a Penrose triangle in which some lines are curved (see Figure 6.20 in Chapter 6).

We will show in the next chapter that we can impose constraints on the labelling of sets of lines by disallowing such impossible depth cycles. These extra constraints complement traditional constraints given by a catalogue of labelled junctions. For example, the labelling $(+, +, +)$ for lines AB , BC , CD in Figure 2.11 is illegal: since AB and DC are parallel, they meet at a point Q at infinity (as we extend both AB and DC upwards), surface S necessarily passes through Q , and the convex label for BC implies that surface T passes behind Q , but this then contradicts the fact that DC passes through Q . In general, a labelling for a set of lines is illegal if it implies that at some point Q the depths d_1, \dots, d_r of surfaces S_1, \dots, S_r satisfy $d_1 \leq d_2 \leq \dots \leq d_r < d_1$ [77, 43].

2.4 Cubic Corners and Improbable Objects

Faced with an ambiguous drawing, we tend to see familiar features, such as cubic corners (vertices formed by the intersection of faces lying in three mutually orthogonal planes). An interesting example which illustrates the preference of human vision for cubic corners is shown in Figure 2.12(a). Our first impression is that this is a drawing of three cubes. However, we can see that this interpretation is physically impossible by looking at the central Y junctions of the three blocks. The stems of these three Y junctions are all parallel, as are the three

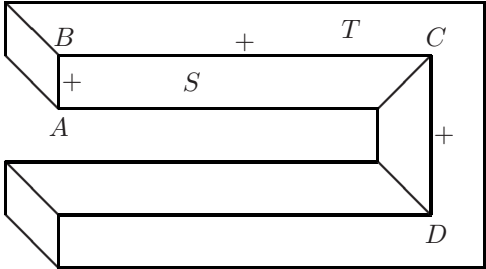


Figure 2.11: An object which is an impossible projection of a polyhedral object in which AB, CD are parallel.

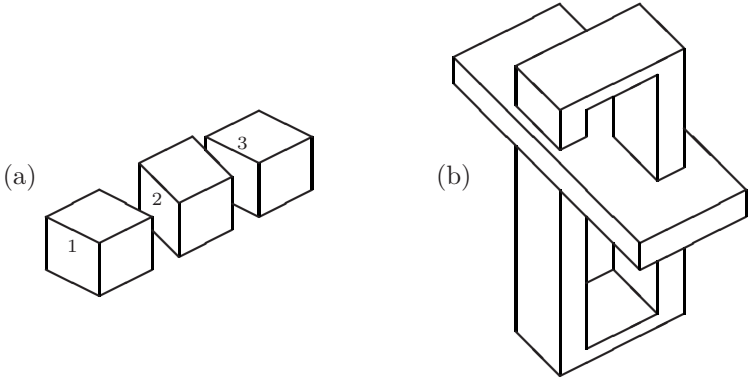


Figure 2.12: (a) A drawing which is not the projection of a collection of objects with cubic corners; (b) a drawing which is impossible if collinear lines in the drawing are projections of collinear 3D edges.

right branches, but the left branches (lines 1, 2, 3) are not parallel. We are forced to deduce that at least one of the objects is not a cube. It is interesting to note that even when a drawing is clearly impossible, as in Figure 2.12(b), we are still capable of producing a globally incoherent but locally coherent interpretation. For example, all the vertices in Figure 2.12(b) are interpreted as cubic corners, which allows us to assign a gradient to each visible surface.

The human tendency to see cubic corners whenever this is locally coherent provides another contradiction when trying to find a globally coherent interpretation of Figure 2.7(a). When we observe vertex A , we have the impression that it is a cubic corner with the directed edge AB sloping away from the viewer. When we observe vertex B , we have the impression that it is a cubic corner with the directed edge AB this time sloping towards the viewer. In fact, we see contradictory slopes on all vertical lines in the drawing. Despite these contradictions, we nevertheless interpret all vertices of the object as cubic corners.

As observed by Burns [17], cubic corners abound in man-made objects due to gravity and common manufacturing methods such as carpentry. We are so used to seeing objects with cubic corners that a physically realizable drawing which cannot be the projection of an object with cubic corners sometimes appears highly improbable. Figure 2.13 gives a very striking example; the drawing in Figure 2.13(a) appears possible whereas the drawing in Figure 2.13(b) appears impossible. Sugihara [156] has pointed out that drawings such as Figure 2.13(b) are in fact realizable as a projection of objects in which some of the vertices are not cubic corners, but it is practically impossible for a human being to see the correct interpretation. Our preference for cubic corners is so strong that in this case human vision fails miserably compared to machine vision, which can correctly reconstruct a 3D scene projecting into this drawing.

2.5 Impossible Intersections

Another important condition that must be satisfied by a valid 3D interpretation is that no two objects (or parts of the same object) can occupy the same point in 3D space. The drawing in Figure 2.14(a) is unrealizable (assuming that faces A , B are parallel and that faces C , D are parallel), since the object should intersect itself but does not. Figure 2.14(b) shows a similar example: the presence of collinear and parallel lines implies that surfaces T and U are parallel, which means that the hole in surface S is too small.

An important challenge is to explain more subtle examples of drawings which are unrealizable as the projection of objects with cubic corners and to translate this explanation into a simple generic constraint. Figure 2.15(a) shows an example of an improbable drawing. To demonstrate that it is unrealizable as the projection of objects with cubic corners, we require some rather complex reasoning such as the following: AB and CD are parallel, hence surfaces S and T are parallel (assuming A and C are cubic corners); raising surface T so it coincides with surface S provides a contradiction since the width y of T is too large (as shown in Figure 2.15(b)). The challenge is to find a psychologically plausible

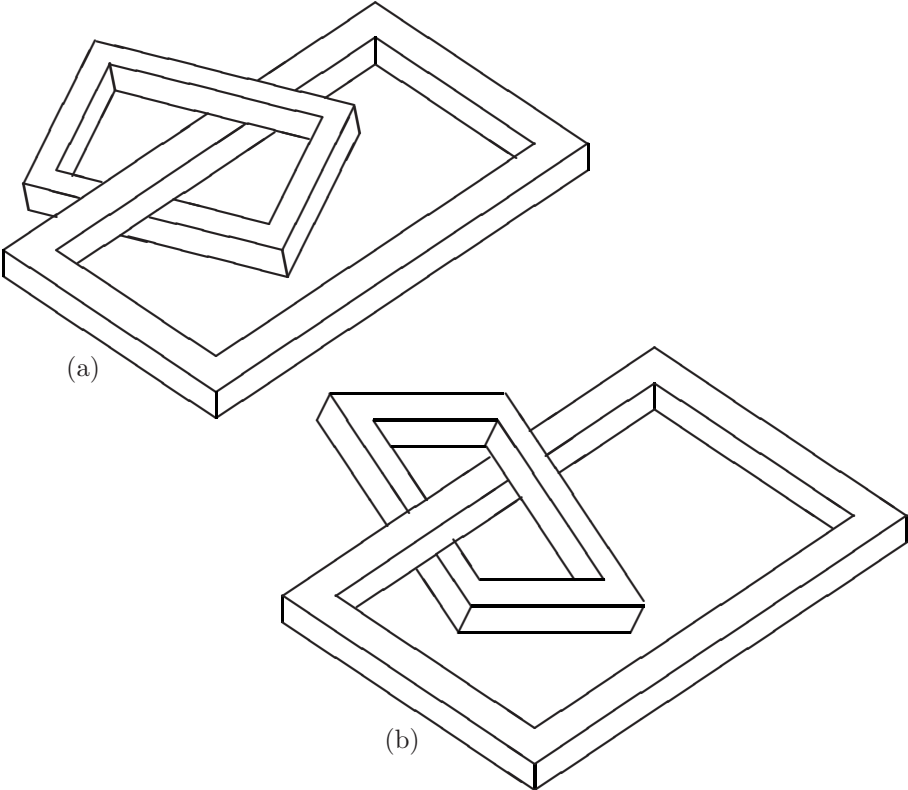


Figure 2.13: Figures which are seen as possible (a) and impossible (b) projections of objects with cubic corners.

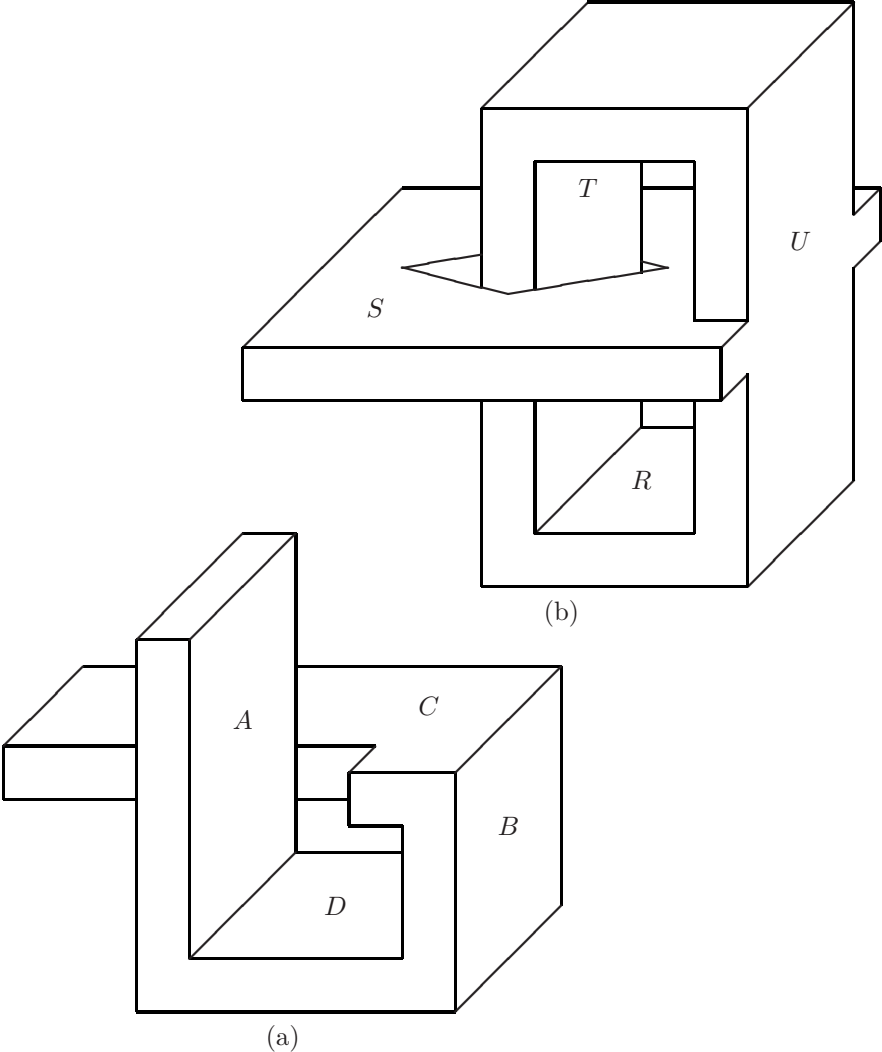


Figure 2.14: (a) A drawing which is unrealizable if faces A, B are parallel and faces C, D are parallel; (b) a drawing which is unrealizable if surfaces R, S are parallel and faces T, U are parallel.

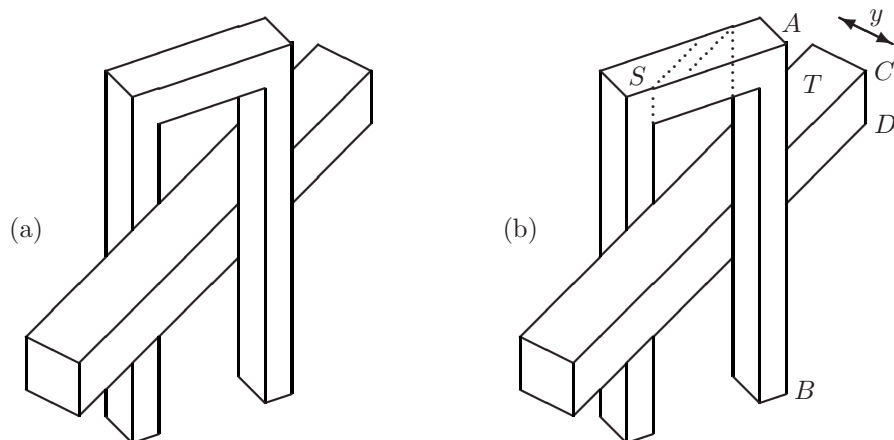


Figure 2.15: A drawing which is not realizable as a collection of objects with cubic corners since the distance x between the two parallel dotted lines on surface S is less than y .

explanation for the capacity of human vision to detect such contradictions.

2.6 Impossible Wireframe Projections

An artist's line drawing usually only depicts the visible edges of a 3D scene; those edge-segments which are occluded by faces nearer the viewer are invisible. On the other hand, a wireframe projection of an object is a projection of *all* the edges of the object, including occluded edges. Traditionally, engineering drawings are wireframe projections, in order to allow the draughtsman to depict both the back of the object and any internal structure (such as holes or concavities). Wireframe projections are the natural choice of input to a program whose aim is the 3D reconstruction of an object from one or more drawings.

The drawings in Figure 2.16(a)-(d) are impossible wireframe projections of solid objects. Consider, for example, Figure 2.16(b). When attempting to interpret this drawing as a solid object, we identify three trapezoidal faces $ABCD$, $CDEF$, $EFAB$. However, trying to complete a solid model by adding other faces leads inevitably to failure. For example, $ABCF$ cannot be a valid face since this would mean that three faces terminated along AB ; this is impossible in a manifold object. In Chapter 7 we extend the traditional line-labelling problem to wireframe projections by labelling each line not only with a semantic label (convex, concave, occluding) but also with two numerical labels indicating the number of surfaces in front of and behind the edge. This labelling scheme detects the physical impossibility of the wireframe projections in Figure 2.16(a)-(d).

Figure 2.16(e) has a legal labelling but is not the wireframe projection of a polyhedron. In fact, due to digitization errors, most drawings will not satisfy

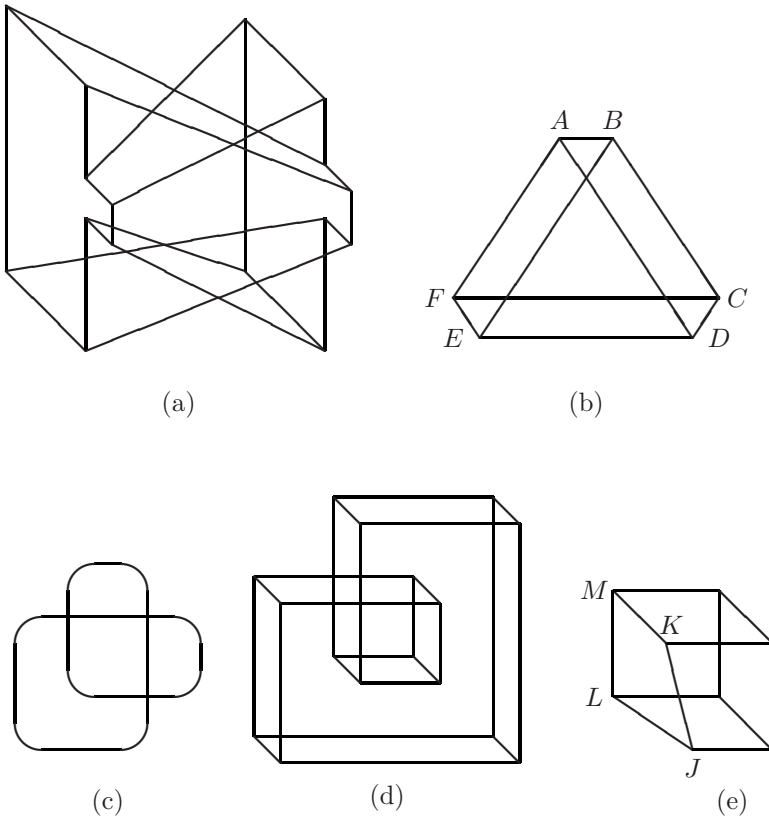


Figure 2.16: (a)-(d) Examples of impossible wireframe projections of solid objects; (e) an impossible wireframe projection of a polyhedron.

the necessary geometrical constraints to be a valid projection. This problem is known as *superstrictness* [155]. The correction of impossible drawings is a challenging problem in its own right. For example, to correct Figure 2.16(e) we have many choices, including: shifting junction J to the left (to produce a parallelepiped), shifting junction K or junction L to the right (to produce a wedge-shaped object), shifting junction M to the left (to produce an object with only one rectangular face) or simultaneously shifting all eight junctions (to produce an object with no rectangular faces).

Chapter 3

Labeling Line Drawings of Polyhedra

3.1 Historical Background

The interpretation of line drawings is a classic problem in machine vision. The pioneers in this field [20, 75] concentrated on perfect projections of opaque polyhedral objects and on the problem of labelling each line as concave, convex or occluding. Each line L in a drawing is assumed to be the projection of a 3D edge formed by the intersection of two planar surfaces S_1, S_2 . Line L is occluding if only one of S_1, S_2 is visible in the drawing. Otherwise, L is concave or convex depending on whether the exterior angle between S_1 and S_2 is less than or greater than π , respectively. With the further restriction to trihedral vertices (i.e. vertices formed by the intersection of three faces), there are only a limited number of legal labellings for each different junction type (L, Y, W, T). The complete list of legal junction labellings is often known as the Huffman–Clowes catalogue.

When three distinct planar surfaces meet at a point to form a 3D vertex, they divide space into eight octants. Each octant may be empty or filled with matter. Figure 3.1 shows the six possible trihedral vertices thus obtained, assuming that the object is a 3D-manifold in the neighbourhood of the vertex. (There is, in fact, a seventh vertex which is not shown since it is simply a reflected version of vertex C). The viewpoint may be situated in any of the empty octants. By exhaustion, it is easy to draw up a list of all labelled projections of the vertices in Figure 3.1 from all possible viewpoints. This catalogue is given in Figure 3.2. All rotations of these labellings are clearly also legal, which means, for example, that a Y junction has, in fact, five possible legal labellings. Many workers considered an even simpler set of possible vertices by (rather arbitrarily) disallowing vertices of type C and D in Figure 3.1 (known as extended trihedral vertices) in which an edge passes through the vertex.

Although determining whether a given line drawing has a global labelling

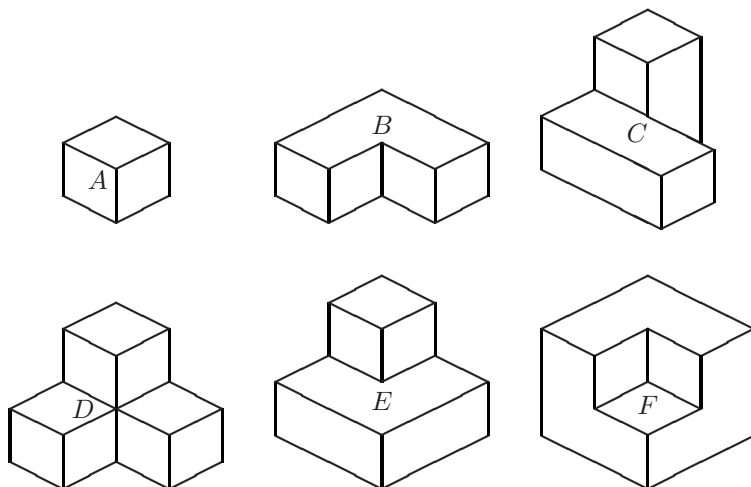


Figure 3.1: The six basic 3D-manifold vertices formed by the intersection of three distinct planar surfaces.

consistent with the Huffman–Clowes catalogue is known to be an NP-complete problem [92], the median-case complexity has been empirically observed to be $O(n)$ [126] and polynomial-time algorithms exist for certain special cases [90, 127]. Furthermore, the Huffman–Clowes catalogue is often sufficient to reduce the theoretically exponential number of labellings to a manageable number of legal labellings when it is used in conjunction with the *outer-boundary constraint*, which says that when a drawing represents an isolated object or group of objects, the outer boundary can be assigned a unique labelling corresponding to an occluding edge.

This initial success was tempered by the following points:

1. The catalogue of labelled junctions provides necessary but not sufficient conditions for the physical realizability of a drawing.
2. Classifying lines as projections of concave, convex or occluding edges only provides partial information about the corresponding 3D scene.
3. The restriction to perfect projection of polyhedral objects with trihedral vertices is too unrealistic for most vision applications.

Sugihara [154, 155] resolved point 1 above by giving necessary and sufficient conditions for the physical realizability of a legally labelled line drawing of a polyhedral scene by expressing the problem as a linear programming problem whose solution represents the equations of visible faces of the objects in the scene. This technique theoretically also resolves point 2, but there is often a great deal of ambiguity in the result which is not present in a human interpretation of the same drawing. For example, a drawing of a cube is immediately

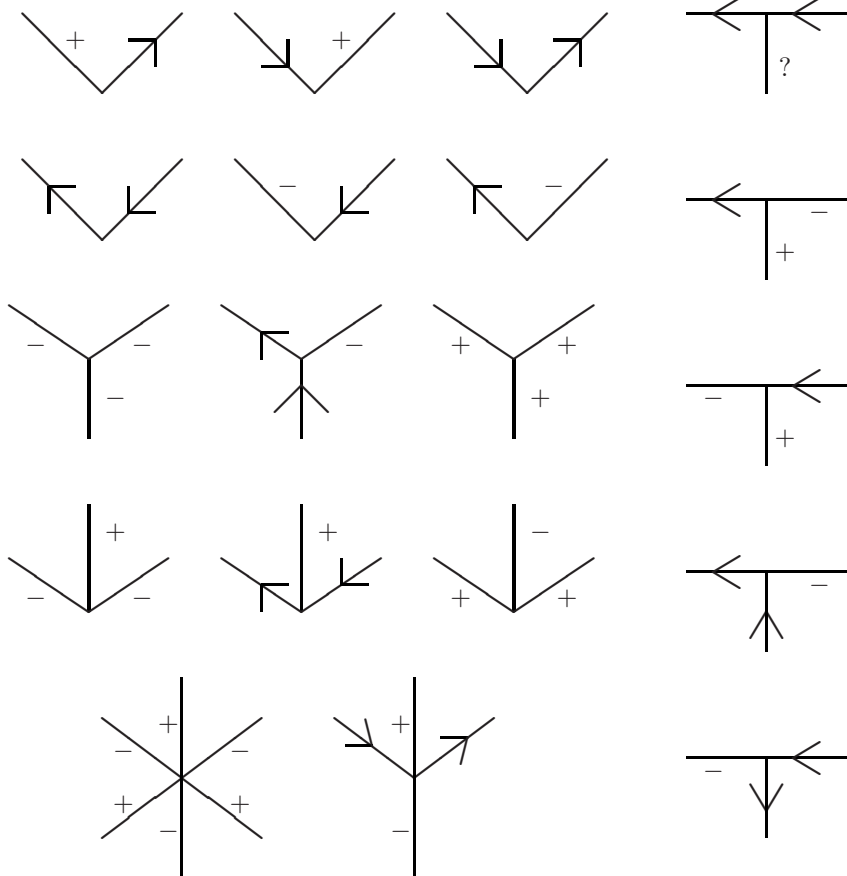


Figure 3.2: The catalogue of labelled junctions which are projections of trihedral vertices. A question mark represents any label.

interpreted by a human being as a cube, even though it could theoretically be the 2D projection of any of a large class of parallelepipeds.

Much progress has been made in recent years concerning point 3. Some systems have been specifically designed to allow for freehand sketching errors [70, 103, 164]. Catalogues of labelled junctions have also notably been given for curved objects [32, 109], for curved objects with edges and surfaces which can meet tangentially [34], for polyhedral objects with tetrahedral vertices [165] and for drawings of scenes with lighting effects such as shadows [173] and contrast failure [38]. When the restriction to polyhedral objects with trihedral vertices is relaxed, theoretical analysis [34, 38] has shown that a catalogue of labelled junctions is insufficient to disambiguate line drawings. Furthermore, experimental trials indicated that the time complexity to find the best labelling grows with the number of valid labellings, which is now an exponential function of the size of the drawing (P.A.C. Varley, pers. com.).

Although junctions can be said to provide most of the information in a technical drawing, other features provide valuable clues. For example, if we know the vanishing points of all lines in a drawing of a polyhedral object with trihedral vertices, then the labelling problem is no longer NP-complete, but can be solved in polynomial time [127]. Other sources of 3D information include the collinearity of a line with a junction or another line [37, 38], parallel lines [36] and straight lines (in the case of curved objects with some straight edges) [37].

When multiple interpretations are possible, many heuristics have been used to find the most plausible interpretation, such as maximizing commonly occurring 3D features such as right angles, vertical edges, symmetries and parallel planes, while minimizing the number of distinct objects, angles and edge lengths in the reconstructed 3D scene [103].

In this chapter we express the line drawing labelling problem as a valued constraint satisfaction problem [50]. This allows us to mix hard constraints (which must imperatively be satisfied) with soft constraints expressing preferences between different combinations of labels. This framework allows us not only to combine several junction catalogues based on different assumptions (such as trihedral/tetrahedral vertices, polyhedra/curved objects) but also to express preferences (for example for right angles or parallel faces).

The main contribution of this chapter is the introduction of novel constraints between unconnected lines or junctions, based on parallel lines, cycles of lines or collinearity. These include generic constraints between lines lying on a path in a drawing as well as preference constraints between the labellings of pairs of junctions lying on parallel lines. Such constraints are essential to avoid an exponential number of legal labellings of drawings of objects with non-trihedral vertices. These non-local constraints permit the propagation of information between unconnected components of a drawing. Among other things, these constraints formalize and generalize constraints between holes or bosses and the boundary of the planar surface on which they lie [167] or between lines separating the same two regions [90, 168]. These new constraints considerably strengthen the trihedral catalogue of Huffman [75] and Clowes [20], which is often insufficient to uniquely identify the correct labelling of a drawing. Our

constraints thus provide an alternative to Sugihara’s necessary and sufficient conditions for realizability [154]. Although incomplete, they have the advantage of being applicable on small subsets of a drawing (in the same way as junction constraints), whereas Sugihara’s test must be applied to each global labelling, of which there may be an exponential number.

The strength of these constraints is demonstrated by their ability to identify the unique correct labelling of many drawings of polyhedral objects with tetrahedral vertices. These new constraints also allowed us to deduce a general polyhedral junction constraint for the case where there is no limit on the number of faces which can meet at a junction.

3.2 Line Drawing Labelling as Optimization

To obtain labelling constraints, such as the Huffman–Clowes catalogue, we must make assumptions about the 3D scenes which may be represented and the projection operation which produces the drawing. Common assumptions include planar faces, trihedral vertices, general viewpoint, general object positions and perfect projection. We can measure the plausibility of a labelling by determining how many of these assumptions must be relaxed (and how many times). For example, consider a drawing and two labellings L_1 and L_2 , such that L_1 requires that two vertices be tetrahedral and L_2 requires that one pair of parallel lines in the drawing not be actually parallel in 3D. Should we prefer L_1 to L_2 or vice-versa, or should we accept both as plausible interpretations?

Different applications may give rise to different answers to this and similar questions. For example, if a line drawing has been derived from a human-entered drawing, then the user may have explicitly specified that certain lines are projections of parallel lines in 3D (but this is not assumed in this book), or knowledge of the application area may imply that tetrahedral vertices are quite common. In all cases, recent work on reduction operations on valued constraint satisfaction problems may be applied to mitigate the theoretical intractability of the resulting optimization problem (reduction operations for valued constraints are described in detail in Chapter 8). Valued constraints are local cost functions which allow us to express preferences between legal labellings and also to mark other labellings as illegal by assigning them an infinite cost [140].

In later sections we study non-local soft constraints based on a natural tendency to minimize the number of distinct surface orientations in the interpretation of man-made objects. But first we begin by studying non-local hard constraints. It is well known that junction constraints alone are not sufficient to ensure realizability [154], and several workers have stated constraints on the labelling of sets of lines not meeting at a junction [77, 90]. In the next section, we formalize constraints on the labelling of a set of lines intersected by a path from line L_A to L_B , where either $L_A = L_B$ or L_A, L_B are parallel. We emphasize that these constraints assume planar surfaces. They may also be applied to drawings of curved objects provided we have sufficient evidence that particular regions are the projection of planar faces. For example, it may be reasonable to

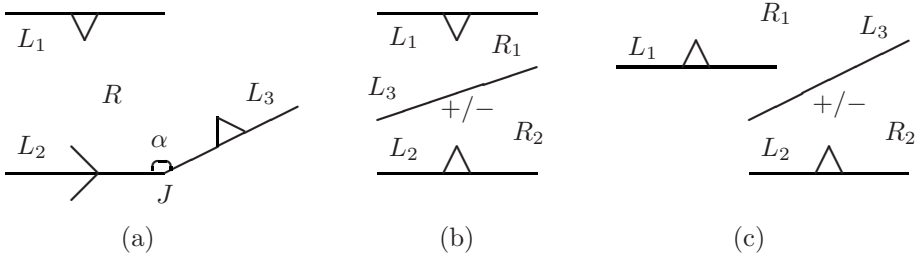


Figure 3.3: (a) The ParOcc constraint: this labelling is impossible if lines L_1, L_2, L_3 are adjacent to region R and lines L_1, L_2 are parallel; (b),(c) the ParCon constraint: this labelling is impossible if lines L_1, L_3 are adjacent to region R_1 , lines L_2, L_3 are adjacent to region R_2 , and lines L_1, L_2 are parallel but are not parallel to L_3 .

assume that a surface which has a polygonal boundary is planar.

3.3 Parallel-Lines Constraint

In this section we assume that a line drawing is an orthographic projection of the edges of opaque polyhedral objects from a general viewpoint. The general viewpoint assumption (GVA) says that a small change in the position of the viewpoint does not alter the configuration of the drawing (including junction types and presence of parallel or collinear lines). The orthographic projection and general-viewpoint conditions imply that parallel lines in the drawing are projections of parallel lines in the 3D scene. If, as will usually be the case, we mark lines as parallel if the angle between them is less than some threshold ϵ (to take into account, for example, rounding errors or a projection which is only approximately orthographic), then the constraints presented in this section become soft rather than hard constraints. The coding of such soft constraints is discussed in detail in Section 3.8.

Figure 3.3 shows combinations of labels that are physically impossible under these assumptions. In all figures we use a generic label ∇ to represent any of the three labels $+$ (convex), $-$ (concave), \rightarrow (occluding with the nearer object below the line). Similarly \triangle represents any of the labels $+$, $-$, \leftarrow . Thus, for example, the label ∇ for line L_1 in Figure 3.3(a) implies that the edge projecting into L_1 lies on the surface projecting into region R . Note that in the ParOcc constraint (Figure 3.3(a)), α can be any angle. A line L is said to be adjacent to a region R if L is part of the boundary of R when considered as a face in the planar graph representation of the drawing. For example, in the ParOcc constraint, L_1, L_2, L_3 are all adjacent to the region R as shown in Figure 3.3(a). Junction J in Figure 3.3(a) is any viewpoint-independent junction (such as a Y, W or L, but not a T junction caused by a depth discontinuity). Let S_1, S_2, S_3

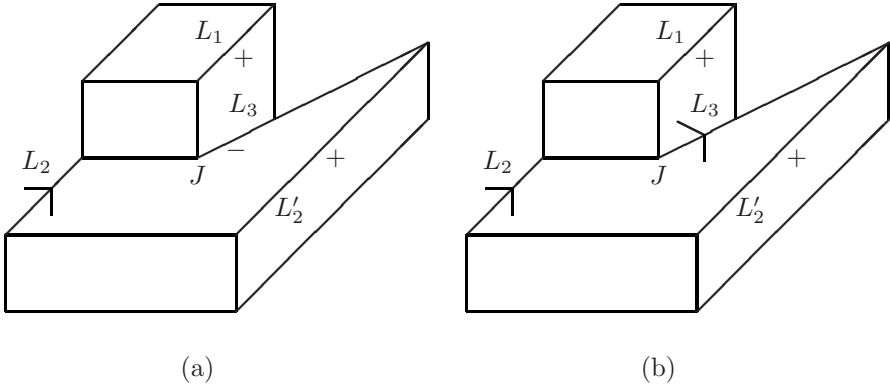


Figure 3.4: The trihedral labelling in (a) violates the ParCon constraint, whereas the tetrahedral labelling in (b) does not

represent respectively the 3D lines in the scene which project into L_1, L_2, L_3 . The labelling in Figure 3.3(a) implies that S_1 and S_2 are coplanar. Since S_2, S_3 intersect, they are also coplanar. Furthermore, since S_1, S_3 are parallel, they are also coplanar. It follows that S_1, S_2, S_3 all lie in the same plane, the face projecting into region R , but this contradicts the occluding label for L_2 . A reflected version of the configuration in Figure 3.3(a), with L_3 to the right of L_2 , is also physically impossible.

In the ParCon constraint (Figure 3.3(b)), the labellings shown are impossible provided L_1, L_2 are parallel but not parallel to L_3 . The proof is omitted since we prove a more general result below. Note that the parallel lines L_1, L_2 do not necessarily face each other; for example, the configuration in Figure 3.3(c) gives rise to the same constraint.

Figure 3.4 shows an example of the use of the ParCon constraint. Applying the trihedral catalogue of Figure 3.2 and the outer-boundary constraint produces a single legal labelling, which includes the four labels given in Figure 3.4(a). However, this labelling violates the ParCon constraint both on L_1, L_2, L_3 and on L_1, L'_2, L_3 . Applying the tetrahedral catalogue [165] instead of the trihedral catalogue produces an alternative labelling which includes the four labels shown in Figure 3.4(b). This labelling, in which junction J is the projection of a tetrahedral vertex, is consistent with the ParCon constraint.

We can generalize the ParOcc and ParCon constraints to a general constraint which can be applied to any pair of parallel lines. Consider a labelled drawing produced by orthographic projection. A *path* is a locus of points in the drawing from a point on a line L_1 to a point on a line L_2 . It may intersect any number of intermediate lines between L_1 and L_2 . A path Π is *parallel* if L_1 and L_2 are

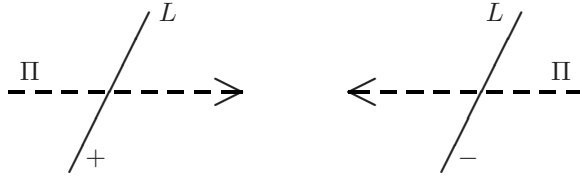


Figure 3.5: The catalogue of strictly monotone intersections

parallel and the tangent to Π is parallel to L_1 and L_2 at each intersection of Π with intermediate lines. In what follows we assume (by rotation of the drawing if necessary) that L_1 and L_2 are horizontal. L_1 may be above or below L_2 in the drawing and Π may approach intermediate lines L from the left or the right (but always horizontally). Figure 3.5 shows two ways in which a path Π , shown as a broken line, may intersect an intermediate line. The direction of Π is indicated by an arrow.

An intersection between a parallel path Π and an intermediate line L is known as *strictly monotone* if L is labelled as in one of the two configurations given in Figure 3.5. The justification for this term is as follows. Imagine an imaginary horizontal line segment H_P in the plane of the drawing such that H_P intersects Π at a point P . Let S_P be the 3D line segment which projects into H_P and which lies on the 3D planar surface visible at P . As P follows the path Π in the drawing and intersects an intermediate line L , S_P twists in 3D so that the depth z_r of its right-hand end either increases or decreases. In both cases shown in Figure 3.5, the depth z_r strictly increases. The angle of line L in Figure 3.5 is arbitrary, but L must be at an angle greater than ϵ to the horizontal to avoid superstrictness problems (i.e. marking a legal labelling as illegal due to rounding errors in the positions of junctions). Note that path Π is shown as a straight line but may be any continuous curve provided that the tangent to Π at its intersection with L is horizontal.

Figure 3.6 shows some ways in which a path may begin or end at the horizontal lines L_1, L_2 . The generic label $\nabla = \{+, -, \rightarrow\}$ is as in Figure 3.3. J represents any viewpoint-independent junction (i.e. not a T junction caused by a depth discontinuity). In Figure 3.6(b),(d) any number of other lines can meet at J , as shown. The angle between L' and $L_1(L_2)$ is arbitrary. No other junction lies on $L_1(L_2)$ between J and point P where Π intersects $L_1(L_2)$.

A parallel path is called *strictly monotone* if

1. It contains only strictly monotone intersections,
2. It begins at a line L_1 in one of the configurations shown in Figure 3.6(a) or (b),
3. It ends at a line L_2 in one of the configurations shown in Figure 3.6(c) or (d),

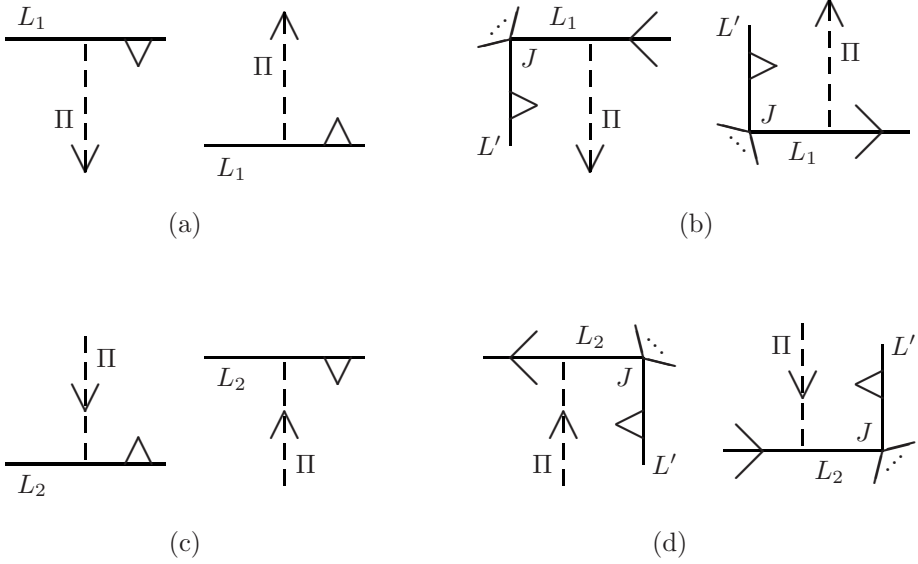


Figure 3.6: (a),(b) How a strictly monotone path Π may begin; (c),(d) how a strictly monotone path Π may end.

4. *Either* it contains at least one intersection *or* it begins with a configuration in Figure 3.6(b) *or* it ends with a configuration in Figure 3.6(d).

It is clear that if the two parallel lines L_1 and L_2 are projections of parallel lines in 3D, then there can be no strictly monotone path joining L_1 and L_2 ; otherwise the line projecting into L_2 would be twisted compared to the line projecting into L_1 . The *parallel-lines* constraint simply says that no strictly monotone parallel path exists in a labelled line drawing. It is easily seen that the ParOcc and ParCon constraints are just special cases of the parallel-lines constraint.

Figure 3.7 shows an example of the use of the parallel-lines constraint. Even assuming trihedral vertices and using the outer-boundary constraint, this drawing still has four legal labellings. All but one of these labellings can be eliminated by applying the parallel-lines constraint to the lines AB, EF with a path Π passing by the intermediate lines BC, ED . The labelling in Figure 3.7(a) is an example of a labelling that violates this constraint. Figure 3.7(b) is the only legal labelling consistent with the parallel-lines constraint.

A strictly monotone path may begin and end at the same line, since a line is necessarily parallel to itself. In this case, we do not need the assumption of an orthographic projection. A strictly monotone path may contain no intermediate lines provided it begins or ends in one of the configurations in Figure 3.6 (b),(d). Figure 3.8 shows a constraint on the labelling of three consecutive lines on the boundary of a region R . Any number of lines can meet at junctions A and B , as shown, and the angles α, β are arbitrary. The labelling shown in Figure 3.8 is

illegal if junctions A, B are viewpoint-independent (i.e. not depth-discontinuity T junctions), there are no other junctions between A and B , and the surface projecting into region R is planar. This constraint is easily deduced from the parallel-lines constraint with a strictly monotone path leaving line AB towards the right and immediately returning to AB .

It is clear that when applying the parallel-lines constraint it is not necessary to specify the actual locus of points traced by path Π : we identify Π with the sequence of lines and regions it intersects. However, if the length t of a path is defined as the number of lines it contains (including the beginning and end lines), then the number of strictly monotone paths of length t is $\Theta(n^t)$ in the worst case where n is the number of lines in the drawing. Therefore, in practice, we recommend limiting the use of the parallel-lines constraint to parallel paths with, for example, at most two intermediate lines. When a line drawing contains many parallel lines, we can often identify parallel planes. For example, a human being sees at a glance that the object depicted in Figure 3.4 has three pairs of parallel planar surfaces. If we know (from their labelling) that lines L_1, L_2 lie on the planar surface projecting into region R , lines L'_1, L'_2 lie on the planar surface projecting into region R' , L_1, L'_1 are parallel, L_2, L'_2 are parallel, but L_1, L_2 are not, then we can deduce that the surfaces projecting into R, R' are parallel planes. In this case, we can extend the parallel-lines constraint as follows. A strictly monotone path can jump from any point in the interior of R to any point in the interior of R' , without affecting the validity of the parallel-lines constraint. We call the resulting constraint the extended parallel-lines constraint.

As a trial of the usefulness of the parallel-lines constraint, we examined the labelled line drawings given in Varley and Martin's paper to illustrate their catalogue of trihedral and tetrahedral junction labellings (Figures 4–9 and 24–166 of [165]). Eliminating mirror-image versions of the same drawing produced a total of 92 drawings. Nine of these drawings are such that if taken out of the context of the paper (where several drawings from different viewpoints are given of each different object), it would not necessarily be interpreted by a human being as indicated by Varley and Martin [165]. These drawings were ignored, which left us with 83 line drawings, containing an average of 14.8 lines each. Out of these 83 drawings, 54 were correctly labelled by applying the outer-boundary constraint (which simply imposes an occluding label on the external boundary of the drawing) and preferring trihedral to tetrahedral labellings. For all the remaining 29 drawings, applying the parallel-lines constraint or its extended version was sufficient to make the correct labelling the unique optimal labelling. In only three cases did we require the extended parallel-lines constraint. In all 29 cases, the number of intermediate lines in the parallel-lines constraint was never greater than two. In the experimental trials described in this section, we applied the parallel-lines constraint (coded as a hard constraint) to all parallel paths in the drawing involving up to two intermediate lines.

In a second trial, we studied the same sample of 83 drawings but this time without applying the outer-boundary constraint. If for each drawing we imagine the same object this time depicted resting on a large flat surface, such as a

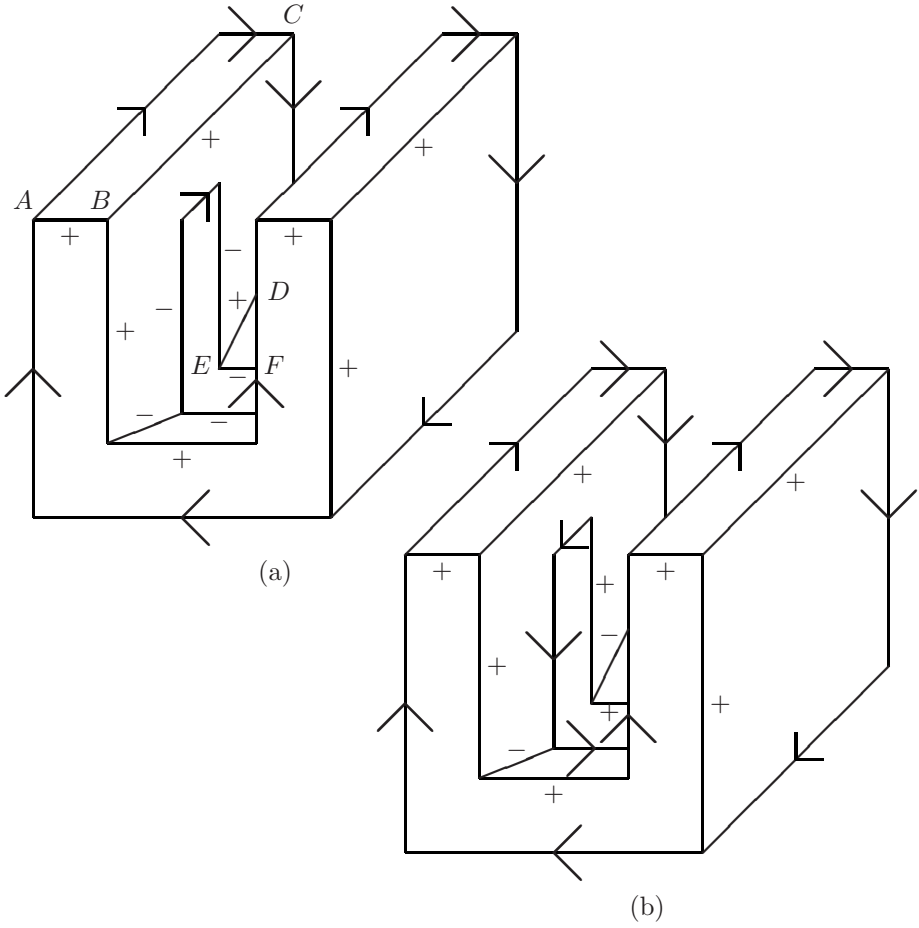


Figure 3.7: (a) A legal labelling according to the trihedral catalogue and the outer-boundary constraint; (b) the unique and correct labelling found by also applying the parallel-lines constraint and propagating

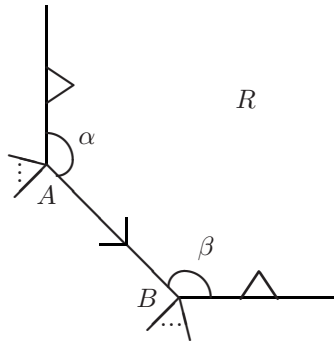


Figure 3.8: This labelling is impossible by the parallel-lines constraint

tabletop, the human interpretation of the drawing does not change, even though the outer-boundary constraint no longer applies. The resulting drawing is only ambiguous for a human being in that we cannot determine whether the object is actually touching the surface or not, and if so along which edges. Surprisingly, the parallel-lines constraint, together with our preference for trihedral junction labellings, was sufficient to identify as optimal exactly the interpretations corresponding to human intuition in all but one of the 83 drawings. Note that the almost 100% success rate in imitating human intuition as to which is the most likely interpretation of these drawings is no doubt due to the fact that Varley and Martin used parallel lines as the main visual cue to avoid ambiguity in their drawings. Other sets of drawings may require reasoning about, for example, collinearity or symmetry.

In [38] we studied the interpretation of line drawings in which lines may be missing due to contrast failure. If contrast failure is assumed to only occur between parallel surfaces, then the parallel-lines constraint is still valid for strictly monotone paths Π beginning and ending at the configurations in Figure 3.6(a) and (c). This follows from the fact that, even if path Π intersects a missing occluding line, the 3D orientation of the occluding and occluded surfaces are identical since these surfaces are assumed to be parallel [38].

In the following section we give a theoretical application, by showing how the parallel-lines constraint can be used in the construction of a junction catalogue for polyhedral vertices.

3.4 A Universal Constraint for Simple Junctions

In this section we derive a constraint on the labelling of a large class of junctions when there is no limit on the number of surfaces which can meet at a vertex.

Definition 3.1 *A simple junction J is any intersection of lines terminating at J such that no two of these lines are collinear.*

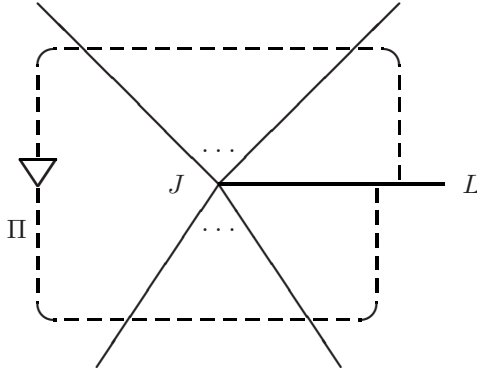


Figure 3.9: A simple junction J .

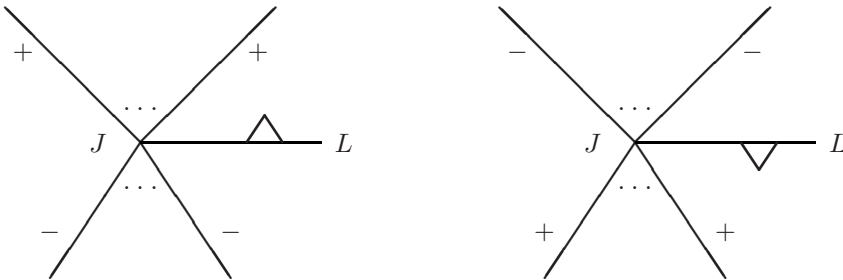


Figure 3.10: The polyhedral-junction constraint: the labellings shown are illegal.

Figure 3.9 shows a simple junction J . For any line L terminating at J , we can apply the parallel-lines constraint to the path Π shown in Figure 3.9, since L is parallel to itself. Note that, in this special case, we no longer require the assumption of orthographic projection to apply the parallel-lines constraint. In this section, therefore, we study the labelling of projections of polyhedral vertices under only a general viewpoint assumption.

It is a direct consequence of the parallel-lines constraint that the labellings shown in Figure 3.10 are illegal. Note that there may be any number $m \geq 0$ of lines arriving at J from above and any number $n \geq 0$ of lines arriving at J from below. The special case when L is the only line terminating at J is clearly also illegal, although this is not a consequence of the parallel-lines constraint.

As an example of the use of the polyhedral-junction constraint, consider the two labellings of a Multi junction in Figure 3.11. The polyhedral-junction constraint tells us that the labelling of Figure 3.11(a) is illegal, whereas the labelling of Figure 3.11 is legal (as will be proved below). The difference between the two configurations is that deleting the concave line leaves a W junction in Figure 3.11(a) and a Y junction in Figure 3.11(b). A $(+,+,+)$ labelling is legal for Y junctions but not for W junctions. Thus Figure 3.11 provides a refinement

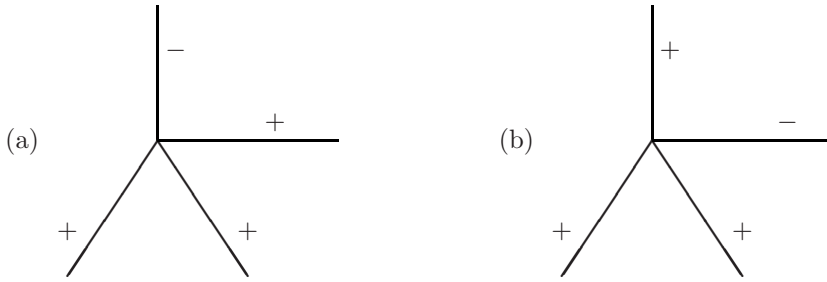


Figure 3.11: The labelling (a) is illegal, but the labelling (b) is legal.

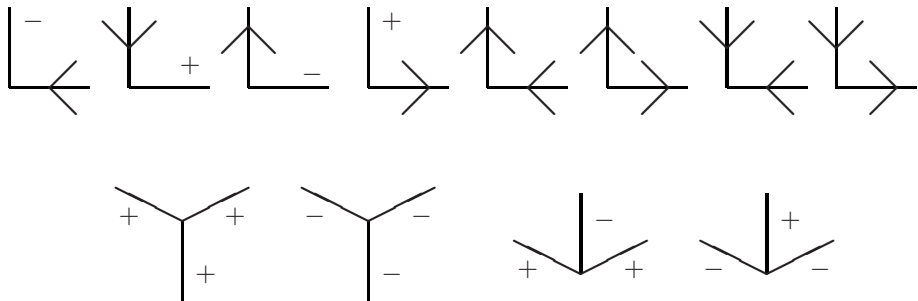


Figure 3.12: Minimal legal polyhedral junction labellings

to the list of labellings for Multi junctions in the tetrahedral catalogue [165].

In fact, the rest of this section is devoted to showing that the constraint of Figure 3.10 is the tightest possible constraint on labellings of simple junctions.

Definition 3.2 A labelled junction J_1 is a sublabelling of a labelled junction J_2 if J_1 can be obtained from J_2 by deleting some number (possibly zero) of lines from J_2 .

For example, a W junction labelled $+-+$ has three L junction labellings, namely $+-, -+, ++$, as sublabellings.

Figure 3.12 shows some legal polyhedral junction labellings of L, Y and W junctions.

Lemma 3.3 All labellings of a simple junction J which are not illegal by the polyhedral-junction constraint (Figure 3.10) contain one of the labelled junctions in Figure 3.12 as a sublabelling.

Proof: Let J be a labelled junction which has none of the labelled junctions in Figure 3.12 as sublabellings. We will show that J is illegal by the polyhedral-junction constraint.

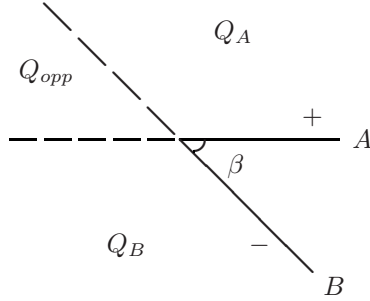


Figure 3.13: Q_{opp} contains either only + lines or only - lines, Q_A contains only + lines and Q_B only - lines.

J cannot have two or more occluding labels, since this would imply that it necessarily had as a sublabelling an L junction with two occluding labels, and all such labellings of L junctions are listed in Figure 3.12. Suppose that J has just one line A with an occluding label, and suppose that A is labelled with an arrow pointing towards the junction (the proof for other case being entirely similar). Since J has neither of the first two L junction labellings given in Figure 3.12 as sublabellings, it must be of the form shown on the left of Figure 3.10 and hence must be illegal.

If each line meeting at J is labelled +, then since J has no Y junction, labelled + + +, as a sublabelling, there must be an angle $\alpha > \pi$ between two adjacent lines. But then J is again illegal by the polyhedral-junction constraint. A similar argument applies if all the lines meeting at J are labelled -.

The only case left to consider is when J has no occluding labels and at least one + label and at least one - label. Let A and B be two lines meeting at J such that A and B have different labels and such that the angle β between A and B is minimal. Without loss of generality, suppose A is labelled + and B is labelled -. This situation is illustrated in Figure 3.13. The extensions of lines A and B divide the plane into four quadrants as shown in Figure 3.13. In Q_{opp} there cannot be two lines with different labels; if not this would contradict the minimality of β . In Q_A there can be no line labelled -, otherwise J would have the W junction labelling - + - as a sublabelling. Similarly, in Q_B there can be no line labelled +, otherwise J would have the W junction labelling + - + as a sublabelling. The labelling of J is then illegal by the polyhedral-junction constraint (with $L = A$ if Q_{opp} contains only + labels and with $L = B$ if Q_{opp} contains only - labels). ■

Lemma 3.4 *Let J be a labelled simple junction. If J has a legal polyhedral-junction labelling J_0 as a sublabelling, then J is also a legal polyhedral-junction labelling.*

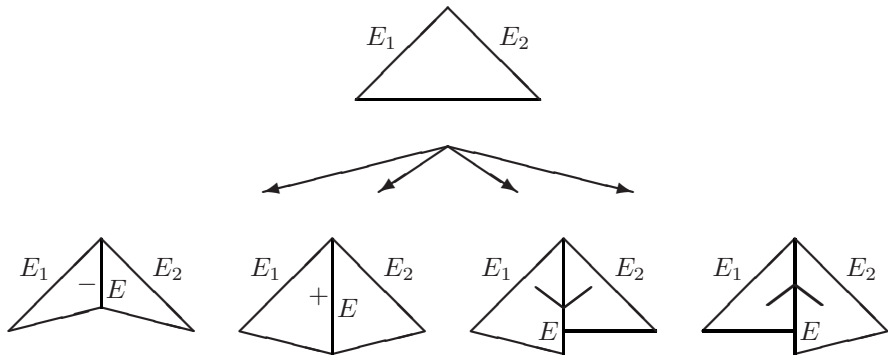


Figure 3.14: Introduction of a new edge E of any type within a surface bounded by non-collinear edges E_1, E_2

Proof: Let J_0 be a legal polyhedral labelling which is a sublabelling of J . Let V_0 be a vertex which projects into J_0 . Suppose that L is a line of J which does not occur in J_0 . L must lie in some region R between adjacent lines L_1, L_2 meeting at J_0 . Let S be the surface which projects into R .

Consider first the case where S is one of the surfaces meeting at vertex V_0 and that S is bounded by 3D edges E_1, E_2 projecting into lines L_1, L_2 . Partition S into two distinct faces S_1, S_2 separated by a 3D line E which projects into line L . Since E_1, E_2 are not collinear, by our assumption that J contains no collinear lines, it is possible to rotate S_1 about E_1 and S_2 about E_2 , by small angles ϵ_1, ϵ_2 , so that E becomes either a concave or a convex edge. By also creating a hidden edge, it is possible, in a similar manner, to construct an occluding edge. Figure 3.14 shows how it is possible to introduce a concave, convex or occluding edge in a vertical surface. For sufficiently small ϵ_1, ϵ_2 , the labelling of lines L_1, L_2 of J_0 remains unchanged. The cases in which one or both of E_1, E_2 occlude surface S can be dealt with by entirely similar constructions.

By repeating this operation for each line L of J not in J_0 , we can clearly construct a vertex V which projects into J . ■

The following theorem now follows immediately from Lemmas 3.3 and 3.4.

Theorem 3.5 *Let J be a labelled simple junction. J is a legal polyhedral junction labelling if and only if it satisfies the polyhedral-junction constraint (Figure 3.10).*

Huffman [77] characterized all legal labellings of simple polyhedral junctions in terms of the possibility of finding in dual space a closed trace corresponding to the junction. Theorem 3.5 provides a more explicit characterization.

The table in Figure 3.15 gives the number of labellings which are realizable as projections of trihedral, tetrahedral or polyhedral vertices, for each junction

Junction type		Trihedral	Tetrahedral	Polyhedral	Combinatorial
L		6	8	8	16
Y		5	32	52	64
W		3	28	52	64
T		4	20	24	64
Multi		–	9	240	256
Peak		–	11	240	256
K		–	8	139	256
ψ		–	–	94	256
X		–	–	48	256

Figure 3.15: The number of labellings for junctions of arity up to four

type of arity up to four. The figures for trihedral vertices refer to the basic Huffman–Clowes catalogue [20, 75]. The figures for tetrahedral vertices refer to Varley and Martin’s catalogue [165]. The lists of legal polyhedral labellings for L, Y, W, Multi and Peak junctions are easily obtained from Theorem 3.5. The list of legal polyhedral T junction labellings can be found in [77], and the lists of legal polyhedral labellings for K, Y and X junctions were obtained by exhaustive search making ample use of the parallel-lines constraint (which incidently was not quite sufficient on its own to eliminate all illegal labellings since two illegal labellings for the ψ junction also satisfy the parallel-lines constraint).

The figures in the rightmost column are the number of combinatorially possible labellings. It is clear that the general polyhedral-junction constraints are far too weak to obtain a unique labelling of a line drawing. Consider a catalogue containing junction types belonging to a set C (e.g. $C = \{L, Y, W, T\}$ in the trihedral catalogue). Let N be the total number of line ends in the catalogue (e.g. $N = 2 + 3 + 3 + 3$ for the trihedral catalogue). Assuming a uniform distribution of junction types, a drawing with n lines contains on average $\frac{2n}{N}$ junctions of each type in C . For each junction type $t \in C$, let p_t be the ratio of the number of legal labellings to the number of combinatorially possible labellings. Since each line can have one of four labels, the average number of global legal labellings of a drawing with n lines is thus

$$4^n \prod_{t \in C} p_t^{\frac{2n}{N}}.$$

We can therefore calculate that the average number of global legal labellings of a drawing with n lines is $\Omega(1.49^n)$ for the tetrahedral catalogue and $\Omega(2.8^n)$

for the polyhedral catalogue. This can be compared with the $O(0.73^n)$ average number of legal labellings using the trihedral catalogue. These figures highlight the need for new constraints involving small subsets of lines to reduce the exponential number of global legal labellings. On the drawings tested, the outer-boundary constraint and the parallel-lines constraint turned out to be the most powerful such constraints. In the following sections we establish constraints on the labellings of sets of lines intersected by a another form of path, based on depths rather than orientations of surfaces.

3.5 Lines Sharing the Same Two Regions

In this section we do not need the assumption of orthographic projection, but we assume that the drawing is a projection of a polyhedral scene from a general viewpoint. In order to introduce a generic constraint on lines intersected by a path in the drawing, we first study the special case of a pair of lines adjacent to the same two regions. In a line drawing of a polyhedral scene, each line separates two distinct regions of the drawing. The 2Reg constraint applies when two non-collinear lines are adjacent to the same two regions. For example, in both drawings in Figure 3.16, the region above line AB is the same as the region below line CD and the region below line AB is the same as the region above line CD . We represent this generic situation by the diagram in Figure 3.17(a), where the straight lines L_1, L_2 labelled l_1, l_2 are lines present in the drawing and the other lines are construction lines which simply indicate which side of the lines share the same region in the drawing. This is, in fact, a very common situation since it occurs at every L junction. There is a distinct and less common case illustrated in Figure 3.17(b).

Figure 3.17 gives the list of legal labellings of the two lines in both cases. Note that the angles between the two lines are arbitrary. For example, if the two lines are parallel, this does not provide a stronger constraint. This 2Reg constraint follows from simple reasoning about the depth of scene points. As an example, consider the case in which $l_1 = +$. It is clear that $l_1 = +$ (convex) implies that a scene point just to the right of line L_2 is nearer to the viewer than a point just to the left of L_2 . This, in turn, implies that $l_2 = \uparrow$. An important point is that we do not need an assumption of trihedral vertices to obtain this constraint. This is, therefore, a very general constraint, from which can be deduced, for example, the L junction constraint in the general polyhedral catalogue. Indeed the list of eight labellings in Figure 3.17(a) coincides with the list of eight legal L junction labellings given in Figure 3.12.

An illustration of the strength of the 2Reg constraint is that it can be used to detect some classic examples of drawings of impossible polyhedral objects. For example, both of the drawings in Figure 3.16 have legal labellings according to the trihedral catalogue, but none of these labellings is consistent with the 2Reg constraint. In all trihedral labellings, lines AB and CD have labels $+$ and $-$, respectively. But the $+ -$ labelling is illegal according to the 2Reg constraint.

A special case of the 2Reg constraint that needs to be considered separately

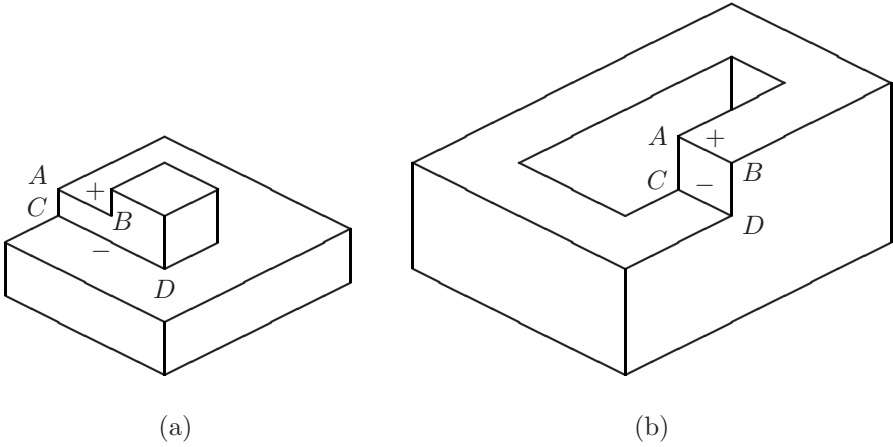


Figure 3.16: Two impossible pictures of polyhedral objects, both detected by the 2Reg constraint ((b) is known as Sugihara’s box [155])

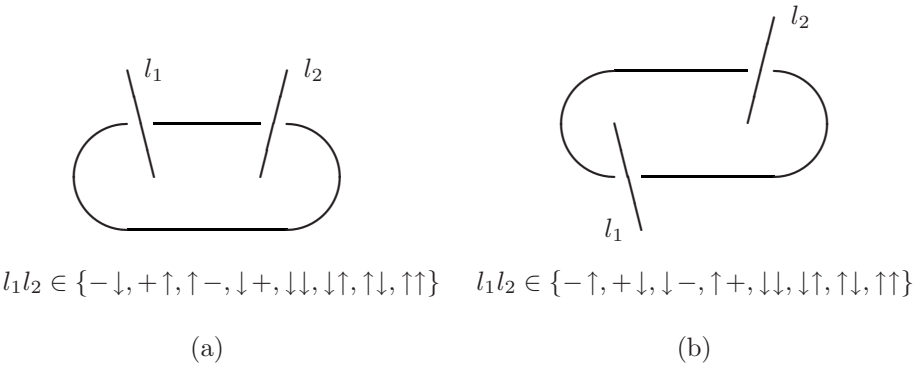


Figure 3.17: The 2Reg constraint

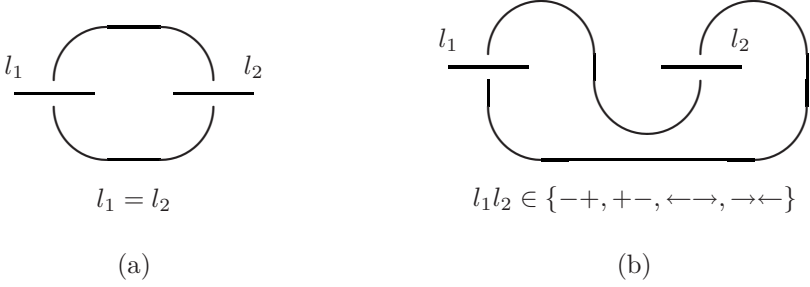


Figure 3.18: The Col2Reg constraint

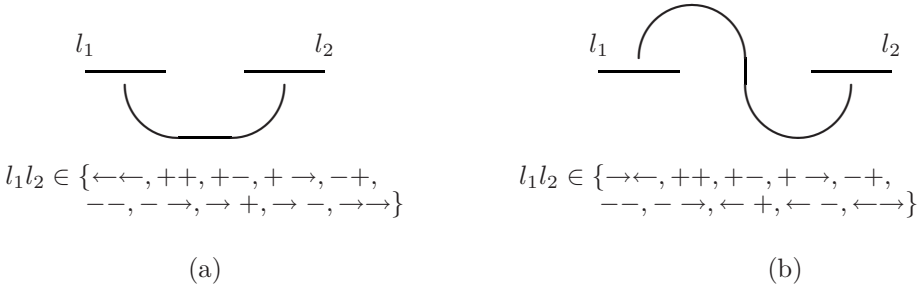


Figure 3.19: the Col1Reg constraint

occurs when the two lines are collinear. In this case, the lists of legal labellings are given in Figure 3.18. These lists are completely different from those for the 2Reg constraint (Figure 3.17). This new constraint, which we call Col2Reg, was derived using the GVA, which implies that collinear lines in the drawing are projections of collinear lines in the 3D scene.

When two lines share only one common region, then without any other information we can deduce nothing about the possible labellings of the lines. However, in the case that the two lines are collinear, this gives rise to a constraint which we call the Col1Reg constraint. The lists of legal labellings are given in Figure 3.19.

To illustrate the utility of the Col2Reg and Col1Reg constraints, consider the example line drawing in Figure 3.20. After applying the trihedral catalogue and the outer-boundary constraint, seven of the line labels are still ambiguous. However, by applying the Col2Reg constraint to the pair of collinear lines (AB , CD) and the Col1Reg constraint to the pairs of collinear lines (EF , GH) and (IJ , HK), a unique labelling is determined after propagation. For example, the fact that the line AB is labelled as convex (+) implies immediately by Col2Reg (case (b) of Figure 3.18) that line CD is concave (-) rather than occluding (\rightarrow). Similarly, the concave label (-) for line EF implies by Col1Reg (case (a)

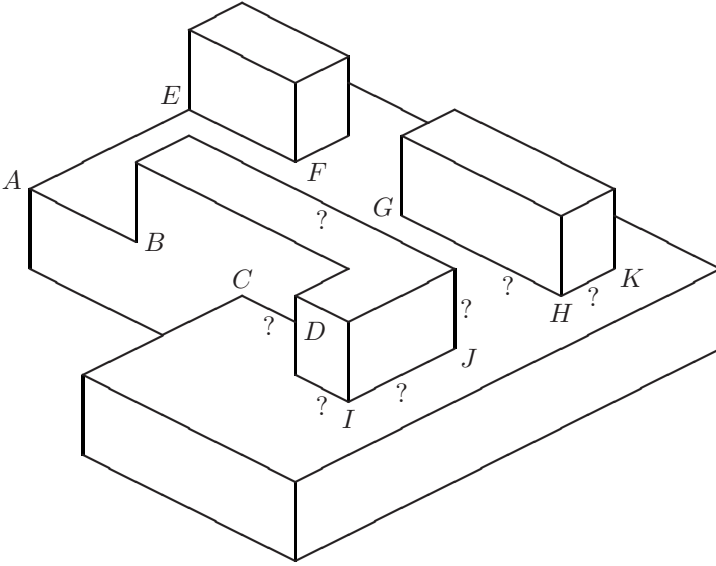


Figure 3.20: A line drawing which has seven ambiguous labels (marked with a question mark) after applying the trihedral catalogue and the outer-boundary constraint, but which has a unique labelling after applying the Col1Reg and Col2Reg constraints

of Figure 3.19) that line GH cannot be labelled as \leftarrow .

Kirousis [90] gave a constraint for L-chains (sequences of lines connected by L junctions) in drawings of objects with trihedral vertices. Stated succinctly, the L-chain constraint says that a $+$ or $-$ label for a line on an L-chain C uniquely determines the labelling of all lines on C and that an occluding label for a line on C reduces the number of possible labels for all other lines to at most three. This constraint is a direct consequence of the 2Reg and Col2Reg constraints. It therefore follows that the L-chain constraint also holds for drawings of objects with non-trihedral vertices.

3.6 Cyclic-Path Constraint

In Figure 3.17(a),(b) it was necessary to make explicit a cyclic path intersecting the two lines since the constraint is not identical for the two distinct cases shown in Figure 3.17. Furthermore, a completely different constraint is obtained when the lines are collinear (Figure 3.18). The 2Reg and Col2Reg constraints can be generalized to the case of a path passing through $n > 2$ lines. Since, even for $n = 3$ we identified 95 distinct cases, we prefer to give a generic constraint valid for all cases and all values of n . The resulting cyclic-path constraint, given below, was inspired by but also strictly generalizes Huffman’s cut-set rule based on reasoning in dual space [77]. Our rule allows for paths of arbitrary shape which

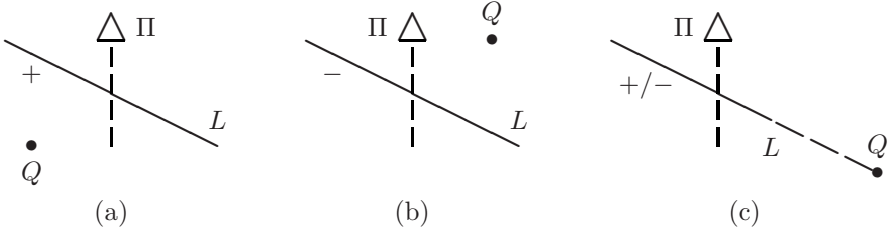


Figure 3.21: (a),(b) The two types of strictly positive intersections; (c) a null intersection.

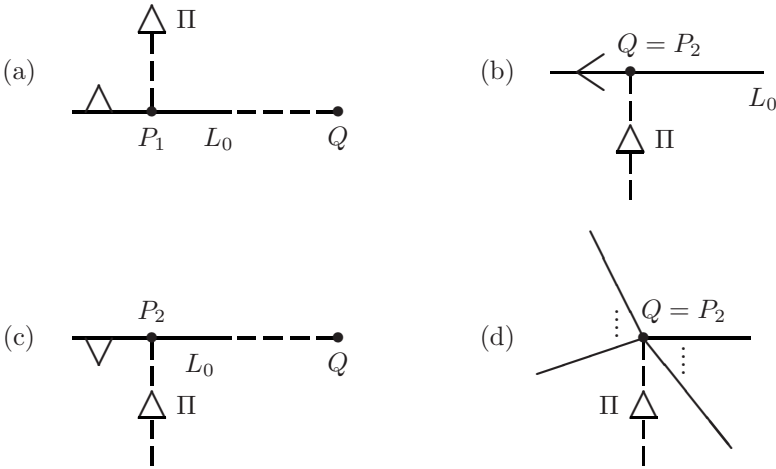


Figure 3.22: (a) How strictly positive cyclic paths begin; (b)–(d) how strictly positive cyclic paths end.

do not necessarily begin and end at the same point, and we correct Huffman’s treatment of intersections with occluding lines. It can be considered as an application of Draper’s sidedness reasoning [59] along a path in the drawing.

A *cyclic path* is a path in a drawing which begins at a point P_1 on line L_0 and ends at a point P_2 on L_0 . It is *anchored* at a point Q collinear with L_0 . Any two or three of the points P_1 , P_2 and Q may, and often do, coincide. Suppose that a cyclic path passes through regions R_1, \dots, R_t which are projections of planar surfaces S_1, \dots, S_t , and let z_1, \dots, z_t denote the 3D depths of these surfaces at a point which projects into Q in the drawing. Then a labelling is clearly illegal if it implies $z_1 \leq z_2 \leq \dots \leq z_t \leq z_1$ with at least one of the inequalities being strict.

We define the intersection of a path Π with a line L , separating regions R_i and R_{i+1} , to be *strictly positive* if the labelling of L implies that $z_i < z_{i+1}$. This is the case for the intersections in Figure 3.21(a),(b), in which Q lies on the same side of L as R_i (respectively R_{i+1}) if L is labelled $+$ ($-$). An intersection

is said to be *null* if the labelling of L implies $z_i = z_{i+1}$. This is the case for the intersection shown in Figure 3.21(c), where Q is collinear with L and L is labelled either convex or concave. A cyclic path Π is *strictly positive* if

1. Π contains only strictly positive or null intersections,
2. Π begins as shown in Figure 3.22(a),
3. Π ends as shown in Figure 3.22(b),(c) or (d),
4. *Either* Π ends as in Figure 3.22(b) and Q does not coincide with a junction *or* Π contains at least one strictly positive intersection.

In the configuration of Figure 3.22(b), if the point Q does not coincide with a junction, then the occluding label implies a strict depth inequality between the two surfaces at Q ($z_1 < z_t$). In the configurations illustrated in Figure 3.22(a),(c), Q may lie either to the left or right of P_1 or P_2 . In Figure 3.22(d), path Π may arrive from any angle. Whatever the angle, the surface S_t projecting into region R_t through which Π arrives at $P_2 = Q$ either intersects or passes behind the vertex projecting into Q , implying $z_1 \leq z_t$.

The *cyclic-path constraint* simply says that a strictly positive cyclic path is illegal since a net increase in depth as we traverse a cycle of surfaces is physically impossible. To avoid superstrictness problems, we could classify an intersection as strictly positive only if the perpendicular distance between the anchor point Q and the extension of L exceeds some minimum value δ . Furthermore, we could choose to only allow the cases in Figure 3.21(c) or Figure 3.22(a),(c) in which the anchor point Q actually lies on line L or L_0 (respectively), to avoid superstrictness problems due to accidental collinearity of Q with these lines. The 2Reg and Col1Reg constraints are just special cases obtained by studying cyclic paths containing either one or zero intermediate lines. The Col2Reg constraint can be obtained from the cyclic-path constraint and the parallel-lines constraint (applied to an imaginary line lying anywhere on a surface but not parallel to the lines in the drawing).

A path is not specified by an actual locus of points, but rather by the sequence of regions and lines it intersects. For any given cyclic path intersecting lines L_0, \dots, L_{t-1} , we do not need to test the cyclic-path constraint for every point Q collinear with L_0 . Imagine the extension of L_0 divided into t segments by its intersection with the extensions of lines L_1, \dots, L_{t-1} . It suffices to test the constraint for $2t - 1$ points Q , one per segment, together with each of the $t - 1$ intersection points. As with the parallel-lines constraint, we recommend applying the cyclic-path constraint only to cyclic paths involving only a small number of lines, since in the worst case there are $\Theta(n^t)$ cyclic paths of length t in a drawing containing n lines.

It is important to note that the cyclic-path constraint does not require an orthographic projection and is hence more generally applicable than the parallel-lines constraint. Applied to our sample of drawings from [165], we found that the cyclic-path constraint invalidated a subset of the labellings invalidated by the parallel-lines constraint (where both constraints were applied to all paths

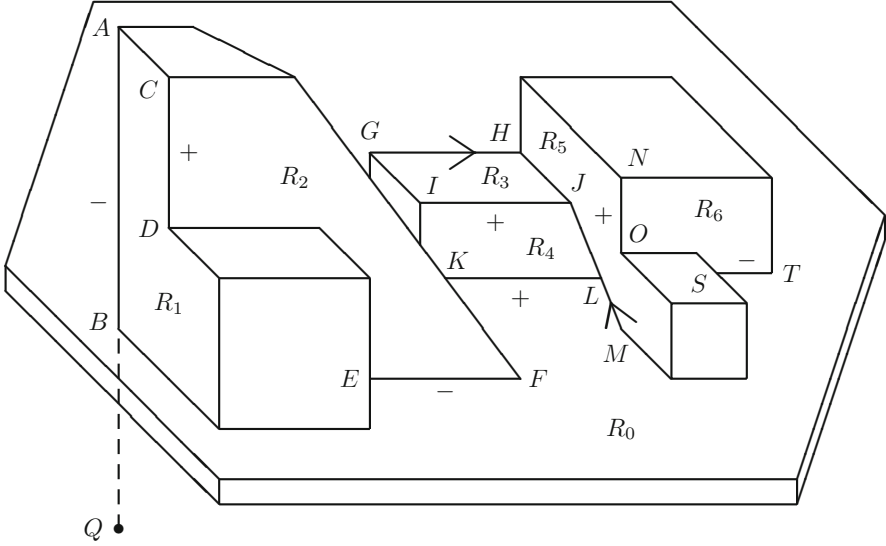


Figure 3.23: Three examples of applications of the cyclic-path constraint

involving up to two intermediate lines). As a concrete example, the polyhedral-junction constraint can easily be deduced from the cyclic-path constraint, instead of the parallel-lines constraint. Nevertheless the cyclic-path constraint and parallel-lines constraint often complement each other. As an illustration of the strength of the cyclic-path constraint, consider the drawing in Figure 3.23. The labels shown are consistent with the trihedral catalogue, the outer-boundary constraint and the parallel-lines constraint. However, a cyclic path passing through regions R_1, R_2, R_0 and intersecting lines AB, CD and EF invalidates the labelling $(-, +, -)$ for (AB, CD, EF) if we choose the anchor point Q as shown. Similarly, a cyclic path passing through regions R_3, R_4, R_0 and intersecting lines GH, IJ, KL invalidates the labelling $(\rightarrow, +, +)$ for (GH, IJ, KL) if the anchor point is chosen to be any point on GH . A more subtle example is the labelling $(\leftarrow, +, -)$ for (LM, NO, ST) invalidated by a cyclic path passing through regions R_5, R_6, R_0 , intersecting lines LM, NO, ST and anchored at any point of LM .

3.7 Parallel Junctions on Distinct Faces

The constraints described in this section express a preference for a small number of distinct orientations of 3D edges in the scene reconstructed from the drawing. We do not assume planar faces, and these preference constraints can thus be applied even when surfaces are curved. In this case, the straight lines depicted

in the figures in this section represent tangents to the curved lines which meet at a junction. All the constraints on junction pairs given in this section are only valid if the total number of distinct orientations of 3D edges meeting at the corresponding pair of vertices is equal to three. Thus these constraints certainly hold if we can assume that all faces in the scene are parallel to one of only three planes. In general, this is too strong an assumption and hence these constraints simply express a preference for 3D interpretations involving pairs of trihedral vertices V_1, V_2 such that each of the planes meeting at V_1 is parallel to one of the planes meeting at V_2 . Such interpretations are more likely in the case of man-made objects, which tend to have many parallel planes, by a simple application of Bayes's theorem.

Before giving our constraints on parallel junction pairs (pairs of junctions involving at least one pair of parallel lines), we reproduce in Figure 3.24 the Huffman–Clowes (trihedral) catalogue of legal labellings of L, Y and W junctions. We follow Parodi and Torre [127] in dividing the set of labellings for Y and W junctions into subcategories $Y(+)$, $Y(-)$ and $W(+)$, $W(-)$. For example, a $Y(+)$ junction is the projection of a convex vertex whereas a $Y(-)$ is the projection of a concave vertex or a saddle point. Knowledge of the positions of the vanishing points of all lines (under perspective projection) is sufficient to classify Y and W junctions as $+$ or $-$ [127]. For notational convenience, we also divide the six labellings of L junctions into four subcategories (called $L_1(+)$, $L_1(-)$, $L_2(+)$, $L_2(-)$) as shown in Figure 3.24. Let V be a trihedral vertex projecting into an L junction J . One of the edges meeting at V is not visible in the drawing. Let H denote the projection in the drawing of this hidden edge. When extended, the two visible lines meeting at J divide the plane of the drawing into four quadrants. For each of the four subcategories $L_1(+)$, $L_1(-)$, $L_2(+)$, $L_2(-)$, the hidden line H lies in a different quadrant. In the Huffman–Clowes catalogue, the set of legal labellings for an L junction is simply the union of the sets of legal labellings for $L_1(+)$, $L_1(-)$, $L_2(+)$, $L_2(-)$ junctions.

Consider any pair of junctions J_1, J_2 in the drawing. Suppose that the two vertices V_1, V_2 projecting into these junctions are both trihedral and, furthermore, that each of the three edges meeting at V_1 is parallel to one of the edges meeting at V_2 . Then the list of possible labellings of J_1, J_2 is given by the table of possible junction-types in Figure 3.25 (if J_1, J_2 are Y or W junctions), Figure 3.26 (if J_1 is a Y or W junction and J_2 is an L junction) or Figure 3.27 (if J_1, J_2 are both L junctions). We call the corresponding constraint the Par3-3, Par3-2 or Par2-2 constraint, according to the number of lines meeting at junctions J_1 and J_2 . For example, assuming trihedral vertices and that parallel lines are projections of parallel 3D lines, the pair of Y junctions at the top of the left hand column of Figure 3.25 must be the same sign (i.e. $Y(+)$ $Y(+)$ or $Y(-)$ $Y(-)$). As another example, the sixth configuration of pairs of L junctions given in Figure 3.27 has only two legal labellings which involve only three distinct face orientations. The constraints in Figures 3.25-3.27 were derived by exhaustive search over all possible types of vertex pairs formed by parallel planes, making use of the catalogue of labelled trihedral junctions in wireframe

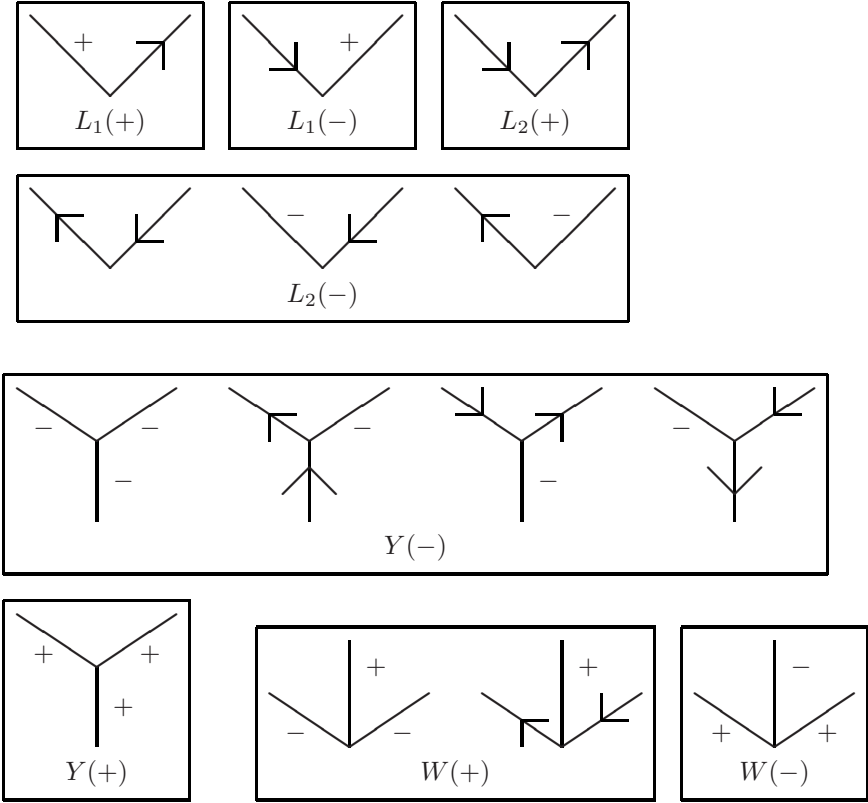


Figure 3.24: The catalogue of labelled L, Y and W junctions which are projections of trihedral vertices.

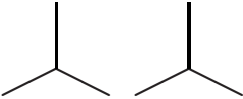
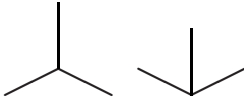
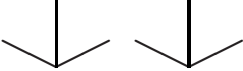
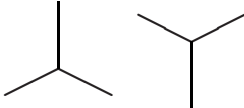
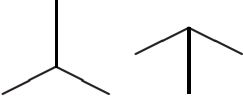
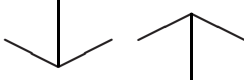
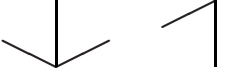
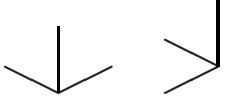
same sign	opposite sign
	
	
	
	

Figure 3.25: The Par3-3 constraint: pairs of junctions in the left-hand column are of the same sign; pairs of junctions in the right-hand column are of opposite signs

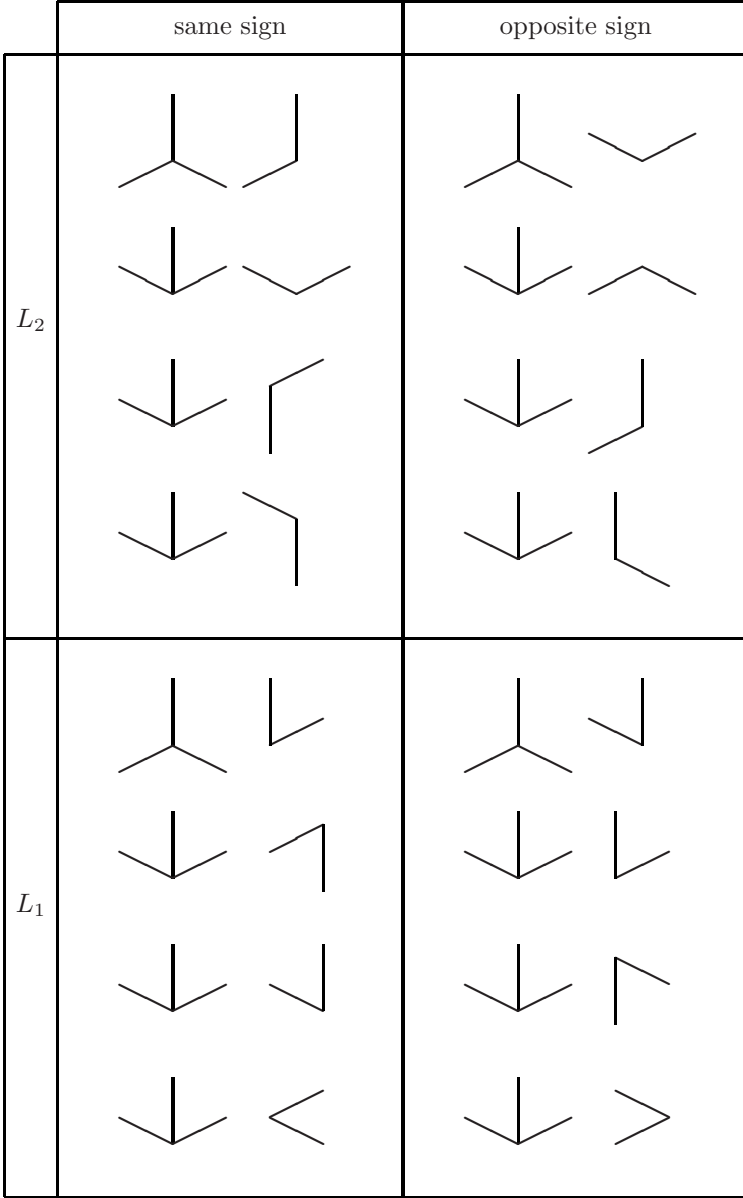


Figure 3.26: The Par3-2 constraint: pairs of junctions in the left-hand column are of the same sign; pairs of junctions in the right-hand column are of opposite signs; L junctions in the top half of the figure are L_2 junctions; L junctions in the bottom half of the figure are L_1 junctions.

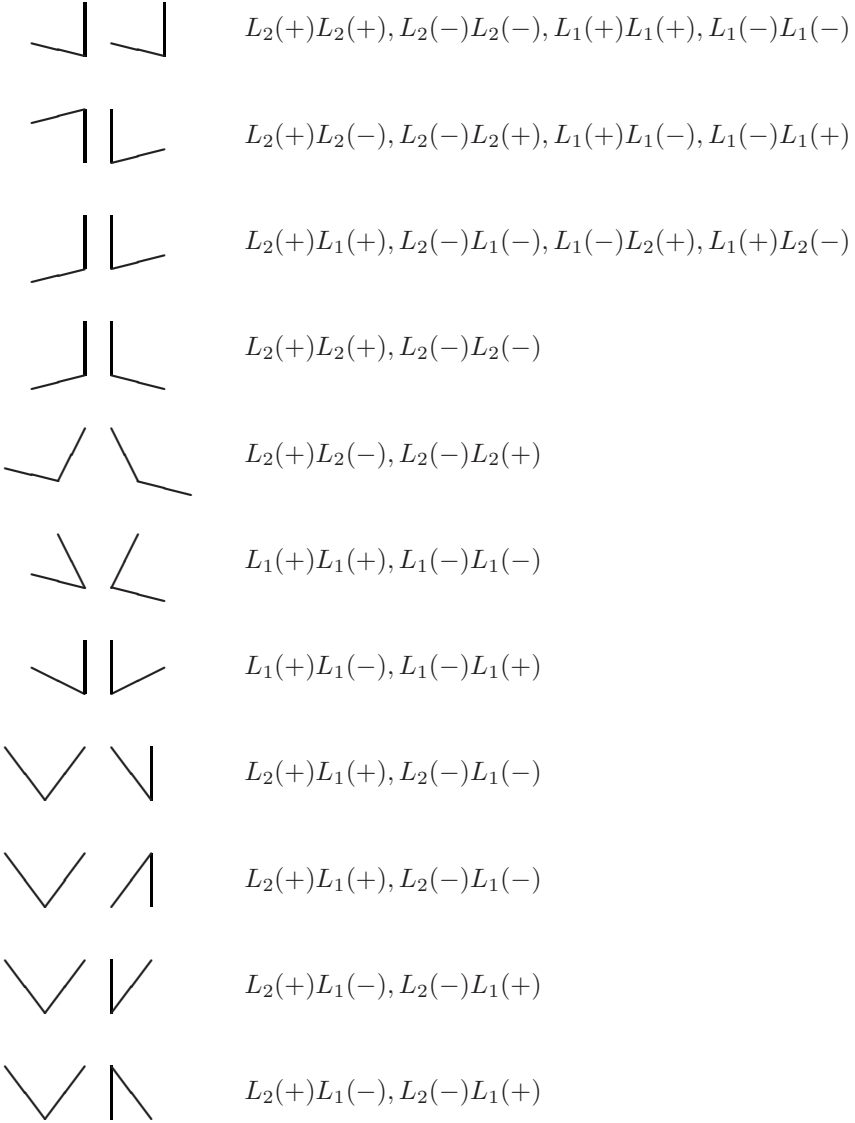


Figure 3.27: The Par2-2 constraint: for each pair of L junctions shown on the left, the possible junction types are shown on the right.

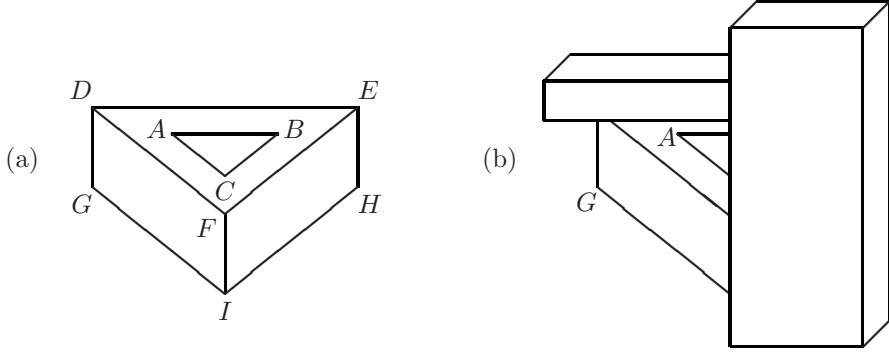


Figure 3.28: (a) A line drawing which has eight legal labellings according to the trihedral catalogue and the outer-boundary constraint but has only one labelling satisfying the Par3-2 constraint; (b) a line drawing which can be uniquely labelled using the Par2-2 constraint.

projections [42].

Consider the drawing in Figure 3.28(a). The trihedral catalogue and the outer-boundary constraint imply the unique labelling of some lines, but triangle ABC remains very ambiguous. Indeed, triangle ABC has eight legal labellings. Note that ParOcc eliminates three of these eight labellings, but this still leaves five physically possible labellings. Par3-2 applied to the pair of junctions (D, A) indicates that A is an $L_1(+)$ junction, which implies a unique labelling for the whole drawing, corresponding to the most natural interpretation. Note that applying Par3-2 to any of the junction pairs (B, E) , (C, F) , (C, I) or Par2-2 to either of the junction pairs (G, A) , (B, H) also implies the same unique and correct interpretation of the drawing. Therefore, Par3-2 and Par2-2 would allow us to interpret correctly triangle ABC even if all but one of the junctions D, E, F, G, H, I were occluded in the drawing. Figure 3.28(b) shows such an example.

The encoding of the Par3-3, Par3-2 and Par2-2 constraints as valued constraints is discussed in detail in Section 3.8.

3.8 Encoding of Soft Constraints

In a Valued Constraint Satisfaction Problem (VCSP), a valued constraint is represented by a local cost function and the aim is to minimize the sum of these cost functions [50, 140]. State-of-the-art VCSP solvers [53] maintain a form of soft arc consistency (such as FDAC or VAC) during branch-and-bound search and use appropriate variable/value ordering heuristics [98]. For example, FDAC propagates hard constraints (represented by infinite costs) in all directions and propagates finite costs towards earlier variables in the instantiation order so as to produce a better lower bound with which to prune the search tree. This

chapter is concerned uniquely with hard and soft constraints and their encoding as a VCSP; algorithmic aspects of VCSPs are treated in Chapter 8.

For a given optimization problem, many different encodings are possible as a VCSP. We say that the encoding of a line drawing labelling problem is *faithful* if the set of optimal solutions to the VCSP coincide with the most likely interpretations. A simple, but naive, encoding associates a fixed cost to the violation of each soft constraint. Consider a pair of parallel lines, involved in a large number m of parallel-lines or parallel-junctions (Par3-3, Par3-2 or Par2-2) constraints. It is possible, by a single violation of the GVA, that these two lines are in fact projections of 3D edges with two distinct 3D orientations, meaning that these m constraints are invalid. The naive encoding is therefore not faithful, since the likelihood that the GVA is violated is independent of m .

A more faithful encoding is obtained by the introduction of auxiliary variables, which of course has the drawback of increasing the size of the search space. For example, we can introduce a variable par_S for each set S of parallel lines in the drawing (par_S being true iff all lines in S are projections of parallel 3D edges). Then each parallel-lines constraint along a parallel path Π beginning and ending at lines in S can be encoded as a hard conditional constraint of the form

$$par_S \Rightarrow (\text{the parallel-lines constraint is satisfied by the lines on } \Pi).$$

An assignment $par_S = \text{false}$ incurs a fixed finite penalty corresponding to the violation of the GVA.

In the case of imperfect line drawings, such as those derived from freehand sketches, we can obtain an optimal partitioning of 2D lines into sets S_1, \dots, S_r of near-parallel lines, by using, for example, a linear-time optimal segmentation algorithm applied to the sorted array of their angles [35]. An auxiliary boolean variable par_{S_i} would be required for each S_i containing at least two distinct lines involved in at least one parallel-lines or parallel-junctions constraint.

A Par3-3 constraint between J and J' can be simply encoded as a hard conditional constraint of the form

$$par_{S_p} \wedge par_{S_q} \wedge par_{S_r} \Rightarrow (\text{the Par3-3 constraint is satisfied on } J, J'),$$

where the lines meeting at junctions J and J' belong to the sets of parallel lines S_p, S_q, S_r . In the case of Par3-2 constraints, the corresponding conditional constraint is a soft constraint, since even if the two visible lines are projections of parallel 3D edges, we are merely expressing a preference for vertices in which the hidden third edge is parallel to the visible third edge. In other words, the constraint

$$par_{S_p} \wedge par_{S_q} \wedge par_{S_r} \Rightarrow (\text{the Par3-2 constraint is satisfied on } J, L)$$

can be violated with a fixed finite cost chosen as a function of the likelihood that the hidden third edge is parallel to the visible third edge. (As above, the lines meeting at J belong to the sets S_p, S_q, S_r .) The encoding of the Par2-2 constraint is entirely similar.

Alternative, more faithful, encodings exist at the cost of the introduction of more auxiliary variables. For example, we could introduce an auxiliary variable V_L with domain $\{0, \dots, r\}$, for each L junction L , where $V_L = i$ if the hidden third line is parallel to the lines in S_i and $V_L = 0$ if this hidden line is parallel to no other visible lines in the drawing. In other words, we attempt to explicitly reconstruct the directions of hidden lines.

3.9 Non-Manifold Scenes

We have assumed throughout this chapter that objects are 3D manifolds with a 2D manifold surface. Some workers have studied the case of non-manifold objects. In particular, the set of scenes containing objects which may be either solid or wafer-thin is often known as the origami world. Since the origami world allows many more vertices, such as those formed by cutting or folding a sheet of paper, the semantic labelling constraints for origami world objects are generally considered to be weaker than the equivalent constraints for manifold objects [87, 125]. However, this is under the assumption that wafer-thin and solid occluding edges are assigned the same label. Introducing a new label for a wafer-thin occluding edge produces a more informative labelling scheme, provided that transitions from wafer-thin edges to solid edges cannot occur (or are unlikely, in a soft constraint formulation of the labelling problem).

When more than one manifold object can occur in a scene, the union of the objects may not be manifold. This occurs, for example, when the edge of one polyhedral object touches a face of another polyhedral object. The resulting edge has the properties of a concave edge as far as depth reasoning or casting of shadows is concerned but has the properties of an occluding edge as far as junction labellings are concerned. One solution to this dilemma is to introduce a new label $\rightarrow -$, as suggested in [37, 38]. Waltz [173] also introduced new labels for cracks, that is 3D lines along which two object edges meet.

3.10 Discussion

In this chapter we have presented constraints for the labelling of line drawings of polyhedral scenes. The resulting constraints are necessary but not sufficient conditions for physical realizability. The aim of the research presented in this chapter is to identify low-arity hard or soft constraints which can be applied before or during the search for an optimal interpretation. We consider such constraints to be essential for any practical line drawing interpretation system which is not restricted to objects with trihedral vertices. Indeed, based on junction constraints alone, we have seen that the number of legal labellings would be an exponential function of the size of the drawing.

We have chosen to restrict our attention to constraints on lines intersected by a path as well as constraints derived from the presence of parallel or collinear lines. Many other constraints exist. Examples include constraints derived from

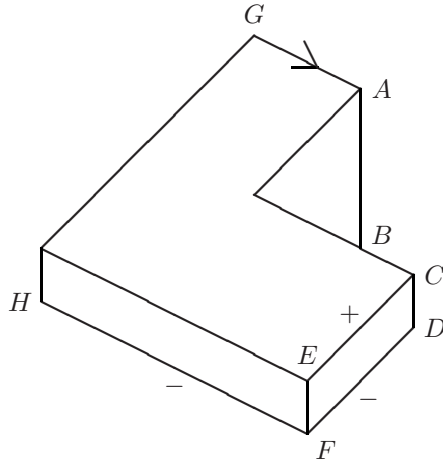


Figure 3.29: An example of a physically unrealizable labelled drawing of a polyhedral scene that is not detected by our constraints.

the presence of lines collinear with junctions [38], collinear lines [103], skew symmetry [132], vanishing points [127] and cubic corners [29, 166] (studied in depth in the following chapter). Furthermore, we have only studied low-arity constraints, and thus our constraints are necessarily incomplete even in the treatment of parallel and collinear lines. For example, the labelled drawing in Figure 3.29 is physically unrealizable, although it satisfies all the constraints given in this chapter. Physically unrealizability is a consequence of the following argument. From the labelling and the presence of parallel lines, we can deduce that $GACE$ and DFH lie on two parallel planes. But then, since AB is parallel to CD , it follows that B should not be visible since it must lie behind the plane DFH . We can thus deduce an arity-4 constraint based on the presence of three pairs of parallel lines. It is, of course, debatable whether constraints based on the presence of several pairs of parallel lines will be sufficiently robust to be worth applying in practical applications.

Chapter 4

Discrete Inflation Using Cubic Corners

Although a line drawing usually represents a whole family of three-dimensional objects, human beings manage to communicate using this highly ambiguous medium. We postulate that communication is possible due to the universality of criteria used to determine the most likely interpretation. In this chapter, we study in particular the preference for cubic corners in interpretations of line drawings of polyhedra.

We introduce a simple boolean labelling scheme for line drawings of polyhedra under orthographic projection. For each 3D edge AB , we assign a direction to the projection of AB to indicate whether point A or point B is nearer to the viewer. Although incomplete, such depth information is sufficient to automatically detect certain classes of impossible figures. This simple labelling scheme, combined with the maximization of the number of cubic corners, correctly finds the complete frontal geometry for many line drawings. We envisage a direct application of this scheme in software to automatically add 3D information to line drawings.

From an algorithmic point of view, the novelty of this approach is to perform optimization over finite domains which opens the possibility of a complete search for an optimal interpretation.

4.1 Computer-Enhanced Perception

Machine vision is often considered as an alternative to human vision. It is rarely considered as a means of *enhancing* human vision. We consider the interpretation of line drawings as an exemplary application area for computer-enhanced perception.

We present a line labelling scheme (which extends the traditional Huffman–Clowes labelling scheme) together with soft constraints expressing preferences for vertices at which three planar faces meet at right angles (known as cubic

corners). The resulting labelled line drawing often contains sufficient depth information for a computer to shade the surfaces of polyhedral objects according to their depth and/or orientation, or alternatively to add false depth contours, thus providing a pseudo-3D representation of the depicted object. This labelling scheme could therefore be used in an interactive drawing aid to automatically enhance drawings by adding depth information.

The depth-labelling scheme is based on the preponderance of parallel lines and cubic corners in man-made objects. Burns [17] emphasizes the importance of parallel lines and cubic corners in human visual processing. Varley and Martin [166] found parallel lines and cubic corners to be effective visual cues for determining the relative depth of adjacent vertices on experimental trials of their machine vision system on a sample of 479 semantically labelled line drawings.

4.2 Machine Interpretation of Line Drawings

Human interpretation of drawings of man-made objects is no doubt so effective because of the huge quantity of visual experiences which have been absorbed and summarized in the form of a multitude of unconscious rules and preferences. Machine vision systems, on the other hand, generally use a small number of rules, optimized mathematical algorithms and fast microprocessors to solve specific well-defined problems. Two classic approaches in the computer interpretation of line drawings of polyhedral scenes are

- The semantic labelling of lines (as occluding, convex or concave) subject to a catalogue of legal junction labellings [75, 20]
- The determination of the equations of the planar object faces, given a legal global labelling of the drawing, using linear programming [154, 155].

Semantic labels on their own provide an incomplete summary of the 3D scene depicted in a drawing, whereas precise equations of faces provide rather more information than necessary. In this chapter we are interested in sketches or illustrative diagrams, possibly created within a word-processing program, rather than technical drawings encountered, for example, in architecture or engineering. The latter are a formal specification of a 3D object, whereas the former do not necessarily obey the strict rules of physical realizability imposed by the linear programming approach. Furthermore, from a computational point of view, the linear programming approach admits generalizations in which we impose extra linear constraints (such as two parallel lines being projections of parallel lines in 3D, or a vertex being a cubic corner [166]) but does not adapt easily to allow us to express preferences between different physically possible interpretations.

We propose a knowledge-rich soft-constraint-based approach involving the introduction of extra discrete variables (instead of real-valued parameters) and extra soft constraints (representing preferences for more likely 3D shapes). Our approach is close to that of Ding and Young [57], who used a truth-maintenance system to apply both strict geometric constraints and heuristic rules based on

Gestalt principles (favouring interpretations involving parallelism, perpendicularity and symmetry) for complete object reconstruction from a single line drawing with missing lines.

Chapter 3 showed how constraints between distant lines complement the traditional junction-labelling catalogues. Without such constraints, drawings of objects with tetrahedral vertices [165] have, on average, an exponential number of legal global labellings. We extend these constraints by giving extra soft constraints based on cubic corners, parallel edges and planes, using new labels to represent edge orientation. This approach does not suffer from the superstrictness of the linear programming approach [73, 155] (in which a small error in the position of a junction can render a drawing unrealizable) and hence is no doubt closer to human visual processing.

The traditional machine vision approach to finding the most likely interpretation of a line drawing is to use some form of incomplete search to optimize a real-valued objective function. This objective function is the sum of compliance functions expressing preferences for likely object features such as right angles, symmetry, isometry, verticality, coplanarity, collinearity and equal angles [111, 147, 103, 166, 169]. We propose a finite-domain optimization formulation of the problem of determining the relative depth of vertices, which allows us to express both hard constraints (derived from geometrical reasoning) and soft constraints (expressing preferences between different interpretations). Over finite domains we can perform a complete search. Furthermore, we can take advantage of recently developed techniques for the simplification of soft constraint satisfaction problems [39, 41, 50, 98] (Chapter 8). These local reduction operations on soft constraints generalize well-known classical discrete relaxation algorithms [173, 55].

4.3 Depth Labels

Figure 3.24 gives the Huffman–Clowes catalogue of labelled junctions for projections of trihedral vertices. A global semantic labelling of a line drawing provides important local 3D shape information, since each line is labelled as occluding, convex or concave. This basic catalogue has been extended to allow for much wider classes of drawings, involving objects with tetrahedral vertices [165], lighting effects (shadows and contrast failure) [173, 38] and curved objects [109, 32, 34, 37]. In the case of curved objects and in the absence of any other constraints (such as straight lines, collinearity, skew symmetry, etc.), the junction catalogue provides a necessary and sufficient condition for realizability of a drawing and can be tested in linear time [36]. In Figure 3.24 we divide the set of possible labellings for each junction type into subtypes. The $W(+)$, $W(-)$, $Y(+)$ and $Y(-)$ subtypes were first introduced by Parodi and Torre [127]. Under perspective projection, knowledge of vanishing points of all lines allows us to identify all three-line junctions as $W(+)$, $W(-)$, $Y(+)$ or $Y(-)$, which again leads to an efficient labelling algorithm [127]. The two lines which meet at an L junction divide the drawing into four quadrants. The third hidden line can lie in

any of these quadrants: the $L_1(+)$, $L_1(-)$, $L_2(+)$, $L_2(-)$ subtypes correspond to these four cases.

However, it is possible to label lines with other information besides the semantic labels of the Huffman–Clowes catalogue (occluding, convex, concave). We propose a simple depth label: an arrow indicating the direction of increasing depth. Assuming orthographic projection and that no 3D edges are perpendicular to the line of sight, each line in the drawing can be assigned an increasing-depth direction. This will be indicated by a closed arrowhead, to distinguish it from the occluding label represented by an open arrowhead.

Some combinations of semantic and depth labellings are physically impossible. These are given in Figure 4.1(a). These strict constraints are extremely weak. There are no physically impossible depth labellings of $L_1(+)$, $L_1(-)$, $L_2(+)$ and $L_2(-)$ junctions. However, if we take into account the fact that many man-made objects have cubic corners (vertices at which three orthogonal planes meet), then we can impose a soft constraint translating a preference for projections of cubic corners. Figure 4.1(b) lists the possible labellings of $Y(+)$, $Y(-)$, $W(+)$, $W(-)$, $L_1(+)$, $L_1(-)$, $L_2(+)$ and $L_2(-)$ junctions which are projections of cubic corners. This is a very strong constraint, since in each case the depth labelling is unique. The set of trihedral vertices can be extended to include the vertices illustrated in Figure 4.2. Although there are more than three faces meeting at vertices A and B , these faces lie in only three planes. If these three planes meet orthogonally, then vertices A and B , together with the mirror image of A , provide three new kinds of cubic corners. Figure 4.3 lists the labellings which have to be added to our catalogue of projections of cubic corners. Note that the depth labelling (composed of the closed arrows) is uniquely determined by the shape of the junction (i.e. the junction type plus the identification of acute angles in the drawing).

Perkins [130] observed that only certain Y and W junctions can be projections of cubic corners (Figure 4.1). The most succinct statement of Perkins's rules is that when the three lines meeting at a Y or W junction J (a projection of a cubic corner) are extended through J , then no three of the resulting six half-lines lie within an angle of $\frac{\pi}{2}$ [103].

However, it is common to find anomalous combinations of angles in human-produced drawings of objects with cubic corners. For example, in many of the figures illustrating this chapter there are $Y(+)$ junctions representing a cubic corner in which one of the angles is a right angle, even though all angles should be obtuse. We propose to tolerate such errors in drawings by expanding the categories of acute and obtuse angles (in the catalogue of Figure 4.1) so that they both include right angles (or even all angles in a small interval $[\frac{\pi}{2} - \epsilon, \frac{\pi}{2} + \epsilon]$).

The two lines which meet at an L junction, when they are extended to infinity, divide the plane into four quadrants. The semantic labelling of the L junction determines in which quadrant the hidden line lies. Consider an L junction orientated as in Figure 4.1. The hidden line H lies in the right quadrant if the junction is of type $L_1(+)$, the left quadrant if it is of type $L_1(-)$, the lower quadrant if it is of type $L_2(+)$ and the upper quadrant if it is of type $L_2(-)$. In each case, the hidden edge slopes away from the viewer as it

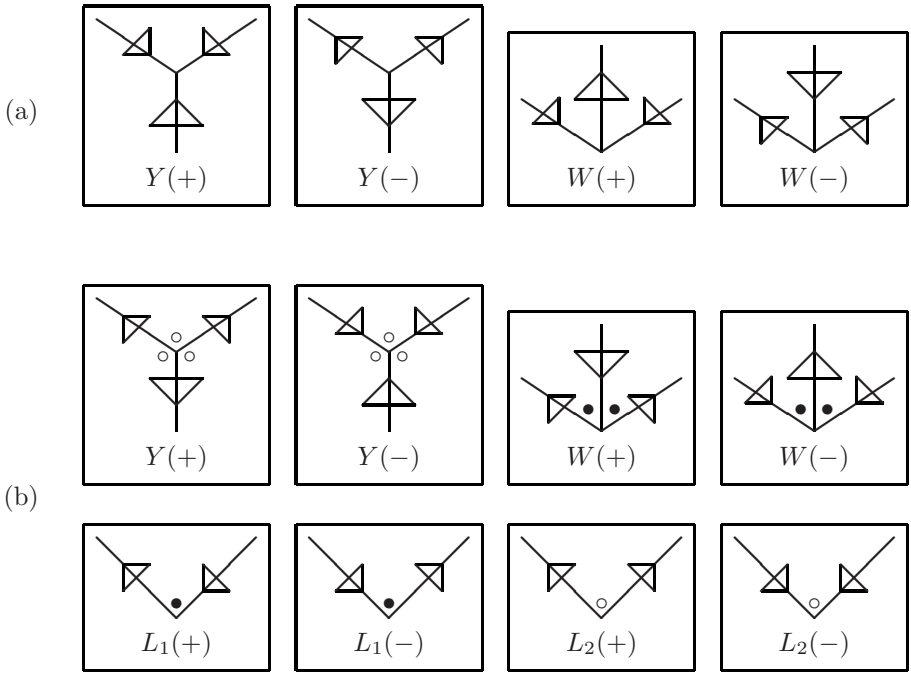


Figure 4.1: (a) Impossible depth labellings of projections of trihedral vertices. (b) Possible depth labellings of projections of cubic corners: angles marked \bullet are acute and angles marked \circ are obtuse; furthermore, the sum of the two acute angles at a W junction is an obtuse angle (Perkins's rules [130]).

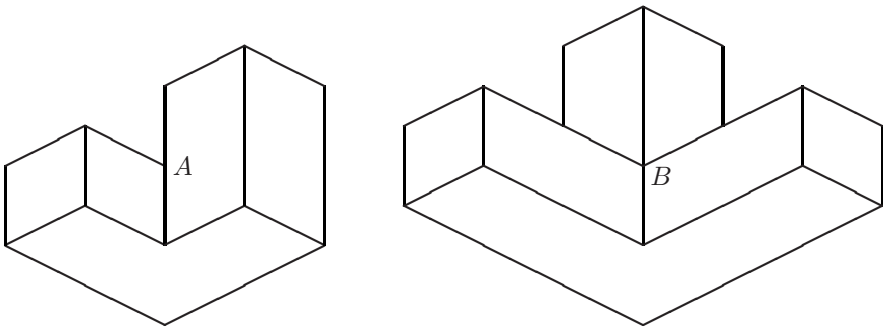


Figure 4.2: Extended trihedral vertices.

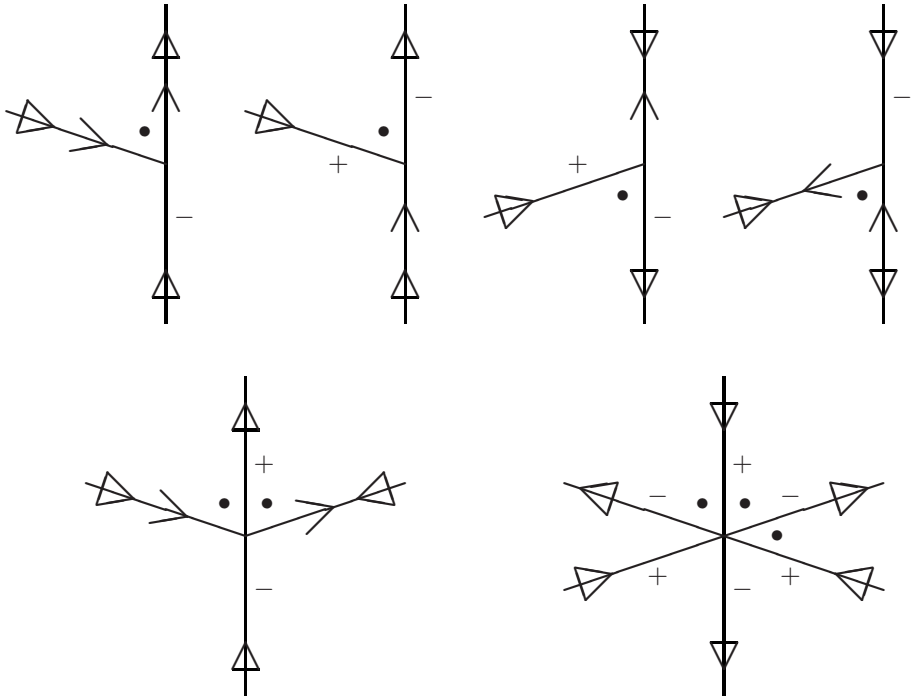


Figure 4.3: Labelings of cubic-corner extended trihedral vertices. Angles marked • are acute and the sum of the two acute angles at a Ψ junction (*bottom left*) is an obtuse angle.

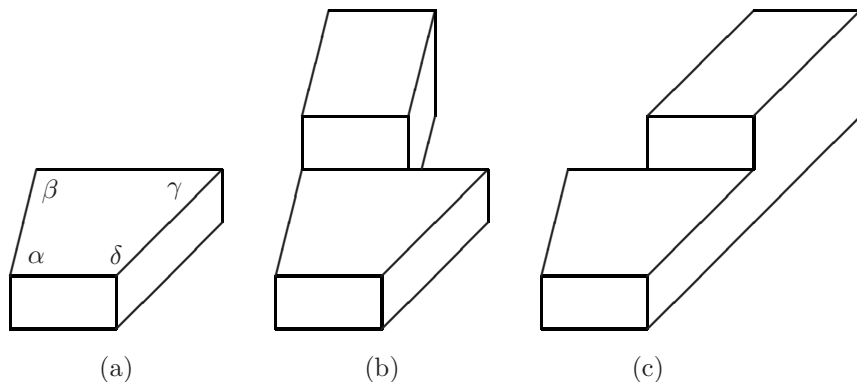


Figure 4.4: (a) An ambiguous figure: is it α and β which are right angles or γ and δ (or none)? (b) In this object, α and β are seen as right angles. (c) In this object, γ and δ are seen as right angles.

leaves the vertex, i.e. the depth label of H is an arrow pointing away from the junction. Furthermore, the hidden line H together with the two visible lines satisfy Perkins's criterion.

Figure 4.4(a) is the projection of an infinite class of 3D objects. For example, the 3D angle α could be any angle satisfying $\alpha < \pi$. Nonetheless, there are three preferred interpretations, one in which α and β are right angles, another in which γ and δ are right angles and a third in which $\alpha = \delta$ and $\beta = \gamma$. The drawings in Figure 4.4(b),(c) each have a different single preferred interpretation, which is consistent with the hypothesis that human interpretation maximizes the number of cubic corners in the reconstructed object.

4.4 Depth Labels and Impossible Figures

As possible evidence of the psychological relevance of depth labels in the study of human vision, consider the difference between the drawings in Figures 4.5 and 4.6. These are both examples of impossible figures which have legal semantic labellings. Assuming all vertices are cubic corners, Figure 4.5 has a legal global depth labelling whereas Figure 4.6 does not. Indeed, in Figure 4.6 every line has opposite depth labels assigned to it by the junctions at its two ends. When we observe Figure 4.5, we are aware of some form of physical impossibility, but we nevertheless interpret each line as being sloped towards or away from the viewer. This depth labelling is exactly the one found by applying the junction catalogues (Figures 4.1 and 4.3). Figure 4.7 is a similar example based on the Penrose triangle. Both drawings are examples of impossible figures which have legal semantic labellings. Assuming that all vertices are cubic corners, the drawing on the left has a consistent depth labelling, corresponding to the impression of 3D shape we have when we observe this drawing (while at the same time realizing that it is physically impossible). Again assuming that all

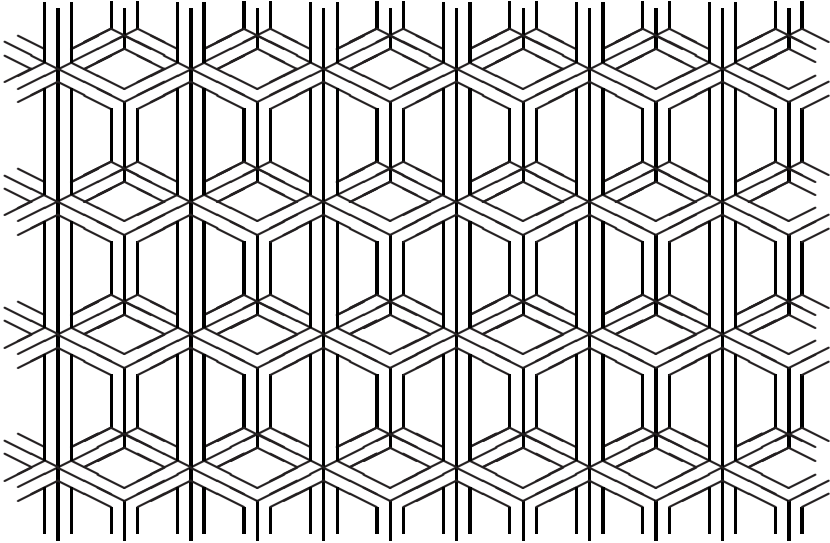


Figure 4.5: A tiling of the plane based on an impossible figure. A consistent depth labelling exists of the whole figure assuming each vertex is a cubic corner.

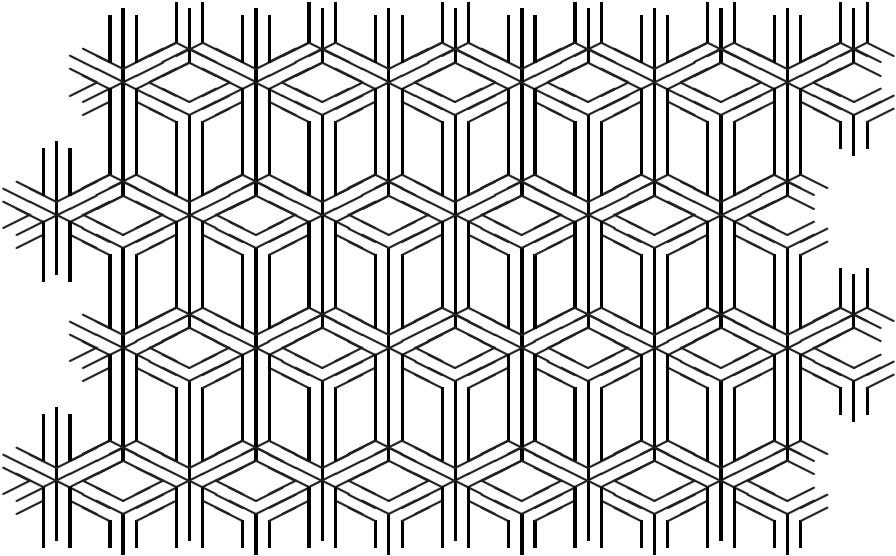


Figure 4.6: A tiling of the plane based on an impossible figure, in which every line has contradictory depth-label cues assuming all vertices are cubic corners.

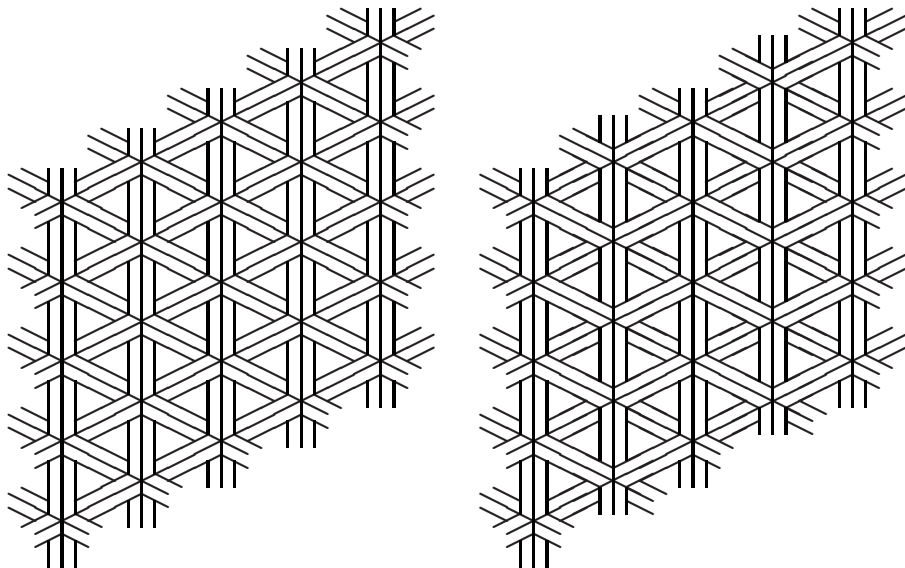


Figure 4.7: Tilings of the plane based on the Penrose triangle. The one on the left has a consistent depth labelling, whereas the one on the right does not. In both cases we assume all vertices are cubic corners.

vertices are cubic corners, the drawing on the right has no legal depth labelling, which corresponds well with our inability to interpret the non-vertical lines as sloping towards or away from the viewer.

As further examples of the use of our depth-labelling scheme to detect impossible figures, consider the drawings in Figures 4.8 and 4.9. In both cases, we assume cubic corners and produce a depth labelling. This then leads to a contradiction since there is an impossible depth cycle. In Figure 4.8(b), this is flagrant since the arrows shown in the figure indicate a cycle of points in 3D of increasing depth, which is clearly impossible. In Figure 4.9(b), one line has an arrow pointing in the opposite direction, but this decrease in depth is more than cancelled out by the longer parallel line whose arrow indicates increasing depth. Of course these impossible depth cycles only prove that there is no legal interpretation of the drawings involving objects with only cubic corners. Both drawings are physically realizable. Similar reasoning allows us to deduce that the drawings in Figures 2.2(c), 2.3(b), 2.4, 2.14(a), 2.8(a) and 2.8(b) are unrealizable as projections of objects with cubic corners. However, we emphasize that depth labelling followed by the detection of depth cycles does not detect all impossible figures (notably the drawings in Figures 2.9 and 2.13).

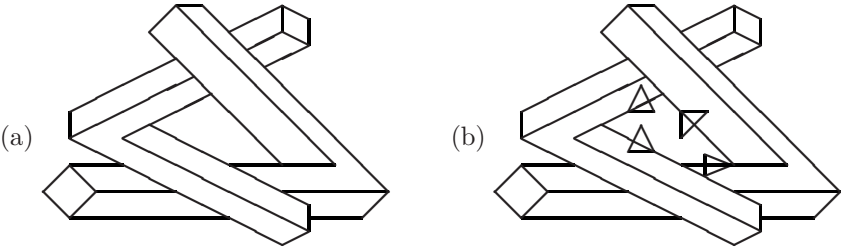


Figure 4.8: A figure which is impossible, assuming all vertices are cubic corners, due to the impossible depth cycle shown.

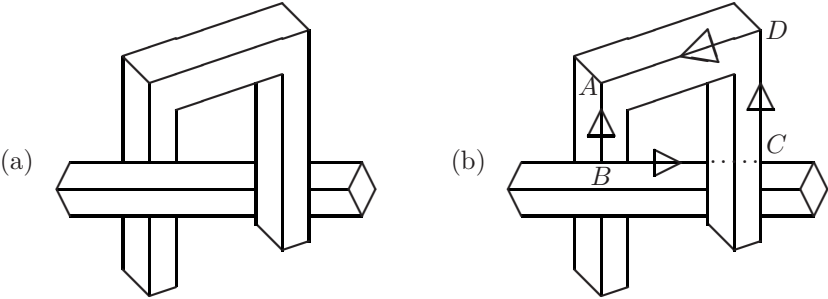


Figure 4.9: A drawing which is not realizable as a collection of objects with cubic corners due to the inconsistency of the depths of points along cycle $ABCD$.

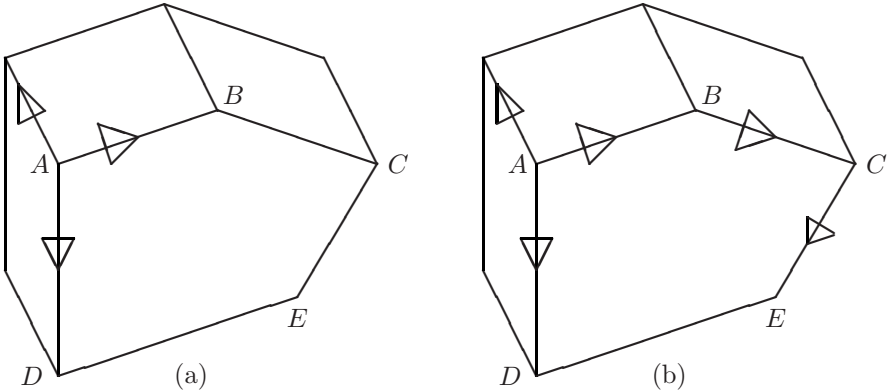


Figure 4.10: Example of the propagation of depth labels

4.5 Propagation of Depth Labels

An important property of depth labels is that they can propagate to other lines via one of the three following rules based on pairs of parallel edges, sets of coplanar edges or cubic corners:

1. Edges which are parallel in 3D have the same depth label.
2. If L_1, \dots, L_p are projections of a set of coplanar edges (lying in a plane P) and v_1, \dots, v_p are the 2D unit vectors created from these lines by orienting them according to their depth label, then there exists a 2D unit vector v (corresponding to the projection in the image plane of the normal to P) such that each L_i ($i = 1, \dots, p$) subtends an acute angle with v .
3. Let L_1, \dots, L_p, P and v_1, \dots, v_p be as above and suppose that L_0 (with corresponding unit vector v_0) is the projection of a line such that L_0, L_1, L_2 are the projections of the three edges meeting at a cubic corner. Then v (above) is given by $v = -v_0$.

As an example of the propagation of depth labels according to these three rules, consider the drawing in Figure 4.10(a). Junction A is a candidate for the projection of a cubic corner (according to Perkins's rules [130]), whereas B is not. The catalogue in Figure 4.1 gives the depth labelling shown, assuming that propagation of semantic labels has allowed us to identify A as a $Y(+)$ junction. The depth label of line BC shown in Figure 4.10 follows directly from rule 2 above. The depth label of CE follows from rule 3 above. (Note that if we were given the depth labels of AD and AB , without knowing that A was a cubic corner, then we could still apply rule 2 to deduce the depth label of BC , but the depth label of CE would be ambiguous.) The depth labels of all other lines in the drawing then follow from rule 1 above.

It is important to point out that the three propagation rules above only apply to drawings produced under orthographic projection. Under orthographic projection a straight line has the same depth label at both ends. However, under perspective projection this no longer holds. Consider the 3D line AB in Figure 4.11(a): assuming that A and B are both cubic corners, the nearest point to the viewer along line AB lies at the point, midway between A and B , at which the line of sight is perpendicular to AB . Consider a perspective projection with centre of projection O . Let L be a line in the drawing which is the projection of a 3D edge E and let P_E be the plane which passes through O and is normal to E . Denote by S the point of intersection of E with P_E and by T its projection in the drawing. We assign a distinct depth label to each point A on L as follows: the depth labelling at A is such that the arrow points away from point T . We call T the transition point of L . Under perspective projection, we have to modify our rules for propagating depth labels (via sets of parallel or coplanar lines) as follows:

1. If lines L_1, \dots, L_r are projections of parallel 3D edges E_1, \dots, E_r , then the transition points T_1, \dots, T_r of lines L_1, \dots, L_r are collinear. They lie

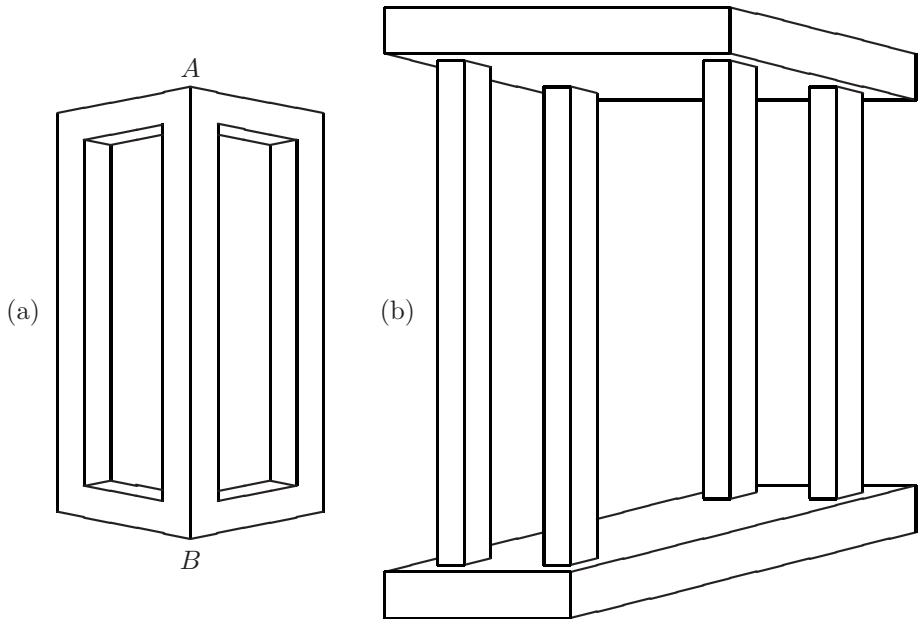


Figure 4.11: (a) A perspective projection; (b) a physically unrealizable drawing.

on a line which is the projection of the plane P which passes through the centre of projection O and is normal to all of the edges E_1, \dots, E_r .

2. Let line L be the projection of a 3D edge lying on a plane P , and let V be the vanishing point of 3D edges which are normal to P . Then the depth label at a point Q on L is such that L considered as a vector oriented by this depth-label subtends an acute angle with VQ .

Consider the drawing in Figure 4.11(b) (adapted from [61]). Assuming that all vertices are cubic corners allows us to deduce that all the lines in the drawing which are neither horizontal nor vertical are projections of parallel 3D lines and hence must meet at a vanishing point somewhere off to the right of the drawing. The catalogue of depth-labelled junctions (Figure 4.1) allows us to assign depth labels to all line ends. The fact that the two ends of each long vertical line have a different depth label is an important reason for considering this drawing as a perspective rather than an orthographic projection. The above propagation rules allow us to propagate these labels to every point of every line. Physical impossibility does not follow directly from the depth labelling but rather from recognition that the subdrawing consisting of any two columns, together with the base and the roof, form a simple example of an impossible object from the impossible closed curve class (described in Section 2.1).

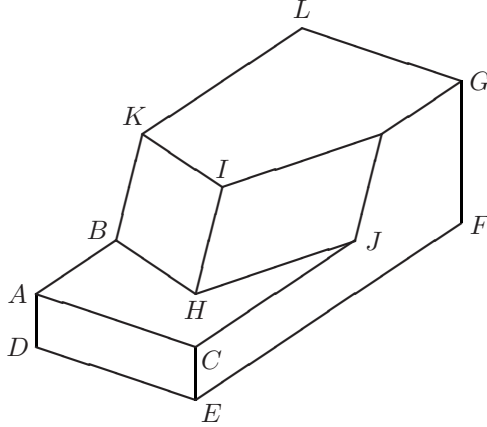


Figure 4.12: Examples of the application of the orthogonality rules.

4.6 Orthogonality Constraints on Cubic Corners

To identify cubic corners, we applied the simple criterion that the interpretation involving the largest number of cubic corners was the most likely one. The following orthogonality rules were also applied to avoid obvious geometric inconsistencies. $L_1 \perp L_2$ means that L_1, L_2 are orthogonal; $L_1 // L_2$ means that L_1, L_2 are parallel; *cubic*(V) means that vertex V is a cubic corner. In these orthogonality rules, the 3D lines L_1, L_2, L_3 meet at vertex V and the 3D lines L'_1, L'_2, L'_3 meet at V' .

1. By definition, *cubic*(V) if and only if $L_1 \perp L_2, L_2 \perp L_3$ and $L_3 \perp L_1$.
2. If $L_1 // L'_1, L_2 // L'_2$ and $L_1 \perp L_2$, then $L'_1 \perp L'_2$.
3. If the plane of L_1, L_2 is parallel to the plane of $L'_1, L'_2, L_1 \perp L_2, L_1 // L'_1$, but L_2 is not parallel to L'_2 , then L'_1 is not orthogonal to L'_2 .
4. If the plane of L_1, L_2 is parallel to the plane of L'_1, L'_2 and *cubic*(V), but L_3 is not parallel to L'_3 , then V' is not a cubic corner.

For example, in Figure 4.12, suppose that A is a cubic corner and that a semantic labelling of the drawing has allowed us to deduce that A, B, H, J, C are coplanar [37]. Rule 1 implies that AB, AC, AD are orthogonal. Then three applications of rule 2 tells us that C is also a cubic corner. We can also deduce from rule 2, that $GF \perp FE$, which means that F is a valid candidate for a cubic corner. However, rule 3 tells us that B cannot be a cubic corner, since BH is not parallel to AC . Knowing that A, B, H, J, C are coplanar, we can apply rule 4 to deduce that H is not a cubic corner, since plane BAC is parallel to (in fact coincident with) plane BHJ , but HI is not parallel to AD .

As another example, consider the drawing in Figure 4.13(a). The drawing has the unique trihedral semantic labelling shown in Figure 4.13(b). Suppose that vertex A is a cubic corner. Then lines AB, AD, AE have the depth

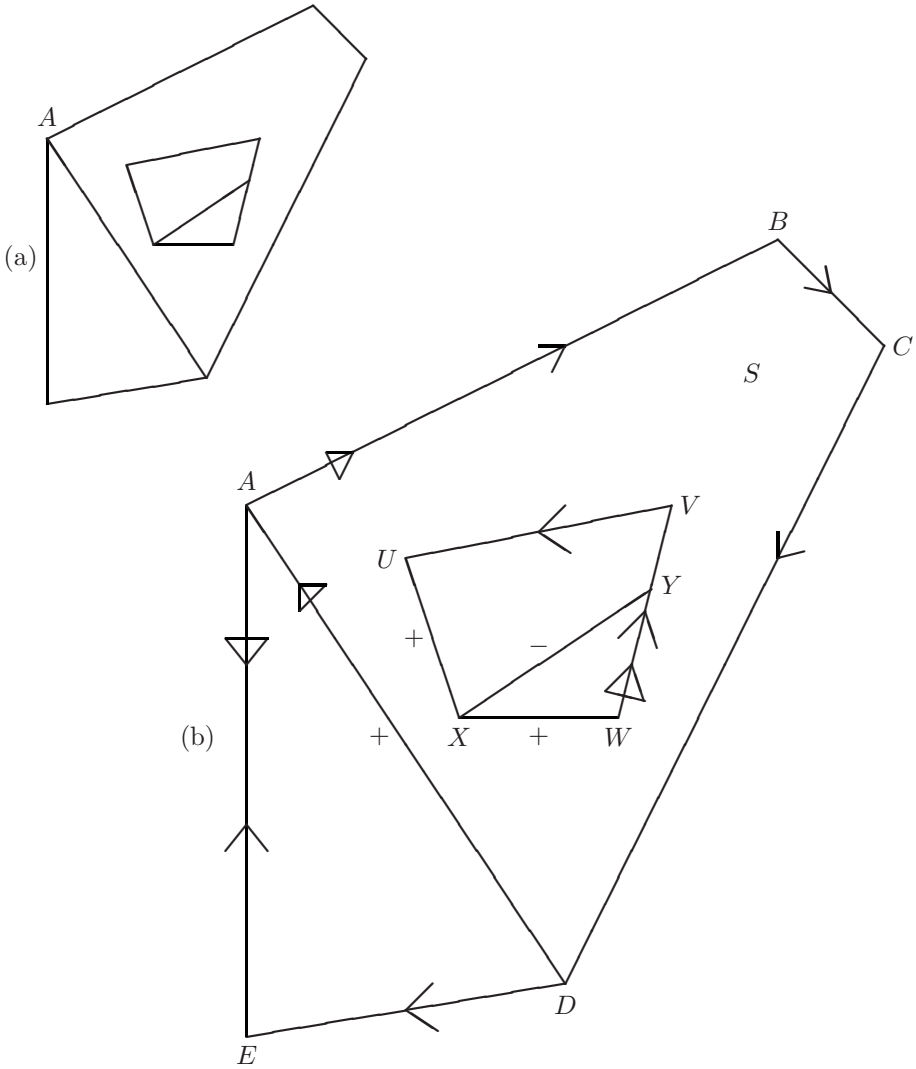


Figure 4.13: Examples of the combination of different rules.

labels shown. From the semantic labelling we can deduce that line WV lies on face S . The depth label of WV , shown in Figure 4.13(b), follows from our propagation rules (since we have the direction of the normal AE to S). The semantic labelling tells us that the projection of W is an $L_1(+)$ junction, but then the depth-labelling catalogue tells us that W cannot be a cubic corner (since the depth label of WV points in the wrong direction). Vertex X cannot be a cubic corner either, since UX, XW lie on the surface S but XY cannot be normal to S since it is not parallel to AE . Vertex B cannot be a cubic corner since BC is not parallel to AD . Similarly, D cannot be a cubic corner since DE is not parallel to AE . Vertices E, U, V cannot be cubic corners since they do not satisfy Perkins's rule.

Vertex C is an interesting case since it involves reasoning about a hidden edge. Any hidden third edge terminating at C is necessarily occluded by the surface S and hence cannot be parallel to AE ; this shows that C cannot be a cubic corner. We can reason not only about hidden lines but also about hidden planes. For example, assuming that vertices A, B, K in Figure 4.12 are trihedral allows us to deduce that D, A, B, K, L are coplanar. (Such coplanarity constraints are described in detail in Chapter 6, where it is shown that they also apply to curved objects with some straight edges.) It then follows that, if A in Figure 4.12 is a cubic corner, then K cannot be a cubic corner since KI is not parallel to AC .

In the fourth orthogonality rule, above, for lines L_3, L'_3 to be parallel, they must satisfy three criteria: their 2D projections must be parallel, these projections must have identical depth labels and L_3, L'_3 must have the same inclination to the image plane. For the calculation of this inclination angle, consider a cubic corner V at which lines AV, BV, CV meet in 3D. Suppose, for concreteness, that V projects into a Y junction and let A', B', C', V' denote the projections of A, B, C, V on the image plane. Let β be the angle $A'V'C'$ and γ the angle $A'V'B'$. Then it is well known [130, 89, 166] that the angle of inclination θ of AV to the image plane is given by

$$\tan^2 \theta = \tan \beta \tan \gamma - 1. \quad (4.1)$$

The sign of θ is given by the depth label assigned to $A'V'$. Figure 4.14 shows an example in which the two vertices A and B cannot simultaneously be interpreted as cubic corners, since the above calculation leads to different inclination angles for the 3D edges AC and BD (and hence for surface S). On the other hand, vertices E and F could simultaneously be cubic corners. Applying the depth-label catalogue, the propagation rules and the orthogonality rules means that, for each 3D face S , we have at most one 3D orientation for S (corresponding to the unit 3D vector normal to S). It is possible that these orientations may be mutually inconsistent, since we are basically using only a simplified version of gradient space constraints [107]. Our aim is to help a human user by approximating human vision rather than to detect all physically impossible drawings.

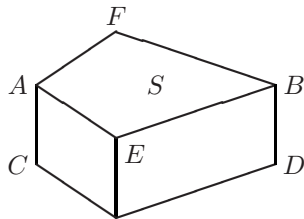


Figure 4.14: Vertices A and B cannot both be cubic corners since this would imply different inclinations of lines AC and BD .

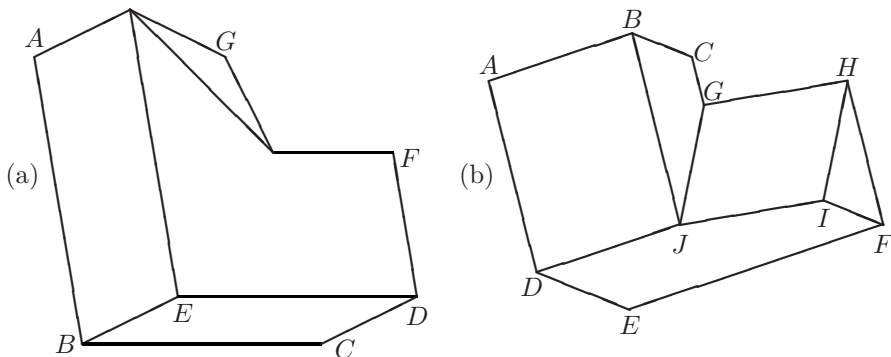


Figure 4.15: Examples of drawings of objects with some cubic corners (adapted from Varley and Martin [165])

4.7 Experimental Trials

We used a sample of 103 line drawings of polyhedral objects under orthographic projection to test the validity of the heuristic which maximizes the number of cubic corners in an interpretation (subject to the orthogonality rules, above). These drawings are the set of figures in Varley and Martin's paper [165] after elimination of repeated or mirror-image drawings. These figures were created for an entirely different purpose, namely the presentation of a catalogue of labellings of projections of tetrahedral vertices. The algorithm we applied to each drawing was:

1. Find a semantic labelling of the drawing using Varley and Martin's tetrahedral catalogue and eliminating ambiguity by applying labelling constraints based on the presence of parallel lines (described in detail in Chapter 3).
2. Maximize the number of cubic corners subject to Perkins's rule, the depth-labelling catalogue of Figure 4.1 and the orthogonality rules above.

In 92 of the 103 drawings, all visible vertices were correctly labelled as cubic or non-cubic. In 5 of the 103 drawings, a single vertex was spuriously identified

as cubic. Vertex G in Figure 4.15(a) is a typical example: it was not intended by Varley and Martin to represent a cubic corner, but this conclusion is not geometrically incoherent. On the remaining 6 of the 103 drawings our algorithm failed miserably. To explain this, consider the simplified examples shown in Figure 4.16. Our algorithm blindly decides that $A, B, C, D, E, F, G, H, I$ are all cubic corners. It is clear that for certain drawings, other heuristics are important, such as symmetry, equality of angles, verticality and isometry [103].

It is worthwhile looking at an example in some detail, namely Figure 4.15(b). Maximizing the number of cubic corners tells us that A, B, C, D, E, F are cubic corners. (For example, if, as an alternative hypothesis, we suppose that I is a cubic corner, then applying the orthogonality rules allows us to deduce that none of B, C, D, E, F, G, H can be cubic corners.) Applying the depth-labelling catalogue and propagating allows us to assign a depth label to all lines.

The 97 drawings for which our algorithm provided a plausible interpretation contain a total of 424 visible faces. The 3D orientation of 343 of these faces was determined directly by applying Equation (4.1). For 62 other faces we were able to determine orientation by applying two very simple geometrical rules:

1. If lines L_1, L_2 lie on face S , lines L'_1, L'_2 lie on face S' , $L_1 // L'_1$, $L_2 // L'_2$ and L_1 is not parallel to L_2 , then faces S, S' are parallel and hence have the same orientation.
2. The orientation of a face S is determined once the depths of three non-collinear points lying on S are known.

As an example of the use of rule 2 above, consider the three faces which meet at I in Figure 4.15(b). Knowing that F is a cubic corner allows us to calculate the orientation of faces HIF and $DJIFE$. Knowing the orientation of the two other faces meeting at vertex I uniquely determines the orientation of the third face $GHIJ$. As pointed out by Draper [59], this method is theoretically incomplete since for some drawings a 3D model exists but cannot be constructed incrementally. For more complex drawings, we could apply the full power of the linear programming approach [152, 154, 155], described in Chapter 6, to calculate those face orientations which are uniquely determined by known face orientations. In an alternative approach, Li [101] uses vectorial equations to parameterize the set of all possible solutions. Methods for avoiding superstrictness include shifting the positions of a minimal subset of junctions [152, 155], finding an optimal simultaneous modification to all junction positions [73] or simply allowing a tolerance in all the equations and inequalities [37].

Applying the two simple geometric rules above left only 19 faces (out of 424) whose orientation remained ambiguous. We estimate that the orientation of 8 of these faces would be unambiguous to most human viewers due to reasoning about hidden surfaces, symmetry and collinearity, whereas the orientation of 11 faces would appear ambiguous to many human viewers. The triangular face in Figure 4.15(a) is a typical example that we placed in the ‘ambiguous’ category.

An important point to make is that, if we do not identify vertices as cubic corners, then the orientations of *all* of the 424 faces are ambiguous. For example,

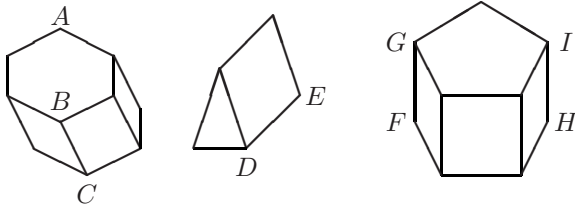


Figure 4.16: Drawings for which human beings prefer interpretations involving symmetry, equal angles and isometry rather than cubic corners.

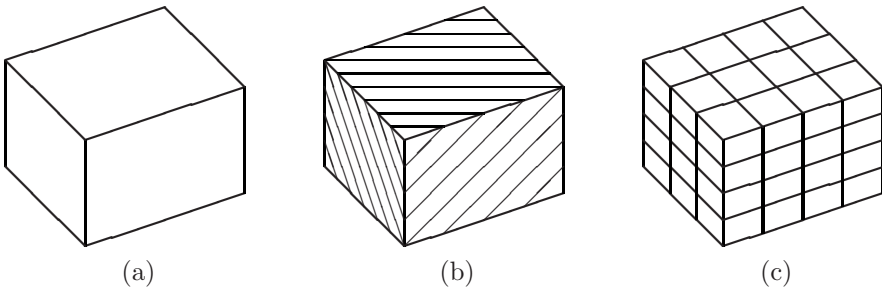


Figure 4.17: (a) A line drawing; (b) with depth contours added; (c) with brick structure added.

the drawing in Figure 4.15(b) could theoretically represent any of an infinite family of objects obtained by squeezing or stretching the object in depth.

4.8 Adding Depth Information to Line Drawings

Two ways of representing surface orientation are shown in Figure 4.17. Under an assumption that the central Y junction of the drawing in Figure 4.17(a) is a cubic corner, we can uniquely determine the 3D orientations of the three visible surfaces using Equation (4.1). These orientations can then be made more explicit in the drawing by adding extra information such as depth contours (Figure 4.17(b)) or surface texture (Figure 4.17(c)). Consider a cubic corner projecting into a Y junction at which lines L_1 , L_2 , L_3 meet. Within the region bounded by lines L_1 , L_2 , the depth contour is perpendicular to L_3 and the perpendicular distance between two consecutive depth contours at depths d and $d + 1$ is $1/\tan\theta$, where $\tan\theta$ is given by Equation (4.1).

Another approach (which could even be combined with depth contours or surface texture) is to shade the faces using some standard computer graphics rendering algorithm. Yet another approach is to use false colours with, for example, more distant surface points being more blue. Elber [60] has demonstrated

that depth information can be displayed in the original line drawing itself by making the width or darkness of lines inversely proportional to 3D depth. A more artistic approach is to make line thickness proportional to surface orientation and to explicitly insert gaps in contours to exaggerate the brightness of contour edges [51].

Whereas many workers have addressed the problem of 3D-object reconstruction from a single line drawing, in this chapter we have a more modest aim of image enhancement by addition of possibly incomplete depth information. We do not encounter the same superstrictness problems as in 3D-object reconstruction. For example, the drawing in Figure 2.2(b) can be analysed without having first to automatically correct the original line drawing [152].

4.9 Vertices Which Are Not Cubic Corners

A trihedral vertex may have some intrinsic structure, even when it is not a cubic corner, which renders it more likely to occur than a random vertex. At a cubic corner, three edges meet at right angles. We can relax this condition in various ways to produce different classes of structured vertices. Consider a vertex V formed by three convex edges E_1, E_2, E_3 . Let F_i denote the face opposite (i.e. which is not adjacent to) edge E_i . Let α_i represent the angle between the two edges lying on face F_i , and let δ_i represent the dihedral angle between the faces which intersect along E_i . We consider that the most likely forms of intrinsic structure that a vertex of a man-made object can possess are given by $\alpha_i = \frac{\pi}{2}$, $\delta_i = \frac{\pi}{2}$, $\alpha_i = \alpha_j$ (for some $i \neq j$) or $\delta_i = \delta_j$ (for some $i \neq j$). At a cubic corner, all such equalities hold. A cubic corner has no degrees of freedom, since $\alpha_1 = \alpha_2 = \alpha_3 = \frac{\pi}{2}$, whereas a general vertex has three degrees of freedom, since $\alpha_1, \alpha_2, \alpha_3$ are arbitrary. In this section we consider trihedral vertices which have one degree of freedom.

Let P_{ij} be a plane bisecting edges E_i, E_j and normal to the plane of these two edges. Then $(\alpha_i = \alpha_j) \Leftrightarrow (P_{ij} \text{ is a plane of symmetry of vertex } V) \Leftrightarrow (\delta_i = \delta_j)$. Furthermore, if i, j, k are all distinct, then $(\alpha_i = \alpha_j = \frac{\pi}{2}) \Leftrightarrow (E_k \text{ is normal to face } F_k) \Leftrightarrow (\delta_i = \delta_j = \frac{\pi}{2})$. From these observations it is easy to deduce that there are four distinct classes of trihedral vertices with exactly one degree of freedom:

Oblique rectangular corner: $\alpha_i = \alpha_j = \frac{\pi}{2}$ (and hence $\delta_i = \delta_j = \frac{\pi}{2}$);

Symmetric vertex with right angle: $\alpha = \frac{\pi}{2}$ and $\alpha_j = \alpha_k$ (and hence $\delta_j = \delta_k$);

Symmetric vertex with dihedral right-angle: $\delta_i = \frac{\pi}{2}$ and $\alpha_j = \alpha_k$ (and hence $\delta_j = \delta_k$);

Equiangular vertex: $\alpha_i = \alpha_j = \alpha_k$ (and hence $\delta_j = \delta_k = \delta_i$).

More than half of the examples of trihedral vertices which are not cubic corners in the 172 figures of [165] are oblique rectangular corners. We therefore

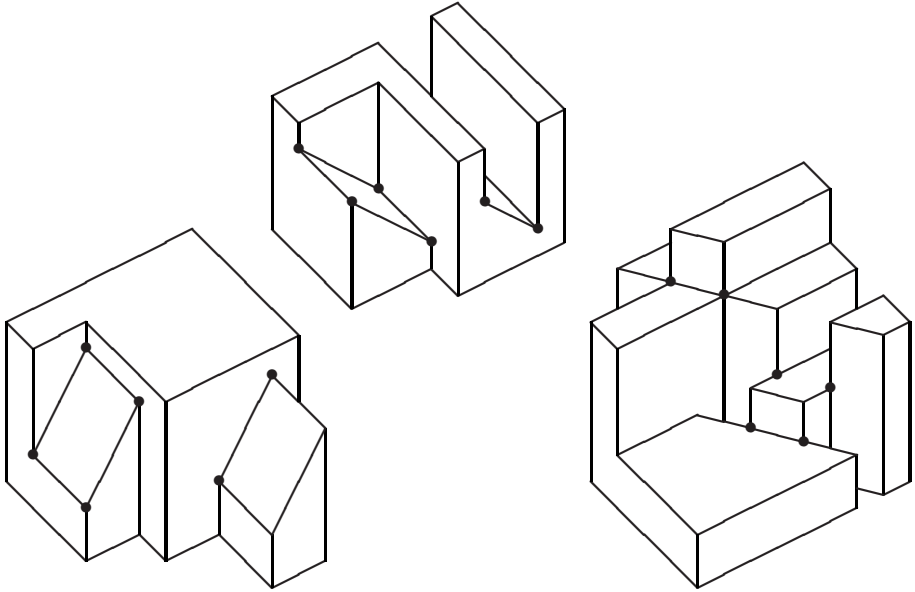


Figure 4.18: Examples of the 18 distinct types of oblique rectangular corners, marked by a \bullet .

investigated the depth labellings of oblique rectangular corners. Figure 4.18 shows the different types of oblique rectangular corners. For each of these 18 vertex-types, we divided 3D space into sectors separated by the planes of the three faces F_i, F_j, F_k together with the two planes N_i, N_j normal to edges E_i, E_j (respectively) and passing through the vertex. Note that plane N_k coincides with the plane of F_k by definition of an oblique rectangular corner. The labelling of the projection of a vertex is identical for all viewpoints lying within the same sector. By exhaustive search we established a catalogue of labelled junctions which are projections of oblique rectangular corners. The most interesting result, which we could, in fact, have obtained directly by simple geometric arguments, is that for $Y(+), Y(-), W(+), W(-)$ junctions whose angles satisfy Perkins's conditions (see caption of Figure 4.1) and for T, Ψ and X junctions with semantic labels and acute angles as shown in Figure 4.3, the only depth labellings of projections of oblique rectangular corners are exactly the same as for cubic corners.

This study of oblique rectangular corners demonstrates the robustness of our depth-labelling catalogue (except for the labelling of L junctions) when the definition of a cubic corner is relaxed so that one of the angles $\alpha_k \neq \frac{\pi}{2}$. However, it should be noted that this result does not hold for symmetric vertices (with right angle or dihedral right angle) or for equiangular vertices.

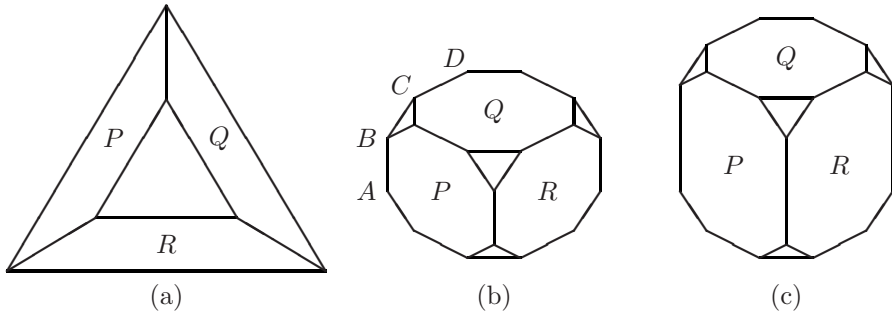


Figure 4.19: Objects which contain no cubic corners, but for which a satisfactory inflation exists by assuming that faces P , Q , R lie in orthogonal planes.

4.10 Discussion

Any drawing has a valid 3D interpretation as the projection of a flat scene in which all its visible lines and vertices are coplanar. A drawing whose most likely interpretation is not flat (i.e. not all lines and vertices are coplanar) is said to *inflate*. It suffices that a single Y or W be interpreted as a cubic corner, for the drawing to inflate. However, many objects do not contain cubic corners. Figure 4.19 shows three examples of such objects. There is no possible interpretation of Figure 4.19(a) containing cubic corners because all six junctions fail Perkins's criterion. A highly implausible, but physically possible, interpretation of Figure 4.19(b) exists in which one Y junction and two L junctions are cubic corners.

It is possible to extend the set of objects we can correctly inflate by looking for *virtual* cubic corners, such as the intersection of the planes of faces P , Q , R in Figure 4.19(b). If three faces P , Q , R pairwise mutually intersect along edges which are not parallel, then the planes of faces P , Q , R meet at a point in 3D space. Let E_{PQ} denote the 3D edge between faces P and Q and let L_{PQ} denote the projection in the drawing of E_{PQ} (with L_{QR} , L_{RP} defined similarly). Extending lines L_{PQ} , L_{QR} , L_{RP} to infinity in both directions produces six half-lines for each of which we can calculate a slope. If three of these slopes are spanned by an angle less than or equal to $\frac{\pi}{2}$, then it is not possible that faces P , Q , R lie on orthogonal planes. This is a simple extension of Perkins's criterion to virtual cubic corners.

Since a virtual cubic corner has the same geometric properties as a real cubic corner, we can apply the depth-labelling catalogue of Figure 4.1 to any virtual cubic corner. Due to digitization errors, it is unlikely that the extensions of L_{PQ} , L_{QR} , L_{RP} will actually meet at a point. We have therefore phrased Perkins's criterion, above, so that it can be tested even if L_{PQ} , L_{QR} , L_{RP} do not meet at a point. Furthermore, Perkins's criterion is applied with a tolerance of ϵ , as described in Section 4.3.

It is possible to detect virtual cubic corners even when one of the faces is

occluded. If we assume that vertices B and C in Figure 4.19(b) are trihedral, then coplanarity constraints [37] tell us that vertices A, B, C, D are coplanar since they lie on the same face. Denote this face by S . Then P, Q, S form a virtual W junction, which can be the projection of a virtual cubic corner. Thus, we can detect four mutually consistent virtual cubic corners in Figure 4.19(b).

We believe that it is important to detect basic 3D structural properties, such as coplanarity, parallel planes, orthogonality and symmetry, before trying to assign exact depths to all vertices. Consider again the simple drawing in Figure 4.4(a). By actively searching for structural properties we can obtain a small number of hypotheses, including $\alpha = \frac{\pi}{2}$, $\alpha = \delta$ and $\delta = \frac{\pi}{2}$, which need to be investigated further. This is indeed the approach of Ding and Young [57]. Optimization over continuous domains (such as the depths of vertices), on the other hand, excludes the possibility of complete search. This means that the objective function must not only be constructed to have its global minima corresponding to the most likely interpretations but must also be well behaved in the rest of the search space (absence of local minima and no large plateaux in between these minima).

Marill [111] suggests minimizing the standard deviation σ of 3D angles between adjacent edges in order to inflate drawings. However, this often gives anomalous results. For example, in Figure 4.19, σ is minimized as all the 3D angles in faces P, Q, R approach $\frac{\pi}{2}$, i.e. as the height of the truncated pyramid approaches infinity. Minimizing the standard deviation of segment lengths [15] provides a satisfactory interpretation of Figure 4.19(a), but it produces other anomalies. For example, the 3D angle θ between faces P and R will be found to be smaller in Figure 4.19(b) than in Figure 4.19(c). The choice of the best objective function for the inflation of drawings of objects with no cubic corners or virtual corners remains for the moment an open question.

4.11 Conclusion

We have introduced a new labelling scheme for line drawings. Labels indicating the direction of increasing depth of the corresponding 3D line provide a partial depth ordering of object vertices. This labelling scheme, based on human preference for interpretations involving cubic corners, allowed us to explain why certain drawings appear impossible. It is also an essential ingredient in our algorithm to add depth information to line drawings. We believe that human vision uses an even richer discrete labelling scheme involving, for example, the identification of coplanar points, pairs of parallel surfaces, collinear points, equidistant pairs of points, equal angles, lines of symmetry, etc. Our simple optimization criteria, namely minimizing the number of tetrahedral vertices [43] and maximizing the number of cubic corners, need to be extended to account for human preference for other common properties of man-made objects such as symmetry, isometry, right angles, parallel planes, etc. Varley et al. [169] list 16 different compliance functions used by various machine vision systems employing numerical optimization.

The experiments reported in this chapter demonstrate the feasibility of extending the traditional line labelling problem [20, 75] in three ways:

- Allowing non-trihedral vertices,
- The introduction of new discrete variables (depth labels for lines and the labelling of junctions as projections of cubic corners),
- The introduction of an optimization criterion (the maximization of the number of cubic corners).

This (or a similar) labelling scheme may prove useful in various applications, including the 3D reconstruction of an object from a wireframe projection (Chapter 7), a sketch [104, 27] or a drawing with missing lines [38, 57].

Chapter 5

A Rich Labeling Scheme for Curved Objects

5.1 Labeling Line Drawings of Curved Objects

Early work on line drawing interpretation made the very restrictive assumption that all objects were polyhedra. Malik [109] was the first to draw up a catalogue of legally labelled junctions for a formally defined class of curved objects by assuming that objects have smooth C^3 surfaces separated by edges representing a discontinuity of the surface normal. No edge or surface is tangential to another edge or surface. As in the polyhedral case, the drawing is assumed to be a perfect projection of a manifold object (a 3-manifold bounded by a 2-manifold) from a general viewpoint. This is the basic set of assumptions we make concerning line drawings of curved objects, although we will discuss how to cope with relaxations of these assumptions using valued constraints.

A new type of line occurs in drawings of curved objects since a curved surface may occlude itself. Consider, for example, the visible boundary of a sphere. The locus of points at which the line of sight is tangential to the object surface is called an *extremal edge*, a fold, a virtual edge or a phantom edge since its 3D position varies with changes in the viewpoint. Its label is a double-headed arrow, but for typographical reasons we use \Rightarrow to represent this extremal label in the text of the book.

When surfaces are curved, the semantic label of a line may change at any point at which L intersects a hidden extremal line. Such transitions from occluding to convex labels, such as point P in Figure 5.2(a), are common in line drawings of curved objects. Other transitions may occur. For example, as pointed out in [32], a convex-concave transition can occur if distinct surfaces are tangential to each other. Under the straight-edge formation assumption, which says that a straight edge is formed by the intersection of two locally planar surfaces [37], the semantic label of a line L is invariant along any straight segment of L . In particular, a straight line has a unique semantic label.

Under our basic assumptions, there are only four ways that lines can project into junctions in a drawing [131]:

- Three or more surfaces meet at a 3D vertex which is topologically equivalent to a polyhedral vertex.
- An edge occludes another (projecting into a T junction).
- A surface intersects a self-occluding curved surface to form a semi-fold point (projecting into a C, curvature-L or a 3-tangent junction depending on the viewpoint).
- A curved surface smooths out so that it no longer occludes itself (projecting into a terminal junction).

Figure 5.1 gives the list of labelled junctions that must be added to the catalogue of labelled junctions for polyhedral objects (such as Figure 3.2 if we restrict ourselves to trihedral vertices). At a terminal junction, a line L terminates at a point P . Under our assumptions, L is necessarily the projection of an extremal edge and L forms a cusp with a hidden extremal line. Terminal junctions have been studied in detail by Koenderink and van Doorn [94, 93] using differential geometry and singularity theory.

Both curvature-L and 3-tangent junctions have reflected versions whose legal labellings can easily be obtained by reflection of the labellings given in Figure 5.1. A black dot on a line represents a discontinuity of curvature. At a 3-tangent junction there is continuity of curvature between the lines labelled $+$ and \leftarrow (since they are projections of the same edge) but a discontinuity of curvature between the extremal line (labelled \Leftarrow) and the surface-normal discontinuity lines (labelled $+$ and \leftarrow). At a C junction there is no discontinuity of curvature. This renders the junction invisible; the C junction is often known as a phantom junction. The presence of a discontinuity of curvature at curvature-L and 3-tangent junctions was proved formally by Nalwa [119]. At a curvature-L junction the continuation of the surface-normal discontinuity line (labelled $-$ or \leftarrow) must lie to the right of the extremal line as we follow this extremal line in the direction of its arrows, since the corresponding edge lies on the hidden part of the self-occluding curved surface. Although physically possible [32], the two curvature-L labellings on the right-hand side of Figure 5.1 are much less likely than the two labellings on the left-hand side. Similarly, the two lower labellings of a C junction in Figure 5.1 are much less likely than the three upper labellings [32]. In the valued constraint framework, we can apply a high cost to such unlikely labellings. Furthermore, we can minimize the number of label transitions by assigning a non-zero cost to the first two labellings of C junctions in Figure 5.1.

When two straight-line segments are collinear, they do not necessarily have the same semantic label. However, under the general viewpoint assumption (GVA), a viewpoint-dependent edge (\Leftarrow , \Rightarrow) should not be collinear with a viewpoint-independent edge ($+$, $-$, \leftarrow , \rightarrow). Furthermore, edges labelled \Leftarrow and \Rightarrow should not be collinear [37].

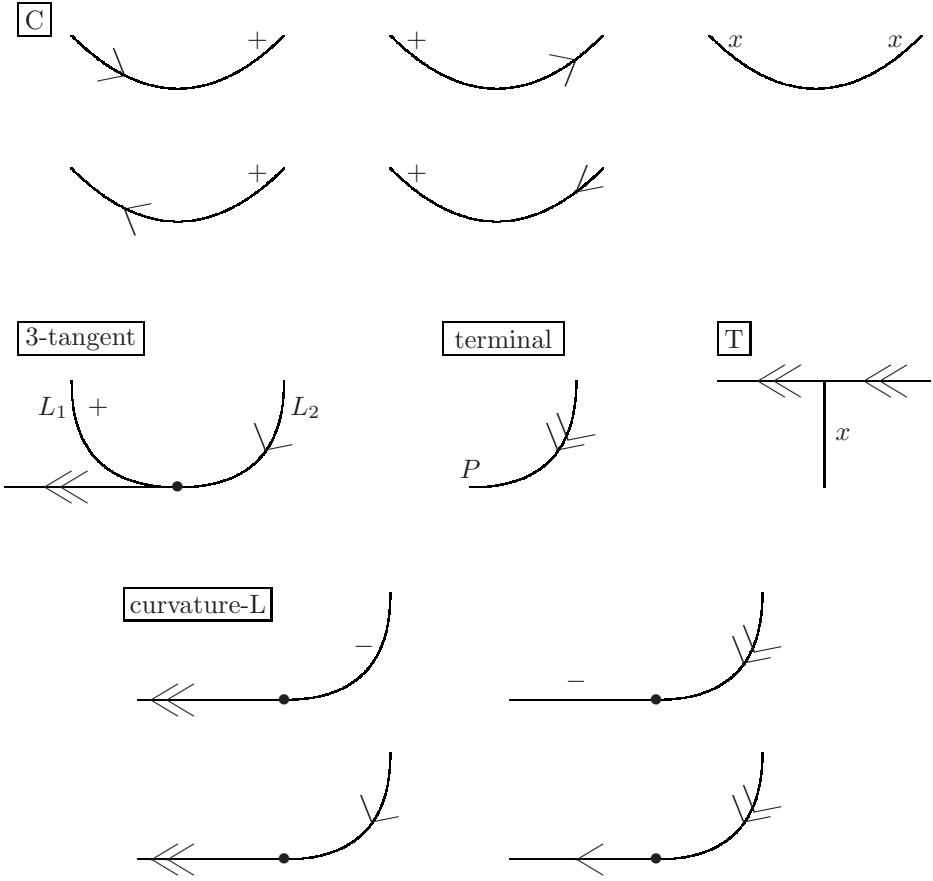


Figure 5.1: The catalogue of labellings of viewpoint-dependent junctions in drawings of objects with piecewise C^3 surfaces (to be added to the labellings of viewpoint-independent junctions). x represents any of the six labels.

We can extend our catalogue to cover objects whose edges and surfaces may meet tangentially [34]. Unfortunately, this considerably increases the number of labellings for C junctions (phantom junctions); indeed, the two ends of a line may now be labelled by any pair of distinct surface-normal discontinuity labels ($+$, $-$, \leftarrow , \rightarrow) due to the possible presence of one or two undetectable C junctions along the line. The resulting constraints are too weak, in the sense that most drawings will now have an exponential number of legal labellings [32]. This problem can be avoided in the valued constraint framework by assigning non-zero costs to the new junction labellings in order to differentiate between the many different physically possible interpretations of a drawing.

The assumption of C^3 surfaces disallows points of infinite curvature, and thus, unfortunately, disallows apices of cones. This can be remedied by simply adding an extra labelling (\Leftarrow , \Leftarrow) to the list of labellings of L junctions (in Figure 3.2). However, allowing surface curvature to tend to infinity at any vertex also makes all junction labellings physically possible for all junction types and furthermore permits undetectable transitions between \Rightarrow and \rightarrow labels. Again, in the soft constraint formulation of line drawing labelling we can assign a large (but nevertheless finite) cost to all junction labellings which require surfaces of infinite curvature.

Determining whether a given label for a given line is part of a consistent labelling of a drawing of a polyhedral scene is NP-complete [92]. An interesting property of the catalogue of labelled junctions for objects with possibly curved C^3 surface patches is that, in this case, the same problem can be solved in linear time [34, 36] (Theorem 9.17). We can use this algorithm as a filter to reduce the size of the space to be searched for an optimal labelling.

5.2 Regularities in Man-Made Objects

Most man-made objects have certain characteristic shape features which distinguish them from natural objects such as rocks, clouds, trees or running water. In this chapter we concentrate on planarity and orthogonality, although clearly other regularities (such as symmetry and isometry) can also provide useful visual clues in the interpretation of drawings of man-made objects. We present a discrete labelling scheme for line drawings of curved objects which can be seen as an information-rich extension of the classic line-labelling scheme in which lines are classified as convex, concave, occluding or extremal. New labels are introduced to distinguish between curved and planar surface patches, to identify orthogonal edges and to indicate gradient directions of planar surface patches. Human vision has a marked preference for interpretations involving planar faces and orthogonal edges, which leads to a natural formulation of the problem as a soft constraint satisfaction problem.

We say that an edge E , whether curved or planar, is *orthogonal* if at each point on E the tangents to the two surfaces which meet at E are orthogonal. Consider, as an illustration, the two drawings in Figure 5.2. Of the 20 visible faces of these two objects, 13 are planar. Of the 57 visible edges, 51 are

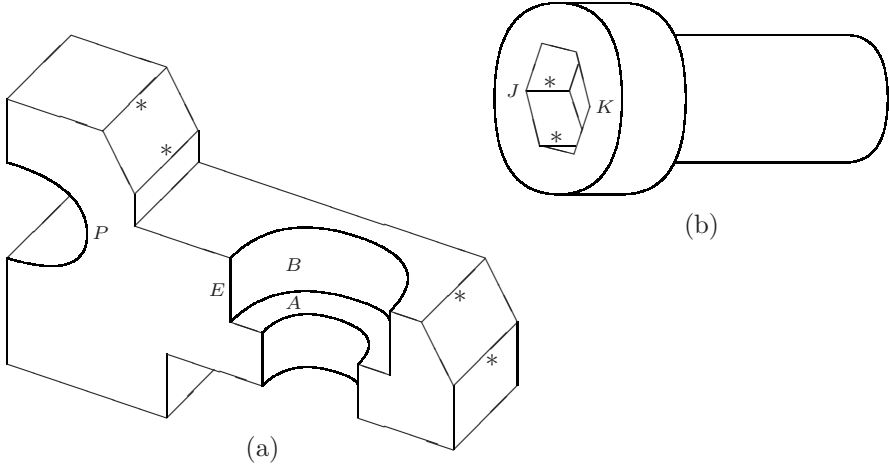


Figure 5.2: Examples of drawings of curved man-made objects (from [12] and [28]). Non-orthogonal edges are marked with an asterisk.

orthogonal, the 6 non-orthogonal edges being marked by an asterisk. Both of these drawings appear unambiguous to a human viewer. In particular, we immediately identify surface *A* in Figure 5.2(a) as planar and surface *B* as curved, although most people have great difficulty in explaining why they came to this conclusion. This chapter is concerned with setting down some basic local geometrical rules which will allow a computer to automatically identify planar surfaces and orthogonal edges. Possible applications include sketch interpretation, automatic indexing of databases of line drawings, the input of 3D object models, the creation of $2\frac{1}{2}$ D illustrations in electronic documents and object recognition in computer vision.

We follow in the tradition of Kanade [87, 88], who advocated the use of more information about the physical world, thereby avoiding overstrict constraints based on unrealistic assumptions (such as planar surfaces meeting at trihedral vertices [75, 20]) or the optimization of a fairly arbitrary objective function. In Chapter 3 we showed how the VCSP [140] provides a framework in which we can combine strict geometrical constraints and preference constraints. An example of a strict geometrical constraint is that parallel 3D lines cannot intersect; an example of a preference constraint is that we prefer a pair of lines which are parallel (within a given error tolerance) in a drawing to be projections of 3D parallel edges, since parallel edges are common features of man-made objects. Our approach is similar to that of Ding and Young [57], who used a truth maintenance system for the analysis of imperfect line drawings. We introduce new constraints for the analysis of line drawings of curved objects, whereas Ding and Young restricted themselves to polyhedral objects.

Human preference for orthogonal edges is so strong that it can outweigh our preference for physically possible interpretations. Figure 5.3 shows an example

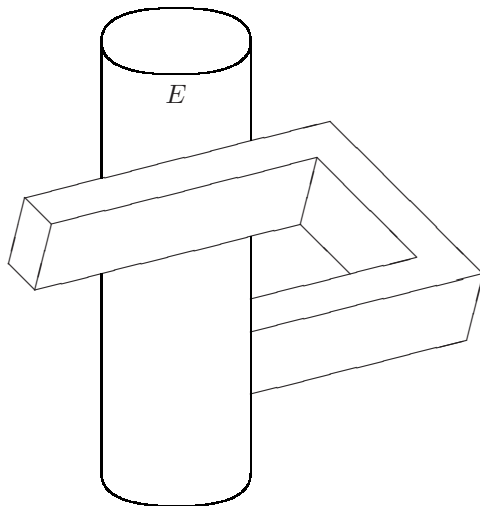


Figure 5.3: An unlikely drawing.

drawing in which all edges are interpreted by a human viewer as orthogonal, even though this is physically impossible. This drawing does have a physically possible interpretation in which edge E is not orthogonal.

Before introducing the strict and soft constraints that we use for the interpretation of line drawings of curved objects, we introduce some notation. If L is a line segment in the drawing, then $\text{orth}(L)$ means that L is the projection of a 3D orthogonal edge. A vertex is known as a *cubic corner* if it is a trihedral vertex at which three orthogonal edges meet.

5.3 Planarity Constraints

Consider a 3D edge E formed by the intersection of two surfaces S, T . We say that S is *locally planar* along E if the tangent plane to S at each point of E is identical. Planar surfaces are necessarily locally planar, but a curved surface may also be locally planar. As an example, consider surface B in Figure 5.2, which is locally planar along edge E , since it has an invariant tangent plane along the whole length of E .

A surface-normal discontinuity edge (a convex, concave or occluding edge) is formed by the intersection of two surfaces S, T . A line L which is the projection of a surface-normal discontinuity edge has a semantic label $+, -, \leftarrow$ or \rightarrow , as illustrated in Figure 5.4(a). We also label each side of L by a planarity label p (for planar) or c (for curved) to indicate whether S, T are locally planar or not. At an occluding edge, one of the surfaces, say T , is invisible. The planarity label of S is written on the side of the line onto which the surface S projects and the planarity label of T on the other side. This is illustrated in Figure 5.4(b): the rightmost label p indicates that the hidden base of the cylindrical part of the

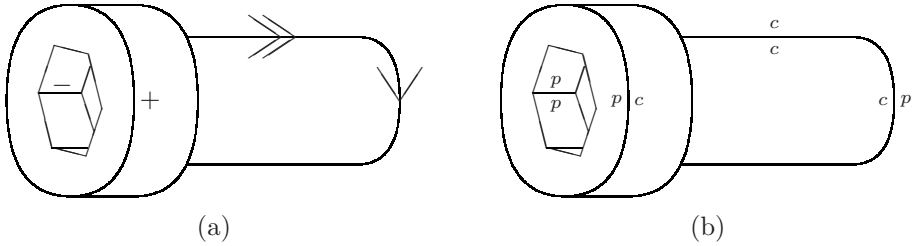


Figure 5.4: Illustration of (a) semantic line labels and (b) planarity labels.

object is locally planar. An extremal edge is the locus of points of intersection of the line of sight with a curved visible object surface. The projection of an extremal edge, known as an extremal line, is labelled by a double-headed arrow, as shown in Figure 5.4(a). By convention, both sides of an extremal line are always labelled c , as shown in Figure 5.4(b).

Let Tan be the set of tangent planes to all the locally planar segments of edges of 3D object(s) depicted in a drawing. We say that the drawing satisfies the GVA (general viewpoint assumption) if no small perturbation in the position of the viewpoint changes the configuration of the drawing (presence and types of junctions, presence of straight lines and parallel lines). Under the GVA, the viewpoint cannot lie in any tangent-plane in Tan .

Having introduced the new planarity labels, we can now give the planarity constraints which relate planarity and semantic line labels. These are given in Figure 5.5. Under the GVA, the projection of a 3D edge E is a straight line if and only if E is a straight edge. The first constraint in Figure 5.5 simply says that a curved line cannot be the projection of the intersection of two locally planar surfaces. The second constraint is simply our convention that both sides of an extremal line are labelled c . The third constraint is a translation of the straight-edge formation assumption.

The last three planarity constraints in Figure 5.5 (showing a phantom, a 3-tangent and a curvature-L junction respectively) allow us to deduce that a surface is curved. Consider a 3D point P at which a surface-normal discontinuity edge E intersects an extremal edge E_{ext} , and let S be the surface in which the extremal edge lies and T_P the tangent plane to S at point P . By the definition of an extremal edge, the viewpoint lies on plane T_P . If S were locally planar along E , then T_P would be the tangent plane to this locally planar segment of E , which would contradict the GVA. This reasoning allows us to deduce the c labels shown in the last three planarity constraints in Figure 5.5. There are three distinct constraints depending on whether the extremal edge E_{ext} or part of the surface-discontinuity edge E is occluded. The dot represents a discontinuity of curvature between the projections of E and E_{ext} . These planarity constraints are valid even in the case of objects with tangential edges and surfaces [34].

Applying the line-labelling catalogue to the drawings in Figure 5.2 and minimizing the number of phantom junctions, produces the correct semantic

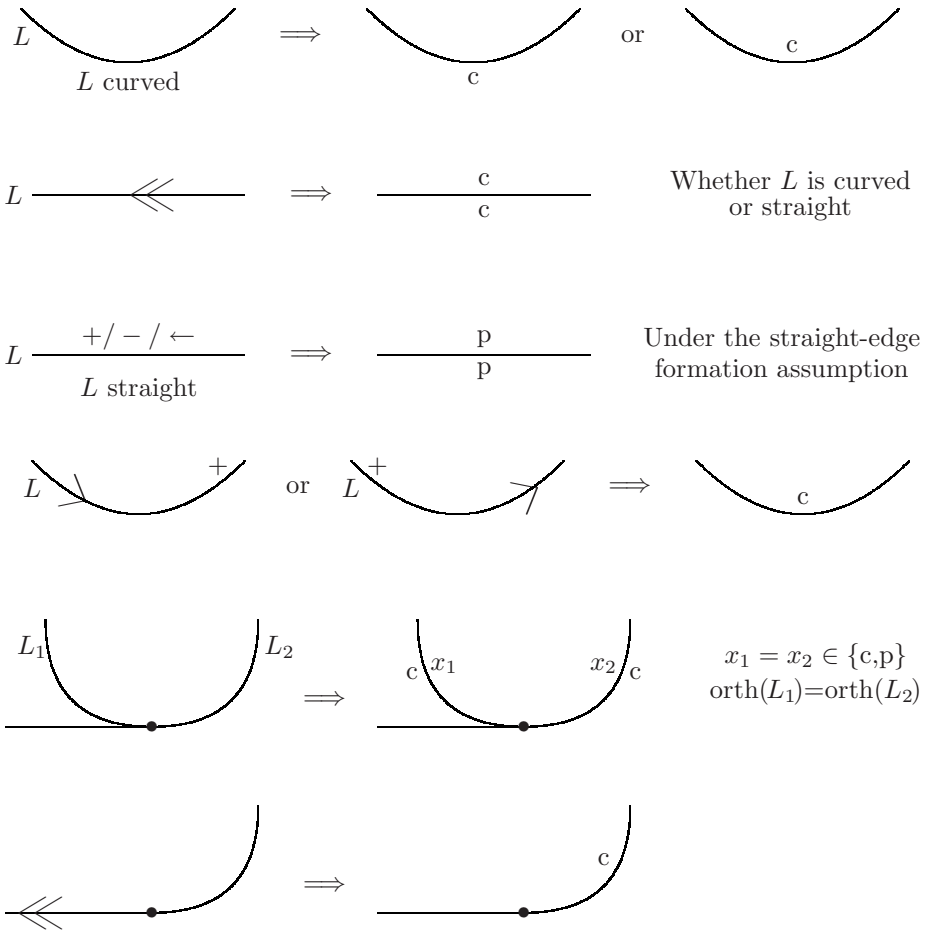


Figure 5.5: Planarity constraints.

labelling of both drawings. Applying the planarity constraints then allows us to identify certain surfaces as locally curved. Since planar surfaces are common in man-made objects, it is natural to then try to maximize the number of planar surfaces in our interpretation of a drawing. However, simply maximizing the number N_p of p labels is not always sufficient to determine the correct planarity labelling: for example, applying this criterion to the drawing in Figure 5.2(a) does not allow us to distinguish between the two labellings $\frac{c}{p}$ and $\frac{p}{c}$ for the edge separating faces A and B .

Considering the drawing as a planar graph G , the faces of G are projections of visible partial surfaces of a 3D object. A face F of G can only be the projection of (part of) a planar surface if all its planarity labels are p . We say that F is *locally planar* if all its planarity labels are p (and *curved* otherwise). Maximizing the number of faces of G which are locally planar is an extra criterion which, for example, allows us to find the correct planarity labelling of the drawing in Figure 5.2(a). Maximizing the number N_{lpf} of locally planar faces of G and maximizing the number N_p of p labels are complementary criteria in the sense that no drawing exists with two planarity labellings L_1, L_2 such that $N_{lpf}(L_1) > N_{lpf}(L_2)$ and $N_p(L_1) < N_p(L_2)$. Therefore, the criterion $N_p + N_{lpf}$ produces a partial order which refines the partial orders defined by the two individual criteria N_p and N_{lpf} .

Note that the planarity label pair of a curved line L can change as we walk along L , since a surface may be partially curved and partially planar. Such transitions can occur in C^∞ surfaces, meaning that there is no detectable trace of the transition in the drawing. We give an example in Section 5.5. Such undetectable c - p transitions do not affect the above discussion.

5.4 Constraints from Orthogonal Edges

It is well known that identifying a labelled junction as the projection of a cubic corner allows one to calculate the 3D orientation of the three faces that meet at the corresponding vertex [130, 89, 166] (Section 4.6). This section shows that we can extend this to curved objects by determining certain information about the surfaces which meet at viewpoint-dependent vertices, based only on the assumption that the 3D edge E is orthogonal.

Consider the first constraint shown in Figure 5.6, reading the implication from left to right. It concerns a labelled 3-tangent junction formed by the projection of a curved orthogonal edge E which is the intersection of a curved surface S_c with a planar surface S_p . Let P represent the 3D point at which the tangent plane to S_c on E passes through the viewpoint, and let T_P represent this tangent. If \mathbf{n} is the normal to the planar surface S_p , then by the orthogonality of E , \mathbf{n} is parallel to T_P . It follows that the projection of \mathbf{n} in the drawing is parallel to the extremal edge (which is the projection of T_P). We symbolize the direction of the projection of \mathbf{n} by a short arrow next to the p on the right-hand side of Figure 5.6.

Similar constraints exist and are given in Figure 5.6 for curvature-L junctions

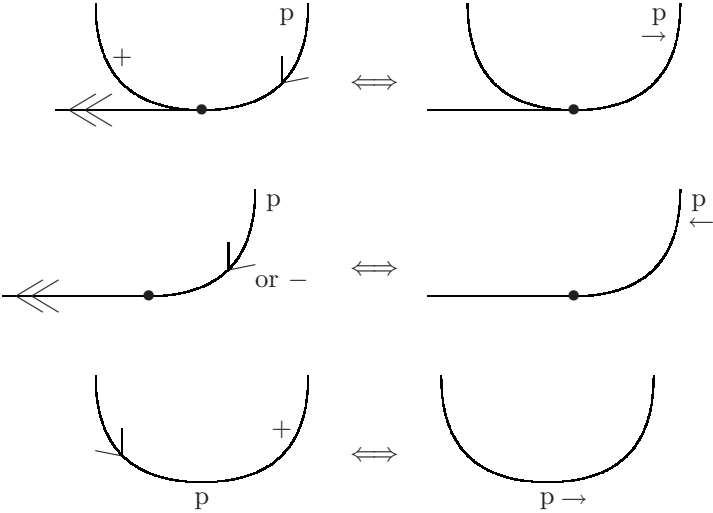


Figure 5.6: Gradient-direction constraints assuming that the surface-normal discontinuity edges are orthogonal. The \Leftarrow implications are consequences of the GVA.

and phantom junctions. The arrowhead indicates the direction of \mathbf{n} . By convention, the 3D orientation of \mathbf{n} , the normal to the planar surface S_p , is always towards the viewpoint. This convention means that in the second constraint, whether E is a concave or an occluding edge, the gradient-direction label is identical (i.e. in both cases the arrow points to the left in the second constraint of Figure 5.6). For the phantom junction shown in Figure 5.6, the gradient direction can only be determined accurately if the position of the phantom junction is precisely known. However, this is an important constraint when read from right to left, since it allows us to locate phantom junctions at the points where the gradient-direction is tangential to the line. The gradient direction is invariant on a planar surface under orthographic projection. Under perspective projection the gradient directions of a surface meet at a vanishing point; at least two gradient directions are required to locate the vanishing point of the normals to a planar surface.

There are also gradient-direction constraints at projections of trihedral vertices V at which surfaces and edges meet non-tangentially. Let L_1, L_2, L_3 be the projections of edges E_1, E_2, E_3 meeting at V , and let F_{12} be the face bounded by E_1 and E_2 . The normal \mathbf{n} to F_{12} is parallel to E_3 in 3D if and only if both E_1 and E_2 are orthogonal at V . Thus, invoking the GVA, which says that parallel lines in a drawing are projections of parallel 3D lines, we can deduce that the gradient direction of F is parallel to L_3 iff both E_1 and E_2 are orthogonal at V .

For each line segment L in a drawing we create a boolean variable $\text{orth}(L)$,

which takes on the value `true` if and only if L is the projection of an orthogonal edge segment. We then impose the strict constraints

`orth(L)` \Rightarrow the gradient-direction constraints apply at each
3-tangent, curvature-L and phantom junction on L

together with the soft constraint which imposes a penalty of w_o if `orth(L) = false`. Following the discussion in Section 5.3, there is also a penalty of w_{ph} for each phantom junction, a penalty of w_p for each c planarity label, a penalty of $w_{l_{pf}}$ for each face which is not locally planar, and a penalty w_t for each planarity label transition (p to c) between the two ends of a line segment. The objective function to be minimized is the sum of these penalties. Strict constraints can be considered as soft constraints whose penalty when they are violated is ∞ [115].

In order to determine the relative weight to be given to planarity and orthogonality, we constructed drawings having two interpretations, one involving more planar faces and the other involving more orthogonal edges. Figure 5.7(a) shows an example. We can interpret this as a box in which all faces are planar except the top face F or as a box in which F is planar but two hidden faces are curved. In the first interpretation, two edges, E_1 and E_2 , are non-orthogonal, whereas the second interpretation involves only one non-orthogonal edge, namely the hidden vertical edge. Since it is the first interpretation which seems to be the most natural, we conclude that human vision prefers planarity to orthogonality, i.e. $w_p > w_o$. Figure 5.7(b) shows another example: we tend to see F as planar and the two edges marked * as non-orthogonal, rather than F as a curved surface and the edges marked * as orthogonal. However, when other visual cues come into play, our preference for planarity can be overridden. Figure 5.7(b) is an example. There are two possible interpretations, one in which face F is planar (in which case edge E is not orthogonal) and another in which E is orthogonal (in which case F is not planar). The latter interpretation appears the most natural, but this is probably because the resulting 3D object has an extra axis of symmetry compared to the former interpretation.

Maintaining arc consistency during search is a standard technique for solving Constraint Satisfaction Problems. Arc consistency has recently been generalized to Soft Constraint Satisfaction Problems [50, 115] (Chapter 8), which has led to the development of efficient general-purpose intelligent complete search algorithms for soft constraint satisfaction [53].

The gradient-direction/semantic-label constraints in Figure 5.8 show the tight relationship between the gradient direction and the semantic label of a line L , whenever L is the projection of an orthogonal edge. The arrows represent any gradient direction which points away from L in the first constraint and towards L in the second constraint. Line L is shown curved but could be straight or curved in the opposite direction. At a point P on an orthogonal edge E at which surfaces S_1, S_2 intersect, the normal \mathbf{n}_1 (\mathbf{n}_2) to surface S_1 (S_2) is parallel to the tangent plane to surface S_2 (S_1) at P . If E projects into a convex (concave) line L , it follows that the projections of $\mathbf{n}_1, \mathbf{n}_2$ point away from (towards) L . Since, by convention, occluding lines have the same gradient-direction labels

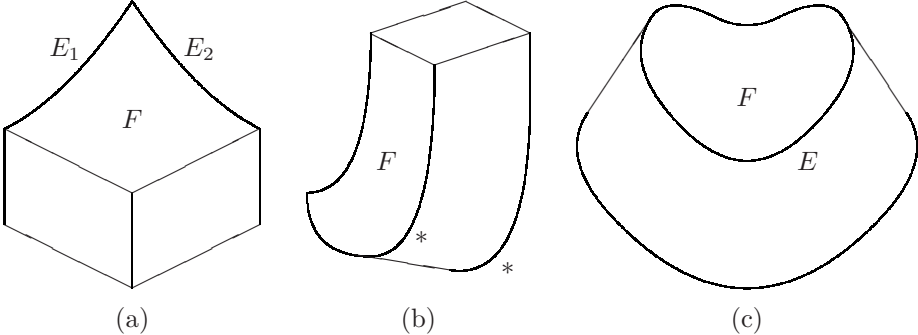


Figure 5.7: Examples of ambiguous pictures. Human vision appears to prefer planarity to orthogonality in (a) and (b) but orthogonality to planarity in (c).

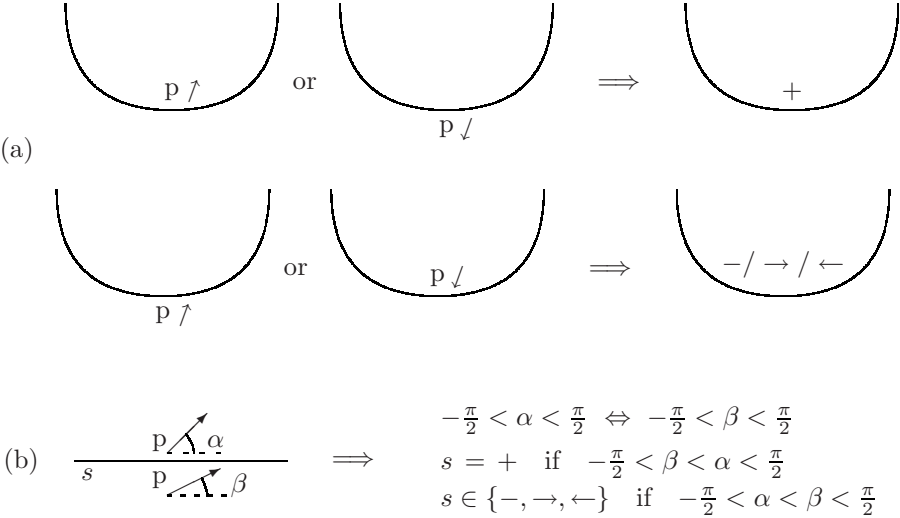


Figure 5.8: The gradient-direction/semantic-label constraints: (a) assuming that the edge is orthogonal; (b) for any edge.

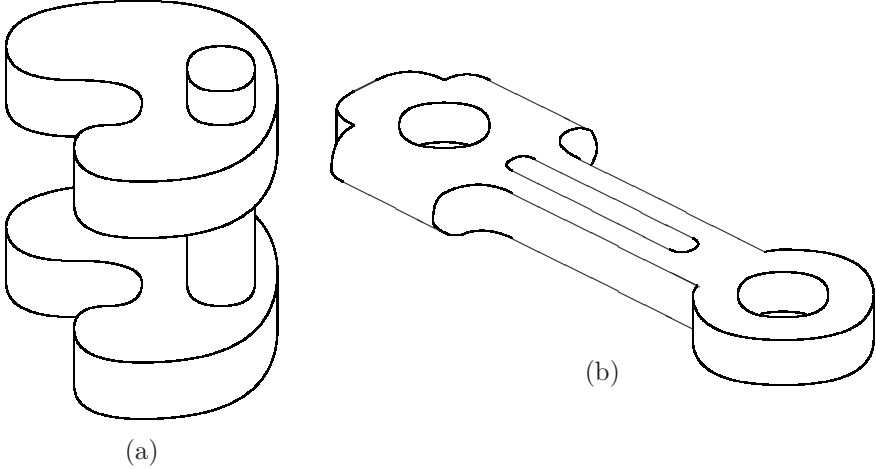


Figure 5.9: Drawings of curved objects with orthogonal edges ((b) is a version of a drawing in [149]).

as concave lines, this proves the correctness of the gradient-direction/semantic-label constraints given in Figure 5.8.

One consequence of these constraints is that a closed curve L which is the projection of a locally planar orthogonal edge cannot have the same semantic label around the whole of L . If L is not a closed curve but instead terminates at a 3-tangent or curvature-L junction, then the gradient-direction constraints allow us to determine the gradient direction. This, in turn, provides us with the semantic label at each point of L via the gradient-direction/semantic-label constraints.

In Figure 5.8(b), surfaces S_1, S_2 are both locally planar and hence L is necessarily straight. The right-hand end of L is further away from the viewer than the left-hand end if and only if $-\frac{\pi}{2} < \alpha < \frac{\pi}{2}$ and if and only if $-\frac{\pi}{2} < \beta < \frac{\pi}{2}$. Furthermore, the dihedral angle between the tangents to surfaces S_1 and S_2 is greater than π if $\alpha > \beta$, equal to π if $\alpha = \beta$ and less than π if $\alpha < \beta$. The constraints of Figure 5.8(b) follow immediately.

5.5 Examples of Drawing Interpretation

Figure 5.9 shows two objects containing only orthogonal edges. Consider the drawing in Figure 5.9(a). Using the semantic labelling scheme [109], the planarity constraints and the gradient-direction constraints, we obtain the labelling given in Figure 5.10 (with semantic labels not shown to avoid cluttering up the figure). This interpretation simultaneously maximizes the number of orthogonal edges, the number of p labels and the number of locally planar surface patches. All surfaces are correctly labelled as planar or curved and the directions of the projections of the normals to the planar surfaces have been correctly

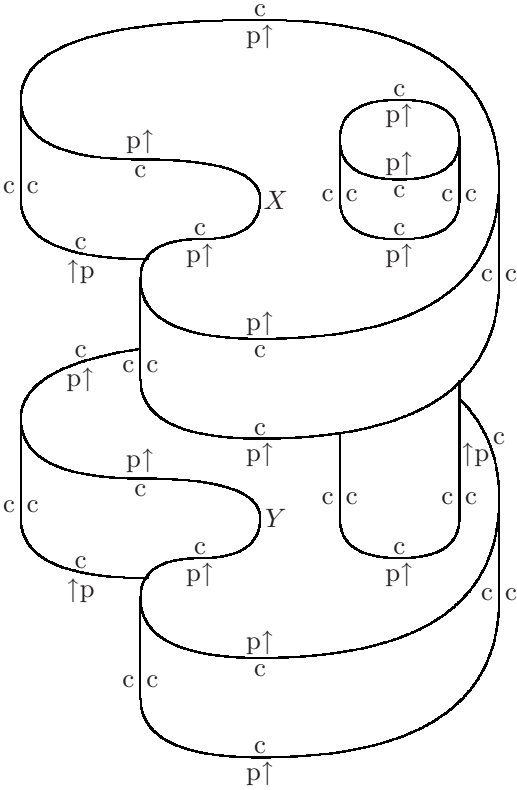


Figure 5.10: The drawing of Figure 5.9(a) with planarity labels and gradient directions.

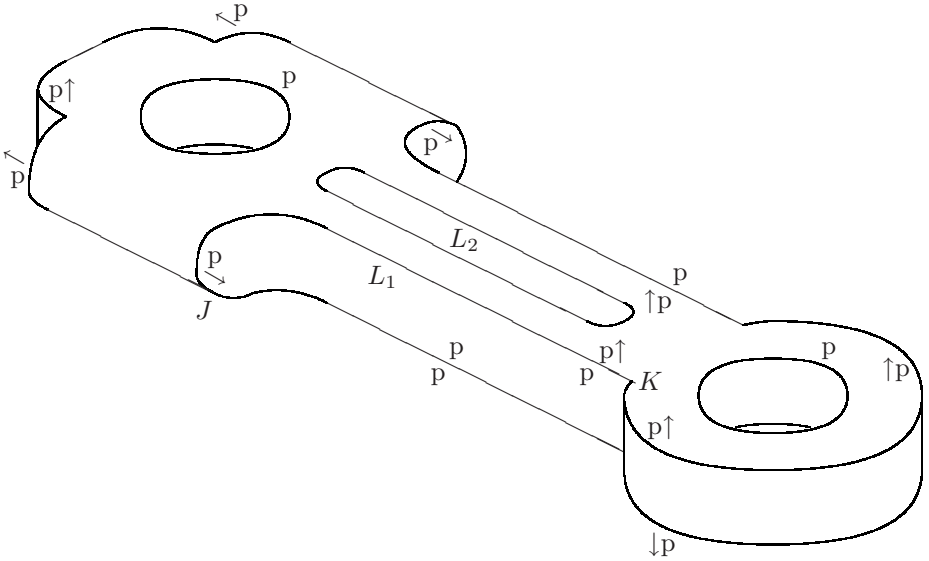


Figure 5.11: The drawing of Figure 5.9(b) with planarity labels and gradient directions.

determined. This provides a much richer interpretation than the semantic labelling alone. The third gradient-direction constraint (Figure 5.6) allows us to precisely locate the two phantom junctions X, Y shown in Figure 5.10 at which there is a transition from an occluding to a convex label.

Consider the drawing in Figure 5.9(b). Using the semantic labelling scheme, the planarity constraints and the gradient-direction constraints, we obtain the labelling illustrated in Figure 5.11. To avoid cluttering up the figure, c labels and semantic labels are omitted.

Line L_1 , which extends from junction J to junction K , provides an example of planarity-label transitions. Starting at junction J , the actual planarity label pair changes from $\frac{c}{p}$ to $\frac{c}{c}$ to $\frac{p}{p}$. These changes are not detected by our constraints, which strictly speaking only inform us about planarity labels in the vicinity of junctions or along the length of straight-line segments.

If a surface is planar along two edges which meet at a vertex, then we can propagate the gradient direction through the corresponding junction. For example, in Figure 5.11, the vertical gradient direction (deduced from the vertical extremal line) propagates through junction K to the straight segment of L_1 . However, neither planarity nor orthogonality propagates through junctions, except at 3-tangent junctions (as shown in Figure 5.5).

The closed curve L_2 in Figure 5.11, which we see as a long hole running down the handle of the object, provides a challenge for our constraints. It is the skew symmetry of L_2 whose axis coincides with the axis of symmetry of the whole object which allows us to identify L_2 as a hole in a planar surface rather than a separate object lying on top of a larger object. Such reasoning is necessary before

we can even find the correct semantic labelling, which is clearly a precondition to apply the planarity constraints and gradient-direction constraints. Naively minimizing the number of phantom junctions leads us to an incorrect semantic labelling. The gradient-direction/semantic-label constraints nevertheless allow us to deduce that a semantic labelling of L_2 involving no phantom junctions is incompatible with the hypothesis that L_2 is the projection of an orthogonal locally planar curve.

We performed trials on 28 drawings containing a total of 229 visible faces. Exactly 94.4% of orthogonality labels and 95.2% of planarity labels were correctly and unambiguously identified. Full details can be found in [45].

5.6 Complete 3D Reconstruction

We have seen that a rich labelling scheme, together with simple local geometrical constraints, allows us to obtain a considerable amount of information about the 3D object depicted in the drawing. This information goes much further than traditional semantic line labels (occluding, convex, concave, extremal) but does not in itself provide a complete 3D reconstruction. For this, we can look for various other common features of man-made objects, such as cubic corners and symmetry.

The drawing in Figure 5.2(a) contains 19 visible cubic corners, all of which involve edges which are parallel to just three orthogonal axes. Some workers apply the simple assumption, under orthographic projection, that if a large number of the lines in a drawing are parallel to one of just three principal directions, then these directions are projections of three orthogonal axes [103]. Another approach, described in detail in Chapter 4, searches for the largest set of mutually compatible cubic corners. For example, in Figure 5.2(a), junctions 1 and 2 cannot both be projections of cubic corners since two pairs of lines meeting at the junctions are parallel but the third line is not. Furthermore, junction 3 cannot be a projection of a cubic corner since it does not satisfy Perkins's criterion [17, 130]. In the case of Figure 5.2(a), both approaches allow us to correctly identify the cubic corners. This, in turn, allows us to calculate the 3D orientations of all the planar faces of the object (see Chapter 4 for more details).

Figure 5.2(b) contains no cubic corners. Indeed, maximizing the number of cubic corners leads to an incorrect reconstruction of the object in which J and K are projections of cubic corners. Although the gradient-direction constraints tell us that the normal to the leftmost planar face of the object is parallel to the extremal edges, we require further information to uniquely determine its 3D orientation. This information would be obtained, for example, by assuming that bilaterally symmetric curves are projections of local surfaces of revolution [120] or, alternatively, by an objective function which prefers equal angles (such as the minimum standard deviation of angles function of Marill [111]) or equal lengths (such as the standard deviation of segment lengths [15]).

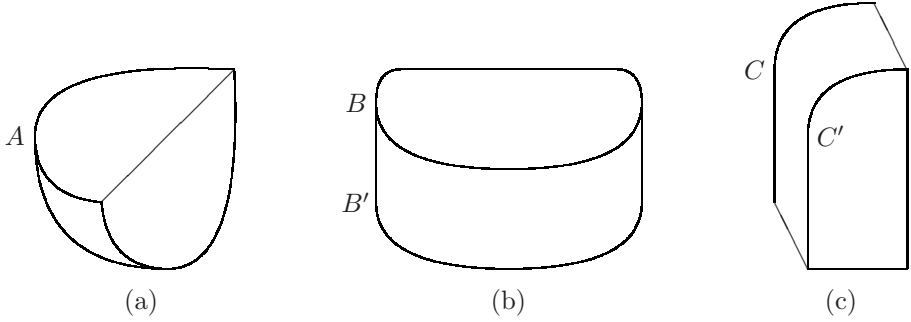


Figure 5.12: Examples of viewpoint-dependent (A , B , B') and viewpoint-independent (C , C') junctions.

5.7 Discussion

A viewpoint-dependent junction is a junction which is the projection of a 3D point whose position varies as a function of the viewpoint. Consider a drawing of a right cylinder. Let J be a viewpoint-dependent junction (i.e. a 3-tangent or curvature-L junction) in this drawing, at which an extremal line L_1 is tangential to a line L_2 (the projection of a surface-normal discontinuity edge E). In the case of a right cylinder, J necessarily occurs at a point of maximal curvature of curve L_2 , whereas line L_1 is straight. All viewpoint-dependent junctions in the drawings in Figures 5.2 and 5.9 have these properties.

These properties are common to viewpoint-dependent junctions, but neither is necessary even if E is an orthogonal edge and one of the surfaces meeting at E is planar. For example, at junction A in Figure 5.12(a), the extremal line is not straight, whereas in Figure 5.12(b), the curvature of the surface-normal discontinuity line is not maximal at junction B . Furthermore, a junction at which a straight line L_1 is tangential to a curved line L_2 at a point of maximal curvature of L_2 is not necessarily the projection of a viewpoint-dependent junction. The viewpoint-independent junction C in Figure 5.12(c) is an example. Nevertheless, the presence of a straight line L_1 and/or a point of maximal curvature of L_2 at a curvature-L junction J increases the likelihood that J is a viewpoint-dependent junction.

We say that two curve segments are *parallel* if they can be put in correspondence by a translation and *similar* if they can be put into correspondence by a translation followed by a dilation. Under orthographic projection, parallel (similar) 3D curve segments project into parallel (similar) 2D curve segments. The presence of two pairs of parallel curve segments at junctions C and C' (or at junctions B and B') in Figure 5.12 indicates that either both junctions are viewpoint dependent or both junctions are viewpoint independent.

Chapter 6

Depth Recovery Through Linear Algebra

This chapter provides a summary of the most important results in the use of linear algebra in determining the realizability of line drawings and recovering missing depth parameters. The coverage of gradient and dual space is largely based on Draper's article [59], and most of the linear constraints for curved objects were first published in [37].

6.1 Gradient Space and Gradient Directions

Throughout this chapter we assume a left-hand coordinate system, so that larger values of z represent more distant points. In this section, we assume that the drawing has been obtained by orthographic projection with the viewpoint at the point at infinity along the negative z -axis. This means that a 3D point (x, y, z) projects into the point (x, y) in the picture plane. The equation of a plane in 3D space is given by

$$ax + by + cz + d = 0. \quad (6.1)$$

To emphasize that a plane has only three degrees of freedom, we can rewrite this as

$$z = px + qy + r, \quad (6.2)$$

where $p = -\frac{a}{c}$, $q = -\frac{b}{c}$, $r = -\frac{d}{c}$. The space (p, q, r) is the *dual space* of the original (x, y, z) space and was first introduced in the interpretation of line drawings by Huffman [76]. Mackworth [107] used *gradient space* (p, q) to check the realizability of line drawings as projections of polyhedra. Gradient space can be seen as an orthographic projection of the dual (p, q, r) space, in the same way that the picture space (x, y) is an orthographic projection of 3D (x, y, z) space.

A special case occurs when $c = 0$ in Equation (6.1), which corresponds to a plane parallel to the z -axis. These planes are mapped to (∞, ∞, ∞) in

dual space. However, under orthographic projection, the general viewpoint assumption (GVA) says precisely that no planes in a 3D scene are parallel to the line of sight (the z -axis) and hence $c \neq 0$.

Points (p, q, r) in dual space correspond to planes in the original (x, y, z) space (and vice versa). Consider a 3D line L which is the intersection of two planes whose corresponding points in dual space are (p_1, q_1, r_1) and (p_2, q_2, r_2) . Points on L lie on both these planes and hence satisfy Equation (6.2) for (p_1, q_1, r_1) and (p_2, q_2, r_2) . After elimination of z , we obtain

$$(p_1 - p_2)x + (q_1 - q_2)y + r_1 - r_2 = 0. \quad (6.3)$$

Let $(x_1, y_1), (x_2, y_2)$ be two distinct points on L . Substituting into Equation (6.3) and eliminating $r_1 - r_2$ gives

$$(p_1 - p_2)(x_2 - x_1) + (q_1 - q_2)(y_2 - y_1) = 0. \quad (6.4)$$

The slope of the projection L' of L in the picture plane is given by $(y_2 - y_1)/(x_2 - x_1)$.

Now consider a line L_d joining points (p_1, q_1, r_1) and (p_2, q_2, r_2) in dual space. The projection L'_d of L_d in gradient space (i.e. (p, q) space) has a slope of $(q_2 - q_1)/(p_2 - p_1)$. If we artificially align the p, q axes of gradient space with the x, y axes of picture space, then Equation (6.4) tells us that L' and L'_d are perpendicular. This property can be used, for example, to prove the following simple lemma (which we already implicitly used in Section 4.7).

Lemma 6.1 *Let A, B, C be three planes in (x, y, z) space which intersect at a point. The gradient (p_B, q_B) of plane B can be uniquely determined from the gradients $(p_A, q_A), (p_C, q_C)$ of planes A, C together with the orthographic projections in the (x, y) plane of the lines of intersection L_{AB} and L_{BC} of the pairs of planes A, B and B, C .*

Proof: The projection in the picture plane is illustrated in Figure 6.1(a). Denote by H_{AB} (respectively H_{BC}) the line in gradient space, passing through the point (p_A, q_A) ((p_C, q_C)) and which is perpendicular to L_{AB} (L_{BC}). Then (p_B, q_B) lies at the intersection of H_{AB} and H_{BC} , as shown in Figure 6.1(a). ■

Another important result, already stated in Section 4.6, is that the orthographic projection of a cubic corner V , together with its semantic labelling, uniquely determines the gradients (p, q) of the three surfaces meeting at V . Drawing analysis in gradient space provides necessary but not sufficient conditions for a drawing to be realizable as an orthographic projection of a polyhedral scene, due to the fact that the third parameter r is ignored. Gradient-space analysis was, to a large extent, superseded by Sugihara's work [155], which involves reasoning in (p, q, r) space and, furthermore, applies equally well to the case of perspective projection. The gradient direction of a plane P , introduced in Chapter 5, is the direction in the image plane of the projection of the normal to P . It is given by $\tan^{-1}(q/p)$ and is hence even less informative than the gradient

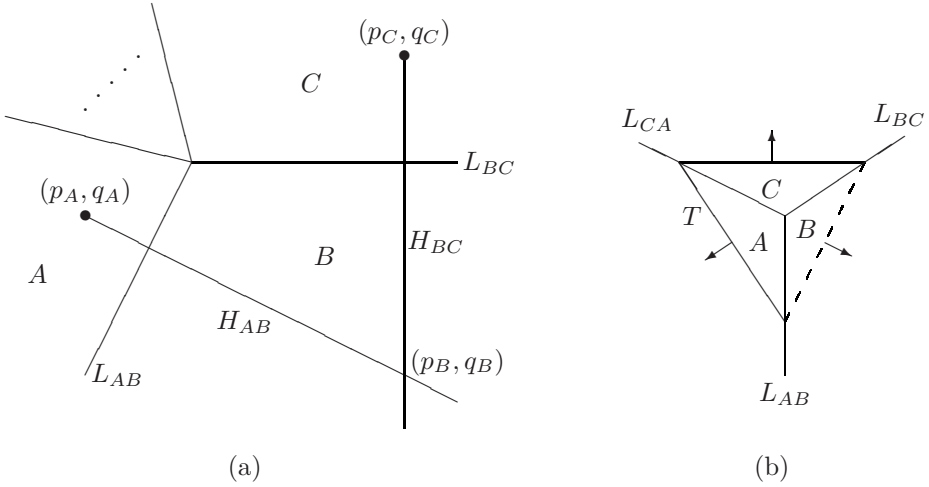


Figure 6.1: (a) The gradient of plane B is uniquely determined by the gradients of planes A and C . (b) The gradient direction of B is uniquely determined by the gradient directions of A and C .

(p, q) . Although gradient-directions provide an incomplete description of planes, we have seen in Chapter 5 how they can be incorporated into a discrete labelling scheme to enrich the classical semantic labels (convex, concave, occluding, etc.).

A similar result to Lemma 6.1 also exists concerning gradient directions. If N planar faces meet at a vertex V , then knowledge of the projection of V , together with its semantic labelling and the gradient-direction of $N - 1$ of the faces, uniquely determines the gradient-direction of the N th face. We give a formal proof of this result below for the case of trihedral vertices.

Lemma 6.2 *Let A, B, C be three planes in (x, y, z) space which intersect at a point V . The gradient direction α_B of plane B can be uniquely determined from the gradient directions α_A, α_C of planes A, C together with the orthographic projections in the (x, y) plane of the lines of intersection L_{AB}, L_{BC}, L_{CA} of each pair of planes.*

Proof: Let P be a plane $z = z_0$ which does not pass through V . Let T be the triangle formed by the intersection of P with planes A, B, C . The three sides of T are each perpendicular to the gradient direction of the corresponding face. To see this, let T_A be the side of T lying in face A , and let $\mathbf{n} = (p, q, r)$ be the normal to face A . Let $(e, f, 0)$ be the unit vector parallel to T_A . Then, since T_A and \mathbf{n} are orthogonal, we have $(p, q) \cdot (e, f) = (p, q, r) \cdot (e, f, 0) = 0$.

Figure 6.1(b) shows the projections into P of the intersection lines L_{AB}, L_{BC}, L_{CA} together with triangle T . The slopes of two of the sides of T clearly uniquely determine the slope of the third side T_B (shown as a dashed line in

Figure 6.1(b)). The gradient direction of B (shown as a short arrow in the figure) is then perpendicular to T_B . ■

Given three non-collinear 3D points lying on the same plane, the parameters (p, q, r) of the plane (as in Equation (6.2)) can easily be eliminated. This leaves a linear system whose only unknowns are the depths z (or inverses of depths in the case of perspective projection). Dual space and gradient space were introduced for the analysis of drawings of planar-faced objects, but they are not so obviously useful when object surfaces can be arbitrary curved surfaces. However, in the more general case of curved objects, linear constraints often exist, such as the coplanarity of 3D points deduced from the presence of straight lines in the drawing. As we will show in the rest of this chapter, these constraints can be exploited to obtain a linear system in which the unknowns are the depths z (or inverses of depths in the case of perspective projection) of 3D points.

6.2 Linear Constraints and Curved Objects

Drawings of curved objects often contain many linear features: straight lines, collinear or coplanar points, parallel lines and vanishing points. These linear features give rise to linear constraints on the 3D position of scene points which can be resolved using standard linear programming techniques. An important characteristic of this approach is that instead of making a strong assumption, such as that all surfaces are planar, only very weak assumptions, which disallow coincidences and highly improbable objects, need to be made to be able to deduce planarity. The linear constraints, combined with junction-labelling constraints, are a powerful means of discriminating between possible and impossible line drawings. They provide an important tool for the machine reconstruction of a 3D scene from a human-entered line drawing.

Human vision combines evidence from many sources in order to choose a unique interpretation for a line drawing. Information extracted from straight lines, collinear points, parallel lines and junctions in the drawing is rapidly integrated to reduce the inherent ambiguity due to the loss of one dimension when a 3D scene is projected into a 2D drawing. The speed of human visual interpretation of line drawings is no doubt partly due to the abundance of such linear constraints in most drawings encountered in practice.

Pioneering work on machine interpretation of line drawings concentrated almost exclusively on assigning semantic labels to lines in drawings of polyhedral scenes [20, 75, 87]. Sugihara [154, 155] was able to state necessary and sufficient conditions for a drawing to be the projection of a polyhedral scene by using not only these semantic labelling constraints but also constraints derived from the assumption that all faces were planar. He expressed these planarity constraints as linear equations between variables corresponding to the parameters of object faces. Constraints derived from a given semantic labelling of the drawing were expressed as linear inequalities. The result was a standard linear programming problem.

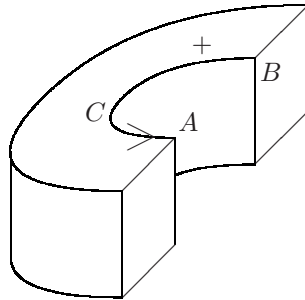


Figure 6.2: Example of a phantom junction C at which a transition occurs between $+$ and \rightarrow labels.

Unfortunately, the restriction to planar-faced objects means that Sugihara's work cannot be directly applied to line drawings of curved objects. Nevertheless, the spirit of his work can be retained whenever linear constraints, such as collinearity or coplanarity, are applicable. Many drawings of curved objects contain straight lines, collinear points, parallel lines or coplanar points, which are, in fact, the key to the interpretation of the drawings. Given an arbitrary 2D curve there is an infinite family of 3D curves which project into it, whereas a straight line in a drawing can, barring coincidences, be assumed to be the projection of a straight line in 3D.

In this chapter, a drawing consists of a set of junctions linked by a set of possibly curved lines. Unlike Sugihara, who used the assumption of polyhedral objects, object faces can be of any C^3 shape. The aim will, therefore, not be to determine the 3D equations of faces, but simply to determine the positions in 3D space of each vertex projecting into a visible junction in the drawing. The following sections give the mathematical derivation of linear constraints in the case of both orthographic and perspective projection. The power of linear constraints and semantic labels can be demonstrated by their ability to distinguish between possible and impossible drawings.

Drawings are assumed to be perfect projections, from a general viewpoint, of scenes containing objects composed of opaque C^3 surfaces separated by C^3 surface-normal discontinuity edges meeting non-tangentially at vertices. Straight edges are assumed to be formed by the intersection of locally planar surfaces. This means that objects behave locally as polyhedra in the vicinity of vertices and straight edges. However, no extra restriction needs to be imposed, such as trihedral vertices [20, 75] or the cyclic-order property [145, 146]. The propagation of semantic labels for lines is sufficient to identify certain well-known examples of drawings as impossible, such as the impossible forks shown in Figure 2.5.

Under the assumption that straight lines are projections of straight edges formed by the intersection of locally planar surfaces, label transitions on straight lines are illegal. Nevertheless, when the surfaces which meet to form an edge may be curved, undetectable transitions from convex to occluding labels are

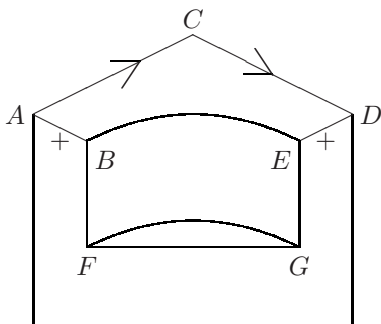


Figure 6.3: An object which is impossible because coplanarity constraints imply that points A, B, E, D should be collinear.

possible and are known as C junctions or phantom junctions. Figure 6.2 shows such a transition at point C lying somewhere between A and B .

Figure 6.3 illustrates an object which is impossible because of coplanarity constraints (which are given in detail below in Section 6.4). It is possible to deduce that A, B, C, D, E are all coplanar by the form of the corresponding junctions in the drawing and by the assumption that surfaces are planar in the vicinity of straight lines. Similarly, the 3D points A, B, F, G, E, D must be coplanar. Together, these two facts imply that points A, B, E, D are collinear since they all lie on the intersection of these two planes. However, the projections of A, B, E, D in the drawing are clearly not collinear. The following section gives a formal mathematical statement of linear constraints that can be used to demonstrate the impossibility of drawings such as Figure 6.3, the Penrose triangle or Escher's 'Belvedere'. The main aim of these constraints is to allow a machine vision system to interpret line drawings of real 3D scenes by determining the position in 3D of object vertices. The fact that these constraints can successfully distinguish between possible and impossible drawings demonstrates their power but is not their main purpose.

6.3 Formulation of Linear Constraints

A perspective projection is assumed with focal length f . Note that an orthographic projection produces slightly different constraints, which are given below in Section 6.5. Let (X, Y, Z) represent the 3D scene coordinates and (x, y) the 2D image coordinates. Under perspective projection,

$$(x, y) = (Xf/Z, Yf/Z). \quad (6.5)$$

The unknowns of the problem are the Z -coordinates of each junction in the drawing together with the semantic labels of each line end. Each line end must be assigned a semantic label such as concave, convex, occluding or extremal. The two ends of the same curved line do not necessarily have the same label,

as was illustrated by the example of Figure 6.2. Semantic labelling of drawings of curved objects is discussed extensively elsewhere in this book, and we do not repeat here the constraints on junction labellings since they are a function of the different restrictions placed on the shape of objects. Instead we concentrate on the constraints involving the Z -coordinates of junctions. Note that some constraints concern both the line labels and the Z -coordinates, meaning that the line-labelling problem and the determination of the Z -coordinates cannot be solved independently.

Suppose that the junctions are numbered from 1 to n , and let (x_i, y_i) be the image coordinates of junction i and (X_i, Y_i, Z_i) the scene coordinates of the vertex which projects into junction i . Under perspective projection, the constraints give rise to linear equations and inequalities not between the values Z_i , but between their inverses, which we denote by $t_i = 1/Z_i$. Three types of constraint must be expressed between 3D points h, i, j, k in terms of the values of the variables t_h, t_i, t_j, t_k :

- Point i is nearer (further) than point j ; or i and j are at equal distance from the projection plane.
- Points i, j, k are collinear.
- Point h lies in front of (behind) the plane of points i, j, k , or points h, i, j, k are coplanar.

The constraint that point i is nearer than point j is encoded by $t_i > t_j$ and equidistance of i and j from the projection plane is encoded by $t_i = t_j$.

It is easily shown that points i, j, k are collinear if and only if

$$d_{jk}t_i + d_{ki}t_j + d_{ij}t_k = 0,$$

where $d_{ij} = x_i - x_j$ (and similarly for d_{jk} and d_{ki}) unless $x_i = x_j = x_k$, in which case $d_{ij} = y_i - y_j$.

In Figure 6.4(a), four 3D points h, i, j, k are shown. The unit vectors e_i, e_j, e_k are the orientations in 3D space of the lines hi, hj, hk . As a concrete example, we can consider e_i, e_j, e_k to be the orientations of the tangents to the three edges leaving vertex h in the direction of points i, j, k . Each of hi, hj, hk is shown as a broken line since there may be no continuous straight edge joining h to i, j, k . Points i, j, k may even be vanishing points rather than vertices. Point j in Figure 6.4(b) is an example of a vanishing point. In Figure 6.4(b), the edge leaving h in the direction of i is curved. Vector e_i is the orientation of the tangent to this curve at h . Since in the drawing the projection of e_i coincides with the 2D line hi , by implicitly using a collinearity constraint we deduce that e_i coincides with line hi in 3D space.

Returning to Figure 6.4(a), a test for whether h is in front of (i.e. on the same side as the centre of projection of) plane ijk is

$$(e_i \wedge e_j) \cdot e_k < 0, \tag{6.6}$$

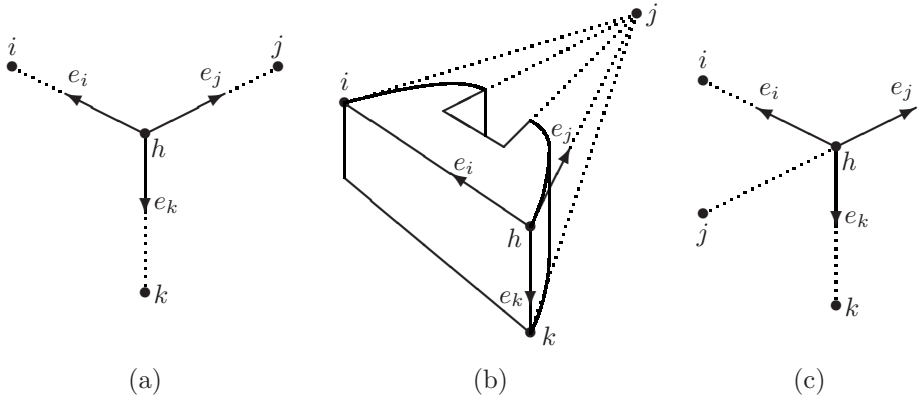


Figure 6.4: (a) e_i, e_j, e_k are the 3D orientations of the edges meeting at vertex h . (b) An example of a vertex h involving a vanishing point j . (c) The case in which the vanishing point j is on the opposite side of vertex h .

assuming a left-hand coordinate system. This is true for any set of four 3D points h, i, j, k , provided that the triangle of the projections of points i, j, k has a clockwise orientation in the drawing. An anticlockwise orientation simply produces a change of sign and test (6.6) becomes $(e_i \wedge e_j) \cdot e_k > 0$.

For notational convenience, let v_i represent (x_i, y_i, f) , where f is the focal length of the perspective projection. In fact, for the constraints given in this chapter, knowledge of f is not necessary and we can simply set $f = 1$. By the definition of the perspective projection in Equation (6.5), $(X_i, Y_i, Z_i) = v_i Z_i / f$. It follows that

$$\begin{aligned} e_i &= a(v_i Z_i - v_h Z_h), \\ e_j &= b(v_j Z_j - v_h Z_h), \\ e_k &= c(v_k Z_k - v_h Z_h), \end{aligned}$$

for some strictly positive constants a, b, c . Substituting these equations in (6.6) and simplifying gives the following criterion for deciding whether h is in front of plane ijk :

$$a_{ijk} t_h + a_{jih} t_k + a_{hik} t_j + a_{jhk} t_i < 0, \tag{6.7}$$

where $a_{ijk} = (v_i \wedge v_j) \cdot v_k$, with $a_{jih}, a_{hik}, a_{jhk}$ defined similarly. This is a linear inequality in terms of t_h, t_i, t_j, t_k . Recall that $t_i = 1/Z_i$, with t_h, t_j, t_k defined similarly.

Note that there is a complete change of sign for each point i, j, k whose projection is actually on the opposite side of the junction. For example, if the vanishing point j were, in fact, as shown in Figure 6.4(c), then condition (6.7) would become

$$a_{ijk} t_h + a_{jih} t_k + a_{hik} t_j + a_{jhk} t_i > 0.$$

The final constraint which we have to formulate mathematically is that four points h, i, j, k are coplanar. This constraint is given by the linear equality

$$a_{ijk}t_h + a_{jih}t_k + a_{hik}t_j + a_{jhk}t_i = 0. \quad (6.8)$$

6.4 Deriving Linear Constraints from a Drawing

6.4.1 Vanishing Point Constraint

A classical assumption in the machine interpretation of line drawings is to limit the number of edges which can meet at a vertex. For example, in the case of trihedral vertices [20, 75], at most three edges meet at each object vertex. A natural extension of this assumption is to say that at most N (non-collinear) edges would meet at a point in 3D even if they were extended to any finite distance in both directions. It follows, from this assumption and the general viewpoint assumption (GVA), that the intersection of $N+1$ extensions of lines in a drawing provides evidence of a vanishing point. Various authors have described practical techniques for the detection of vanishing points in images of real scenes [150, 158].

Vanishing points are a useful visual cue in scenes containing many parallel lines. All lines converging to a given vanishing point are projections of parallel 3D lines. These 3D lines can be considered to meet at a point which lies at an infinite distance from the viewer. If we have deduced that the projection of i is a vanishing point, then we can apply the vanishing-point constraint: $t_i = 0$.

6.4.2 Constraints from Collinearity or Intersections

Section 6.3 described how to express the collinearity of three scene points in terms of the variables t_i . The collinearity constraint simply states that any three collinear points in a drawing are projections of collinear points in 3D. Note that one of the points in the drawing may be a vanishing point. It is by the combined use of the vanishing-point constraint and the collinearity constraint that we exploit the presence of parallel lines under perspective projection.

The dual of the collinearity constraint states that any three straight lines meeting at a point P in a drawing are projections of straight lines meeting at a point q in 3D space. No essentially new constraint is involved, just the identification of q as a 3D point whose coordinates are to be determined. The collinearity constraint can then be applied to each of the lines L meeting at q : if i and j lie on L , then i, j, q are collinear. This dual constraint is redundant when P is a junction or has already been identified as a vanishing point, but it is useful, for example, when P is the intersection of the extensions of three lines in the drawing, as in Figure 6.5. In this case, although it cannot be determined whether P is a vanishing point or the projection of a hidden or truncated trihedral vertex, we can still deduce, for example, that i, j, k, l are coplanar in 3D.

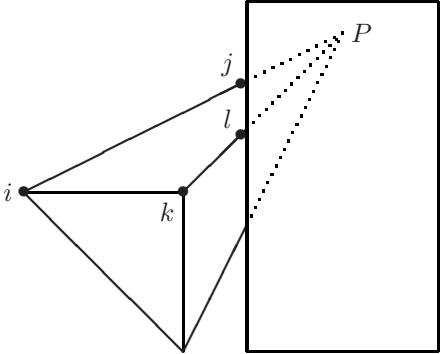


Figure 6.5: The intersection of three straight lines at P implies that i, j, k, l are coplanar since lines ij and kl must intersect in 3D space.

6.4.3 T-junction Constraint

A T junction is not necessarily caused by occlusion. Non-occlusion T junctions occur, for example, at the join of two long rectangular blocks which are stuck together in the form of a cross [32]. In a drawing of a multi-object scene, a T junction may be the projection of a point at which two object edges touch. In a perfect projection of a scene from a general viewpoint, the bar of a T junction is the projection of a single edge which passes in front of or touches the edge projecting into the stem of the T.

In order to express this mathematically, we need to distinguish two points at every T junction, as shown in Figure 6.6(a). The 3D points i and j project into the same T junction, but they lie on the two different edges. The basic T junction constraint is thus

$$t_i \leq t_j.$$

However, if it is known that the two edges do not touch, which is the case, for example, if either line has been labelled as extremal (i.e \Leftarrow or \Rightarrow) or the stem of the T has been labelled as concave ($-$), then point j is strictly nearer to the viewer than i :

$$t_i < t_j.$$

On the other hand, if it is known that the two edges do touch, which is the case, for example, if one half of the bar of the T has been labelled as concave, then the constraint is clearly

$$t_i = t_j.$$

There are certain restrictions on the application of the collinearity constraint (described in Section 6.4.2) to T junctions. If the occluding edge E is a straight edge, then the occluded point i should clearly not be considered to be collinear with E in 3D space. Similarly, if the occluded edge E' is a straight edge, then the occluding point j should not be considered to be collinear with E' . However, in all other cases, the fact that a T junction is a member of a set of collinear

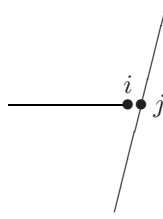


Figure 6.6: The two 3D points i, j which project into a T junction.

points (at least two of which are viewpoint independent) implies, under the GVA, that the two edges touch in 3D and hence $t_i = t_j$. The presence in a drawing of shadows or reflections leaving a T junction can also provide evidence that the two edges touch in 3D, under an assumption of general viewpoint and general light-source positions.

To distinguish the two distinct types of T junction (edges touching or not touching), a semantic label must be assigned to each T junction in a drawing. The labels ‘=’ and ‘>’ indicate, respectively, that the two edges touch or do not touch at a 3D point projecting into the T junction. The introduction of a new label for T junctions (= or >) is a way of ensuring that the determination of the values of the variables t_i remains a classical linear programming problem.

Further constraints on T junctions exist, but only in the case where the junction is labelled ‘=’ and is part of a pair of junctions joined by a straight line. Such constraints involving pairs of adjacent junctions are described below, in Sections 6.4.5 and 6.4.6.

In this section we have assumed that T junctions are caused by an edge occluding another edge (‘>’) or by two surface-normal discontinuity edges intersecting in 3D space (‘=’). If shadows [38], cracks [173], reflections or other surface markings occur in a drawing, then this gives rise to yet another type of T junction, which is neither ‘>’ nor ‘=’.

6.4.4 Convex/Concave Edge Constraints

Various catalogues of legal junction labellings exist for drawings of objects with curved surfaces [32, 34, 36, 38, 109]. Different assumptions on object shape give rise to different catalogues. In order to be able to state very general constraints derived from linear features in a drawing, we do not specify which catalogue is used. Nonetheless, we suppose that some such catalogue has been applied to gain certain information about line labels. The constraints described in this section are derived from knowledge that a line end or a straight line is labelled as concave (+) or convex (-). Whereas Parodi and Torre [127] deduced line-label information from knowledge of the directions of the three edges meeting at a vertex, we deduce information concerning edge directions from knowledge of line labels. The resulting constraints are not simple inverses of those stated by Parodi and Torre but also generalizations to curved surfaces meeting at

non-trihedral vertices.

In order to be able to deduce useful constraints, we require that objects behave locally as polyhedra in the vicinity of vertices or straight edges. To be more specific, we require that:

- For any two object surfaces S_1, S_2 meeting along an edge E incident to a vertex V , the tangent planes T_1, T_2 to S_1, S_2 at V intersect in a straight line L which is tangential to E at V . (This follows, for example, from the assumption that objects have C^3 surfaces and that no edges or surfaces meet tangentially.)
- For any straight edge E on a surface S , there is a unique tangent plane to S along the whole length of E .

Note, however, that we do not need to make the assumption that vertices are trihedral.

Sugihara [154] formulated a basic constraint on convex/concave edges formed by the intersection of two planar faces: when two planar faces F_1, F_2 meet at an edge E and point i lies on F_1 (but not on the plane of F_2), then the semantic label (convex or concave) for E determines whether i lies in front of or behind the plane of F_2 . For example, in Figure 6.7(a), the fact that jk is a concave edge implies that i is necessarily in front of plane jdk . To apply this constraint to curved objects, we have to consider the polyhedral vertex formed by the tangent planes to the surfaces, rather than the surfaces themselves.

Suppose that edge E , incident to a vertex j , is the intersection of two surfaces S_1 and S_2 which are both visible in a drawing. E is thus either a convex or a concave edge. Suppose that k lies on the straight line in 3D which is tangent to E at j . Let T_1, T_2 be the tangent planes to S_1, S_2 at j and suppose that we know that points i, l lie on T_1, T_2 , respectively, but not on their intersection (which, by the first assumption above, is line jk). Knowing whether E is a convex or concave edge, we can deduce whether i is in front of or behind the plane passing through j, k, l (which is exactly T_2). Section 6.3 described how to code such constraints mathematically as strict linear inequalities.

In Figure 6.7(a), the concavity of jk implies that i is in front of plane jdk . In fact, this remains true even if edge jl is an occluding edge passing in front of the surface incident to the concave edge, as illustrated in Figure 6.7(b). Indeed, in this particular case, in which jk is a concave edge and the projections of i and l lie on opposite sides of the projection of jk , the constraint holds whatever the labelling of edges ij and jl .

In all, there are four distinct cases to consider, depending on whether edge jk is labelled convex or concave and whether the projections in the drawing of points i, l lie on the same or different sides of the projection of jk . These four cases are illustrated in Figure 6.8: in the two cases shown in Figure 6.8(a), i must be in front of plane jdk ; in the two cases shown in Figure 6.8(b), i must be behind plane jdk .

There is no upper bound on the number of lines which can meet at the junction in the drawing. As we walk around the junction in a clockwise direction,

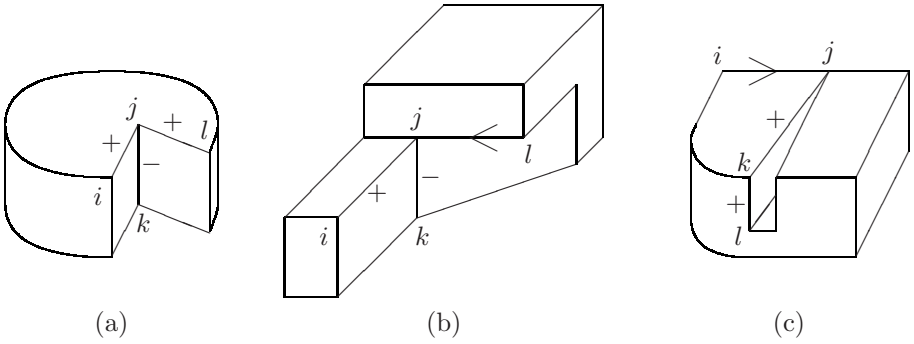


Figure 6.7: In (a) and (b) the fact that jk is a concave edge implies that i lies in front of plane $ijkl$. In (c) the convexity of edge jk implies that i lies behind plane $ijkl$.

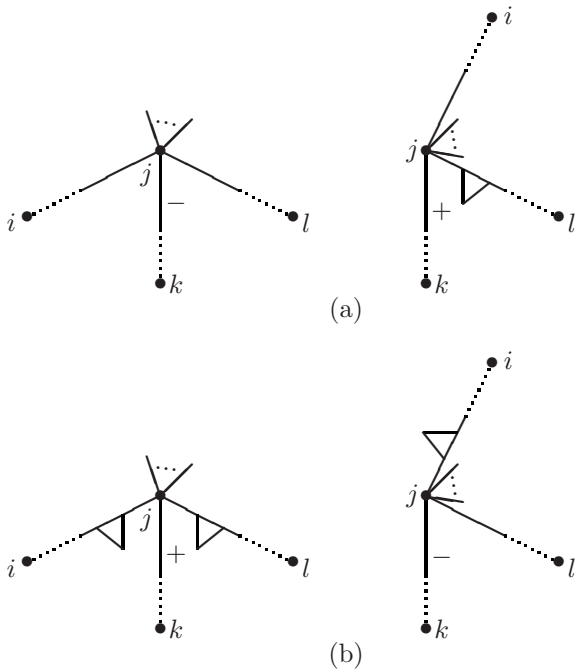


Figure 6.8: Constraints derived from a convex or concave label for a line end: (a) i is in front of plane $ijkl$; (b) i is behind plane $ijkl$.

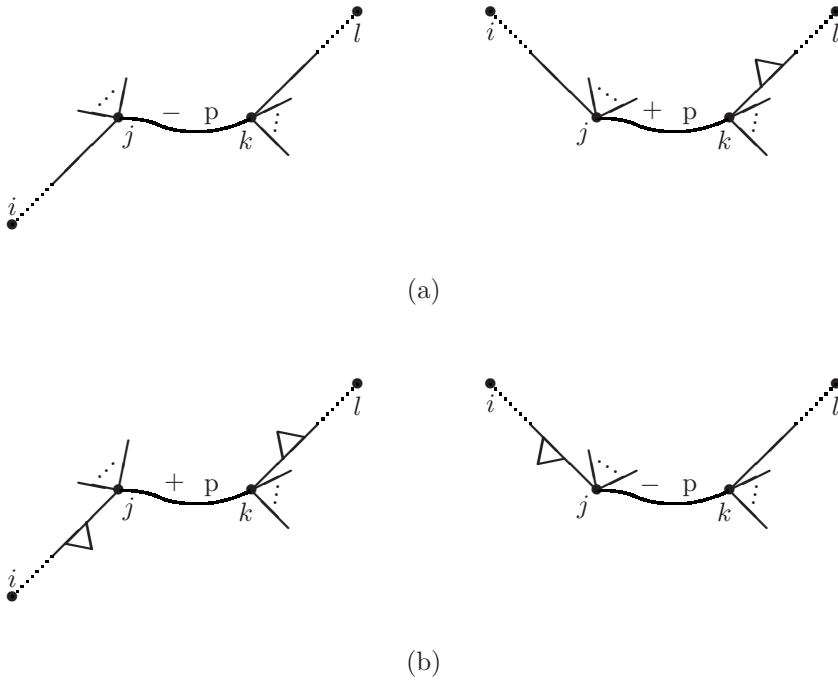


Figure 6.9: (a) i is in front of plane jdk ; (b) i is behind plane jdk .

starting from the projection of jk , the first line we encounter is the projection of the edge whose tangent is ji ; walking in an anticlockwise direction, the first line we encounter is the projection of the edge whose tangent is jl . The only restriction on the application of this constraint is that the projections of points i, j, l are not collinear. We use a downward-pointing triangle as a generic label, which, on a horizontal line, means that the edge which projects into this line lies on the surface which is visible just below the line. It thus represents any semantic label (e.g. $+$, $-$, \rightarrow , shadow [38], crack [173], reflection, surface mark, ramp line [32]) except an occluding edge with the occluding object above the line (\leftarrow). This label is not always necessary, as was illustrated by the example of Figure 6.7(b), in which i lies in front of plane jdk even when jl is an occluding edge.

To be able to deduce an inequality constraint from a junction, we require at least one line label ($+$ or $-$), together with the three points i, k, l (such as vertices or vanishing points), which are collinear with the tangents to three distinct edges meeting at vertex j . If the tangent to an edge at vertex j does not pass through a vertex or vanishing point, then an artificial point i (or k or l) can be created on this tangent.

To encode mathematically the fact that an edge is concave (or convex), we

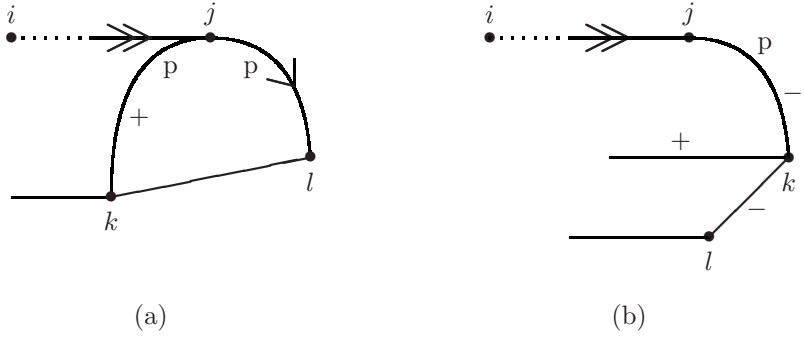


Figure 6.10: Inequality constraints at 3-tangent and curvature-L junctions: (a) i lies behind the plane of jdk ; (b) i lies in front of the plane of jdk .

require points i and l which lie on the tangent planes T_1 and T_2 to the two surfaces S_1 and S_2 which intersect to form the edge. If, for example, the first edge visible in the drawing when leaving jk in an anticlockwise direction is an edge which occludes surface S_2 (e.g. edge jl in Figure 6.7(b)), then we will have to look further afield to find a point l which actually lies on the tangent plane T_2 to S_2 . If S_2 is locally planar along jk (which is the case, for example, if jk is a straight edge), then we may be able to find such a point l , lying on T_2 , on the tangent to an edge at vertex k . An example is shown in Figure 6.7(c), where the convexity of edge jk implies that i is behind plane jdk . In generalizing this constraint, there are again four distinct cases to consider, depending on whether the edge jk is convex or concave and whether the projections of i and l lie on the same or opposite sides of the projection of the straight line in 3D space passing through points j and k . The four cases are shown in Figure 6.9. In the cases given in Figure 6.9(a), i must lie in front of plane jdk ; in Figure 6.9(b) i must lie behind plane jdk . The only restrictions on the application of this constraint are that the projections of j and k cannot be $T^>$ junctions and that neither i nor l can be collinear with the two points j and k .

As we have seen in Figure 6.9, it is possible to derive an inequality constraint for a curved convex/concave edge jk provided jk is a planar curve. This is true even when jk ends at a 3-tangent or curvature-L junction. Figure 6.10 gives two examples. In Figure 6.10(a), the convex label for jk implies that the 3D point i lies behind the tangent plane to the locally planar surface containing points j, k, l . We can consider this as a special case of the general constraint shown at the left of Figure 6.8(b). In Figure 6.10(b), the concave label for jk implies that i lies in front of the tangent plane to the locally planar surface bounded by edge jk . To find a third point l lying in this plane, we have to consider an edge kl leaving k rather than j . We can thus consider this as a special case of the general constraint shown at the left of Figure 6.9(a).

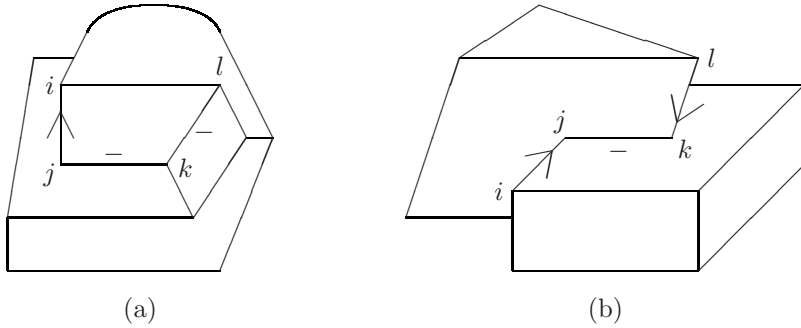


Figure 6.11: A junction pair in which points i , j , k , l must be coplanar; (b) a junction pair in which point i must lie in front of plane $ijkl$.

6.4.5 Coplanarity Constraints

Sugihara's classic work [154, 155] used an assumption of planarity of object surfaces to deduce information about the relative depth of scene points. Rather than making the restrictive assumption that objects are polyhedra, we make only very weak assumptions disallowing coincidences and improbable objects. We showed in the previous chapter that local planarity can, in fact, be deduced from these weak assumptions.

As in the previous section, we assume that a straight line L in a drawing is the projection of a straight edge E formed by the intersection of two surfaces which are planar in the vicinity of E . Similarly, every junction is assumed to be the projection of a vertex which behaves locally as a polyhedral vertex. Based on these assumptions, it is often possible to deduce a constraint concerning the relative positions in 3D space of E and other edges emanating from the vertices at either end of E . It is either an equality or inequality constraint depending on the junction labellings at the ends of L .

In the example in Figure 6.11(a), the occluding and concave edge labels indicate that the four points i , j , k , l all lie on the same face. The example in Figure 6.11(b) differs in that the occluding edge label for ji indicates that i is, in fact, in front of the face on which j , k , l lie. Although in the two examples given in Figure 6.11 ij and kl are straight edges, this is not necessarily the case. The edge leaving j in the direction of i (and the edge leaving k in the direction of l) may be curved. Points i and l may be vanishing points, vertices which just happen to lie on the tangent to the edge or artificial points (as described in Section 6.4.4) lying on the tangent to the edge.

Figure 6.12 shows a general coplanarity constraint. In this constraint, the projections of points i and l may lie on either side of the straight line passing through the projections of j and k . As in Section 6.4.4, the generic upward-pointing triangle label on a horizontal line represents any type of edge which lies on the surface projecting into the region above the line. This label for ij , jk and kl , together with the local planarity label 'p' for jk , indicates that points

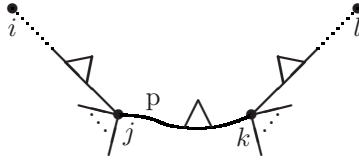


Figure 6.12: The coplanarity of i, j, k, l follows from the fact that jk is a straight edge.

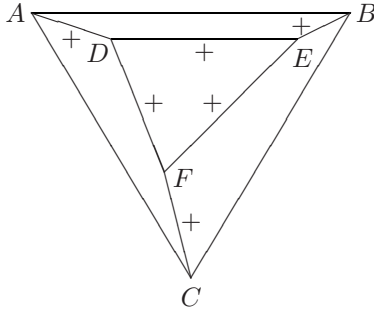


Figure 6.13: An impossible object since the lines AD, BE and CF , when extended, should meet at a point, which is not the case.

i, j, k, l are coplanar, since they all lie on the tangent plane to the surface which is visible above edge jk .

As an example of the use of the coplanarity constraint described above, consider the drawing in Figure 6.13 (adapted from [152]). Both sides of each straight line are locally planar by the straight-edge formation assumption. The coplanarity constraint applied to the straight line DE implies that points A, D, E, B are coplanar. Similarly, B, E, F, C are coplanar, as are C, F, D, A . However, it is impossible to find Z -coordinates for points A to F satisfying these coplanarity constraints. To see this, consider the three lines AD, BE and CF , which, when extended, should meet at a point, which is not the case. The coplanarity constraints coded as linear equations thus show that this is an impossible figure.

Certain coplanarity constraints are clearly redundant and need not be taken into account. For example, the constraint “ i, j, k, l coplanar” provides no information if $i = l$ or if i, j, l (or i, k, l) are known to be collinear. Similarly, an entirely visible planar face bounded by r edges can give rise to a coplanarity constraint on each set of four consecutive vertices. Since only $r - 3$ such constraints are necessary to establish the coplanarity of all vertices on the face, three of these constraints are redundant and can be ignored.

Figure 6.14 shows how the presence of occluding labels gives rise to an inequality constraint rather than an equality constraint. In Figure 6.14(a), the fact that ij is the tangent to an occluding edge means that i lies in front of the plane containing edge jk and tangential to the locally planar surface which is visible in the drawing above jk . Similarly, in Figure 6.14(b), both i and l

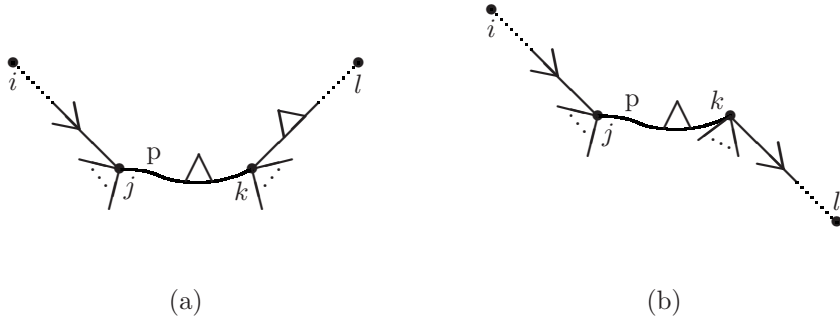


Figure 6.14: The presence of occluding labels imply that i lies in front of plane $ijkl$.

lie above this tangent plane. The resulting constraint is the same as for Figure 6.14(a), that i must lie in front of plane $ijkl$.

In Figure 6.14(a), the projections of i and l may lie either above or below the straight line passing through the projections of j and k , whereas in Figure 6.14(b) the projections of i and l must lie on opposite sides of this line. In both cases, the constraint does not apply if the projections of j or k are $T^>$ junctions. A reflected version of the constraint in Figure 6.14(a) also exists with the occluding edge on the right-hand side.

If objects have tangential edges and surfaces [34], then the constraint is less strong, in that i may be in front of *or on* plane $ijkl$.

In the constraints given in Figures 6.12 and 6.14, the projections of j and k need not be adjacent junctions. The constraints still hold even when there are any number of T junctions or 3-tangent junctions along the projection of edge jk in the drawing; we only require that the projection of jk be labelled ‘p’ and \triangle along its whole length.

6.4.6 Hidden-Surface Coplanarity Constraints

Under the more restrictive assumption of trihedral vertices, other constraints exist in which the plane on which points j, k, l lie is not a tangent plane to a visible surface, but to a hidden surface. An example is shown in Figure 6.15, where points i, j, k, l are coplanar: they all lie on the same hidden face.

Figure 6.15 is just one example of a very general constraint given in Figure 6.16. Points i, j, k, l must be coplanar (since they lie on the same tangent plane to the hidden surface) if the trihedral vertices j and k are joined by an occluding edge labelled ‘p’, as shown in the figure. Recall that the ‘p’ label written above jk refers to the hidden surface. The projections of j and k may be any Y or W junctions, and the projections of i and l may lie on any side of the straight line passing through the projections of j and k .

It is worth observing that the occluding edge label is inessential in the statement of this constraint; if edge jk has any other label, then i, j, k, l are still

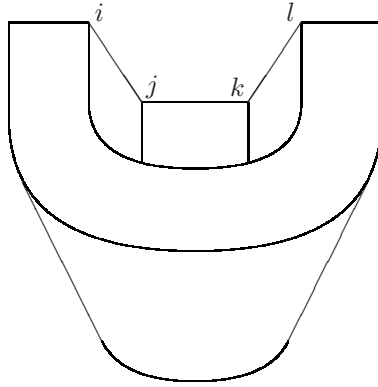


Figure 6.15: An example in which the coplanar points i, j, k, l lie on a hidden face. Vertices j and k are assumed to be trihedral.

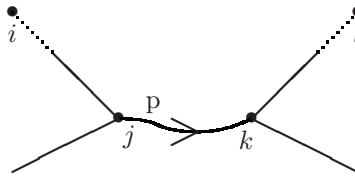


Figure 6.16: If j and k are projections of trihedral vertices, then i, j, k, l are coplanar.

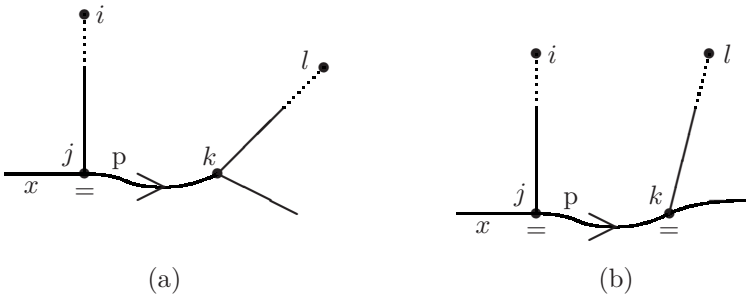


Figure 6.17: Point i lies in front of or on plane $ijkl$, since the continuation of ij passes behind the hidden locally planar surface.

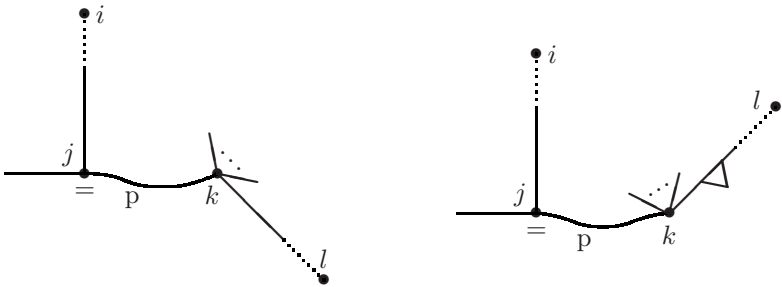


Figure 6.18: Point i must lie in front of plane $ijkl$, since the continuation of ij passes behind the visible locally planar surface.

coplanar by the constraint of Figure 6.12. This result follows from the catalogue of possible labellings for Y and W junctions as projections of trihedral vertices [20, 75] as given in Figure 3.2.

Hidden-surface constraints also arise when the bar of a T^- junction is joined to another junction by a line labelled ‘p’ since the continuation of the stem of the T^- junction must pass behind the hidden locally planar surface. In the two cases shown in Figure 6.17, i must lie in front of or on plane $ijkl$. Furthermore, if it is known that the label x is $-$, then the constraint is much stricter: i, j, k, l must be coplanar. In Figure 6.17(a), the projection of vertex k is any Y or W junction.

Although the continuation of line ij in Figure 6.18 may touch the hidden surface, it certainly passes behind the visible locally planar surface. Thus the constraint in this case is that i lies in front of plane $ijkl$. If tangential edges and surfaces are allowed, then the constraint is weaker: i lies in front of or on plane $ijkl$.

The projection of vertex k in Figure 6.18 can be any junction except a $T^>$ junction. Since vertices are assumed to be trihedral, at most three lines meet at this junction. The downward-pointing triangle label is only necessary when the projections of i and l lie on the same side of the straight line passing through the projections of j and k , as shown on the right-hand side of Figure 6.18.

In the constraints given in Figures 6.16, 6.17 and 6.18, the projections of j and k need not be adjacent junctions. The constraints still hold even when there are any number of T junctions or 3-tangent junctions along the projection of jk in the drawing. Reflected versions of the constraints in Figures 6.17 and 6.18 clearly also exist. Note that the constraints in Figure 6.17 and 6.18 do not hold if j can be a non-trihedral vertex, since in this case i could lie behind plane $ijkl$.

6.5 Orthographic Projection

Under orthographic projection, as opposed to perspective projection, the values to be determined are not the inverses $t_i = 1/Z_i$, but the distances Z_i themselves. It turns out that, to mathematically encode the three basic constraints (equidistance, collinearity, coplanarity) under orthographic projection, it is sufficient to replace t_i by Z_i in the equations given in Section 6.3. Thus, the mathematical formulation of the constraint that i and j must be equidistant from the projection plane is $Z_i = Z_j$; the constraint that i, j, k must be collinear is $d_{jk}Z_i + d_{ki}Z_j + d_{ij}Z_k = 0$; the constraint that h, i, j, k must be coplanar is $a_{ijk}Z_h + a_{jih}Z_k + a_{hik}Z_j + a_{jhk}Z_i = 0$. The coefficients d_{ij} and a_{ijk} are as defined in Section 6.3. Note that a_{ijk} is defined as $(v_i \wedge v_j) \cdot v_k$, where v_i represents $(x_i, y_i, 1)$.

In the inequality constraints there is a change of sign due to the fact that the inequalities concern the Z_i rather than their inverses t_i . Thus, the constraint that i must be nearer than j is $Z_i < Z_j$; the constraint that h must lie in front of plane ijk (where h, i, j, k are as shown in Figure 6.4(a)) is now $a_{ijk}Z_h + a_{jih}Z_k + a_{hik}Z_j + a_{jhk}Z_i > 0$.

Under orthographic projection, there are no vanishing points (and hence no vanishing-point constraint). Parallel lines in 3D project into parallel lines in a drawing, which can theoretically be detected. Let i, j, k, l be four points in 3D such that the projections of lines ij and kl are parallel in the drawing. Then under the GVA, the corresponding 3D lines are parallel, which we can write mathematically as

$$d_{kl}(Z_i - Z_j) - d_{ij}(Z_k - Z_l) = 0,$$

where d_{kl}, d_{ij} are defined as in Section 6.3. Note that this constraint can, in theory, be applied even when there is no line joining i and j (or k and l) in the drawing.

If we assign depth labels to straight lines, as proposed in Chapter 4, then a depth label on a line ij gives rise to a constraint of the form $Z_i > Z_j$. If we use the rich labelling scheme introduced in Chapter 5, then each gradient direction that has been determined gives rise to a linear constraint. Suppose that i, j, k are three non-collinear points in 3D such that the plane P passing through i, j, k has gradient direction α . Recall that $\tan \alpha = q/p$, where the equation of plane P is $Z = pX + qY + r$. Eliminating the parameters p, q, r leads to the linear constraint

$$b_{jk}Z_i + b_{ki}Z_j + b_{ij}Z_k = 0,$$

where $b_{ij} = (x_i - x_j) + (y_i - y_j) \tan \alpha$, with b_{jk} and b_{ki} defined similarly.

Unfortunately, the identification of a line as the projection of an orthogonal edge leads to a quadratic, rather than linear, constraint. Similarly, knowing that two edges are orthogonal leads to a quadratic constraint [117]. However, the presence of a cubic corner, a vertex h at which three orthogonal edges meet, can be translated into three linear constraints, provided we have the semantic labelling of the corresponding junction. Suppose, for example, that lines $ih,$

jh , kh meet at a $Y(+)$ junction. Let γ_{ij} denote the angle between lines ih and jh , with γ_{jk} and γ_{ki} defined similarly. Then, as observed in Section 4.6, the angle of inclination θ_{ih} of line ih to the image plane is given by $\tan^2 \theta_{ih} = \tan \gamma_{ki} \tan \gamma_{ij} - 1$. This gives us the linear constraint

$$Z_h - Z_i = D_{ih} \tan \theta_{ih},$$

where $D_{ih} = \sqrt{(x_h - x_i)^2 + (y_h - y_i)^2}$ is the length of ih in the image plane. Similar linear constraints exist for lines jh and kh .

All other constraints are identical to the perspective projection case. We will not discuss any further the case of orthographic projection since it produces a computational problem of exactly the same type as for perspective projection, namely a line-labelling problem together with a system of linear equations and inequalities.

6.6 Physical Realizability of Drawings

Sugihara [152, 155] showed that line-labelling constraints, together with linear constraints, provide necessary and sufficient conditions for a drawing of a polyhedral scene to be physically realizable. In [36] we showed that line-labelling constraints alone provide necessary and sufficient conditions for a drawing to be physically realizable under the assumptions that object faces are arbitrary C^3 curved surfaces, that vertices are trihedral and that all lines (even straight lines) can be projections of arbitrary C^3 curves. The linear constraints described in this chapter complement the line-labelling constraints for curved objects [36, 109]. They correct the rather unreasonable assumption that a straight line can be the projection of a curved edge.

It is straightforward to modify the proof given in [36] to show that the line-labelling constraints together with the linear constraints given in this chapter provide necessary and sufficient conditions for the physical realizability of a drawing. They are clearly necessary. Given a legal line labelling and the positions (Z_1, \dots, Z_n) of all points projecting into junctions in a drawing, a 3D scene whose projection is identical to the drawing can be constructed as in [36] thanks to the large freedom of choice in the shape of object faces and those edges which do not project into straight lines.

The linear constraints presented in this chapter were inspired by Sugihara's [154] constraints for polyhedral objects, but they go further in that they include constraints on collinear points, parallel lines and hidden faces. Most importantly, they can be applied without the restrictive assumption of planar surfaces.

The notion of impossible figure depends on the assumptions we make concerning the objects depicted. For example, the drawing in Figure 6.19 is a legal projection of a polyhedral object. However, if we impose the extra condition that all vertices must be trihedral (i.e. formed by the intersection of faces lying in only three planes), then this is an impossible object. Assuming that the object is a polyhedron and that B and C are trihedral, vertices A , B , C , D

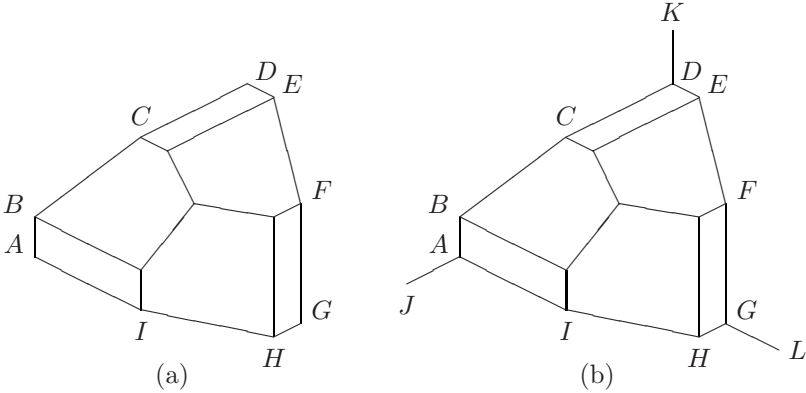


Figure 6.19: A drawing which cannot be the projection of a polyhedron with trihedral vertices.

must all lie on the same hidden planar face. Similarly, vertices A, I, H, G are coplanar. In Figure 6.19(b) we have added an edge leaving vertex A and parallel to CD and to HG . Since the projections of JA, CD and HG are parallel in the drawing, by the GVA we can deduce that JA, CD and HG are parallel in 3D. It follows that JA lies on both of the planes $ABCD$ and $AIHG$, and hence lies on their intersection. A similar argument tells us that DK lies on the intersection of planes $ABCD, DEFG$ and that GL lies on the intersection of planes $DEFG, AIHG$. But then lines JA, DK and GL should intersect at a point in 3D, which is clearly impossible, since their projections in the drawing do not intersect.

6.7 The Computational Problem

The constraints described in Section 6.4 when applied to a labelled drawing produce a set of linear equations

$$At = 0 \tag{6.9}$$

and a set of linear inequalities

$$\begin{aligned} Bt &> 0, \\ Ct &\geq 0, \end{aligned} \tag{6.10}$$

where $t = (t_1, \dots, t_n)$.

Let $d = (\delta, \dots, \delta)$, where δ is any strictly positive constant. The system of Equations (6.9) and Inequalities (6.10) has a solution iff the following set of inequalities, together with (6.9), has a solution:

$$\begin{aligned} Bt &\geq d, \\ Ct &\geq 0. \end{aligned} \tag{6.11}$$

This result follows from the size/distance ambiguity inherent in any drawing, meaning that any solution (t_1, \dots, t_n) to (6.9) and (6.10) can simply be scaled up uniformly so as to satisfy (6.11). Thus, the linear constraints applied to a labelled line drawing produce a standard linear programming problem.

Suppose that, by applying the constraints described in Section 6.4 to a drawing, we have deduced that points A, B, C, D are coplanar, A, B, E are collinear and B, C, D, E are coplanar. The third of these constraints is redundant since it could be logically deduced from the other two. In the presence of such linear dependence between equations in system (6.9), any errors in the position of junctions in the drawing will mean that the system of equations (6.9) has no solution. This phenomenon is known as *superstrictness*. This is an important problem, since errors are inevitable due to the limited precision of floating-point numbers stored in a computer.

One solution is a scheme in which the combined system of equations (6.9) and inequalities (6.11) is replaced by the inequalities

$$\begin{aligned} e &\geq At \geq -e, \\ Bt &\geq d - e', \\ Ct &\geq -e'', \end{aligned} \tag{6.12}$$

where $e = (\epsilon_1, \dots, \epsilon_n)$, $e' = (\epsilon'_1, \dots, \epsilon'_n)$, $e'' = (\epsilon''_1, \dots, \epsilon''_n)$ for some small $\epsilon_i, \epsilon'_i, \epsilon''_i > 0$. It is clearly essential to choose $\delta > \epsilon'_i$ (for at least one i), otherwise system (6.12) has a trivial solution in which all points are coplanar. The values chosen for $\epsilon_i, \epsilon'_i, \epsilon''_i$ determine tolerances in the linear constraints, for example, the extent to which a supposedly planar surface can actually be curved (i.e. the distance of a vertex from the plane on which it is supposed to lie). The value chosen for δ , on the other hand, imposes a lower bound on the distance of a vertex V from a plane P , when it is known that V lies in front of (or behind) P . To avoid physically unrealizable drawings being accepted as realizable, due to the tolerances $\epsilon_i, \epsilon'_i, \epsilon''_i$, we recommend choosing δ to be an order of magnitude greater than all of $\epsilon_i, \epsilon'_i, \epsilon''_i$.

Several different solutions to the superstrictness problem have been proposed. Sugihara [152, 155] deletes picture points until what remains is a maximally generically reconstructible substructure. This corresponds to finding a maximal linearly independent subset of Equation (6.9); a solution t to this subsystem is then used to determine whether corrections in the positions (x_i, y_i) of junctions in the drawing can be made so that the whole system (6.9) has a solution. There is an $O(n^3)$ algorithm to find a maximally generically reconstructible substructure [78], but there are three disadvantages to Sugihara's approach. Firstly, the 'corrected' drawing may not have the same basic structure as the original drawing if the coordinates of picture points are significantly altered. Secondly, not all points are treated equally. Thirdly, the correction process is not possible if one of the deleted vertices lies on more than three non-triangular faces (see [176] or [135] for examples of such drawings).

Superstrictness occurs when there are linear dependencies between the equations in the linear system. One example is when two faces meet along two

distinct edges E_1, E_2 . The projections of E_1, E_2 must be collinear in the picture. Such a linear dependence is known as a *second-order syzygy*. Linear dependencies may also exist between these second-order syzygies, giving rise to what are known as third-order syzygies. Heyden [73] proposed a scheme for correcting the coordinates of junctions in a superstrict picture which treats all points equally. Unfortunately, as for Sugihara's technique, his scheme does not work if there are third-order syzygies.

Ros and Thomas [135] use an incomplete local search algorithm to find a corrected version of an incorrect drawing which minimizes the sum of the squares of the 2D distances of junctions to their equivalents in the original drawing. Their scheme can correct all correctable drawings of opaque objects and treats all points equally, but it is not guaranteed to converge to the global optimum.

By introducing new constraints compared with Sugihara (concerning parallel lines and collinearity), potential redundancy in system (6.9) is such that the number of equations may be greater than the total number of variables (even when all the values of x_i, y_i, t_i are treated as variables). Thus, although the techniques of Sugihara, Heyden or Ros and Thomas could be used to correct certain drawings, the inability to find a correction would not imply the physical impossibility of a drawing.

Whichever technique is used to solve the linear programming problem given by (6.9) and (6.11), this is only part of the problem. Such a linear programming problem may have to be solved for each legal labelling of a drawing, of which there may be an exponential number. Even determining whether a line drawing of a polyhedron with trihedral vertices has a single legal semantic labelling is unfortunately an NP-complete problem [92]. The straight line labelling constraint, which prohibits label transitions on straight lines in a drawing, implies that the labelling problem for curved objects with trihedral vertices is also NP-complete, since it contains the labelling problem for drawings of polyhedral scenes as a subproblem.

In fact, it was shown in [36] that the realizability problem (the problem of deciding whether a drawing is realizable as the projection of a 3D scene) for drawings of curved objects with trihedral vertices is NP-complete under either perspective or orthographic projection due to the combination of junction-labelling constraints and constraints derived from the presence of parallel lines in 3D (Theorem 9.18). The introduction of additional constraints in this chapter does not change this result; in the constructions for the proof of NP-completeness in [36] it is sufficient to avoid the presence of collinear points and straight lines so that none of the additional constraints applies.

Nevertheless, the local propagation of line labels is a powerful tool in eliminating illegal labels for line ends. Furthermore, we have presented the linear constraints of Section 6.4 in such a way that they can be applied with minimal knowledge of actual line labels. An incomplete algorithm, although very effective on all the example drawings given in this chapter, is to propagate line labels through local consistency operations until convergence [55, 160], build the corresponding linear constraints and solve the resulting linear programming problem.

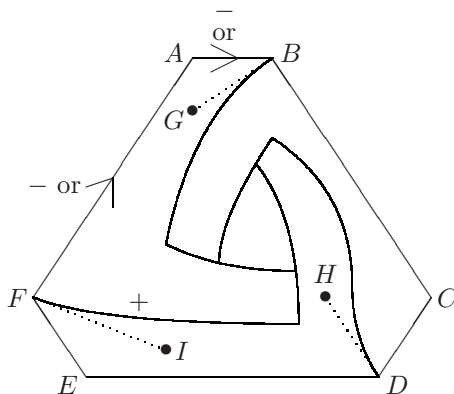


Figure 6.20: A figure which is discovered to be impossible by the propagation of junction-labelling constraints for trihedral vertices followed by the application of the coplanarity constraint.

As an example, consider the drawing in Figure 6.20, which is a curved version of the well-known Penrose triangle. Under the assumption of trihedral vertices, the catalogue of legal junction labellings [32, 109] provides sufficient information about line labels to deduce that I, F, A, B are coplanar and F, A, B, G are coplanar, by two applications of the coplanarity constraint given in Figure 6.12. Thus, I, F, A, B, G are coplanar. By symmetry, G, B, C, D, H are coplanar and H, D, E, F, I are coplanar. These three planes should then intersect at a point in 3D space, which implies that the projections of BG, DH and FI , when extended, should meet at a point in the drawing. This is clearly not the case. Since the corresponding linear programming problem has no solution, we can deduce that this is an impossible figure.

This example illustrates the utility of creating artificial points (G, H, I in Figure 6.20) on tangents to edges. Note that edges FA and AB are not assigned a unique label after local propagation of constraints; the coplanarity constraint can nevertheless still be applied since ‘ \rightarrow ’ and ‘ $-$ ’ both imply that the edge lies on the surface visible below the line.

6.8 Conclusion

The constraints stated in this chapter represent a generalization of known constraints for polyhedral objects to the class of objects with curved surfaces. These constraints are capable of identifying classic examples of impossible figures involving polyhedra but also many more new examples of impossible figures involving curved objects. Instead of assuming planarity of surfaces, the constraints allow us to deduce planarity from labelled pairs of junctions joined by a straight line.

Although differentiating between possible and impossible figures is

theoretically an NP-complete problem, polynomial-time algorithms for the propagation of semantic labels, together with the linear programming approach proposed in this chapter, provide a practical (though incomplete) test for the physical realizability of figures.

The successful integration of information from various sources (junctions, collinear points, straight lines, vanishing points), in order to recover the depth of object vertices, can be considered as an essential first step towards a complete computer vision system which would also incorporate information from other depth-recovery techniques such as shape-from-shading [74, 110] and stereovision [64]. The constraints presented in this chapter are based on the assumption that a given drawing is a perfect projection of a 3D scene. They have direct applications in the reconstruction of a 3D scene from a user-entered drawing.

Chapter 7

Wireframe Projections

The reconstruction of an object from a single 2D projection of a 3D wireframe model is a vision problem with applications in CAD/CAM and computer graphics. This chapter describes a technique for the interpretation of wireframe projections based on assigning semantic and numerical depth labels to lines (first published in [42]). This method allows us to state necessary and sufficient conditions for the physical realizability of a wireframe projection of a curved object. The presence of linear features provides further constraints on the positions of object vertices. For example, each straight line gives rise to a coplanarity constraint between a set of object vertices. We show that extra information, such as vanishing points, parallel lines or user-entered depth-parity information, is sufficient to uniquely determine the face circuits in wireframe projections of polyhedra with simple trihedral vertices. In fact, a polyhedron with simple trihedral vertices can be unambiguously reconstructed from its 3D wireframe model.

From a more practical point of view, we give junction constraints for wireframe projections of curved objects with tetrahedral vertices or with edges and surfaces which can meet tangentially. We also generalize the constraints between distant lines, given in Chapter 3, and the rich-labelling constraints, given in Chapter 5, to wireframe projections of curved objects.

7.1 Introduction

The interpretation of line drawings of opaque objects is a classic problem in computer vision. Necessary and sufficient conditions have been given for the physical realizability of line drawings of polyhedral objects [154, 155] and for curved objects with C^3 surfaces [36]. The computational complexity of testing the realizability of a line drawing has also been extensively studied (see Chapter 9). Although the most general problem is NP-complete [92, 36, 38], the problem is solvable in linear time under certain restrictions on the drawings (knowledge of all vanishing points [127, 36], absence of straight lines [34, 36] or possibility

of contrast failure between parallel surfaces [38]).

In a line drawing of opaque objects, only visible lines are shown. This is appropriate for traditional vision applications. However, for the reconstruction of the 3D shape of a human-entered object-model, a more appropriate input is a line drawing in which both visible and hidden lines are shown. Let the depth of a 3D edge E denote the number of surfaces lying between E and the viewpoint. Visible lines represent edges of depth zero. Sugihara [155] and Alevizos [5] have studied the interpretation of line drawings in which solid lines represent edges of even depth and broken lines represent edges of odd depth. A wireframe projection is a projection of all 3D edges (visible or not) in which no information concerning the depth of lines is explicitly given. When extra information is provided in the form of the 3D coordinates of each object vertex, we call this a 3D wireframe model. The reconstruction of an object from several wireframe projections has been extensively studied [95, 175], as has the problem of converting a 3D wireframe model into a surface-based 3D model [4, 82, 99, 103, 112, 147, 170, 171]. Although the reconstruction of a 3D object from a user-entered wireframe is the most studied problem, other interesting applications exist, such as the indexing of technical line-drawing databases [157].

In this chapter we study the interpretation of wireframe projections. We give necessary and sufficient conditions for the physical realizability of a wireframe projection when no depth information is given (in terms of depth of lines or 3D coordinates of vertices). These necessary and sufficient conditions involve not only semantic labels (convex, concave, occluding, extremal), first introduced by Huffman [75] and Clowes [20] and generalized to wireframe projections by Sugihara [151, 155], but also numerical labels representing the number of surfaces in front of and behind the corresponding 3D edge. Labelling lines by their depth was first suggested by Huffman [75], one of the pioneers in this field.

We firstly restrict our study to wireframe projections from a general viewpoint of objects with C^3 edges and surfaces meeting non-tangentially at trihedral vertices. Later we consider a more general scheme allowing objects with discontinuities of surface curvature [32, 34] or tetrahedral vertices [165]. Figures 2.16(a)–(d) show three drawings which require numerical depth labels as well as semantic labels to demonstrate their non-realizability. Although a consistent labelling exists in terms of semantic labels alone, they are clearly spatially incoherent.

If the drawing contains linear features such as straight lines, parallel lines or collinear lines, then further constraints can be deduced concerning the 3D positions of object vertices. As we saw in Chapter 6, these constraints give rise to linear equations or linear inequalities between the depths z_i of object vertices (in the case of orthographic projection) or between the inverses $t_i = 1/z_i$ of these depths (in the case of perspective projection). In a standard technique, pioneered by Sugihara [154, 155], a drawing is physically realizable if and only if the resulting linear programming problem has a solution.

Figure 2.16(e) shows an example of a drawing which has a legal labelling (both semantic and numerical) but which is impossible by linear constraints. We assume a general viewpoint and orthographic projection, which imply that

parallel lines in the drawing are projections of parallel lines in 3D. The existence of a legal labelling follows from the fact that this drawing can be transformed into a drawing of a cube by changes to positions of junctions that do not alter the configuration of the drawing. Coplanarity constraints deduced from assumptions about the formation of straight edges imply that J, K, L, M are coplanar. However, using the constraint that parallel lines in the drawing are projections of parallel 3D edges, it is possible to deduce that K does not lie in the same plane as J, L, M .

In the machine reconstruction of a 3D object model from a human-entered drawing, it is essential that the drawing be unambiguous. Wireframe models are often said to be ambiguous (see, for example, Mortenson [118]) because they do not contain surface information (as in B-rep models) or volume information (as in CSG models). However, we will show that wireframe models of polyhedra containing only simple trihedral vertices are, in fact, unambiguous.

7.2 Semantic and Numerical Line Labels

We make the following simplifying assumptions:

1. Objects are regular solids bounded by C^3 surfaces separated by C^3 edges (discontinuities of surface orientation).
2. Object vertices are trihedral, i.e. formed by the intersection of three surfaces. The edges and surfaces meeting at a vertex meet non-tangentially.
3. If the scene contains more than one object, then the objects are in a general relative position: a small perturbation in their relative position does not alter the configuration of the drawing (such as the presence of junctions, straight lines or parallel lines).
4. The drawing is a projection of object edges (including viewpoint-dependent edges such as the side of a cylinder) from a general viewpoint, i.e. a small perturbation in the viewpoint position does not alter the configuration of the drawing.

These assumptions exclude certain classes of interesting drawings, such as those involving objects with smooth edges or non-trihedral vertices. The basic preliminary results obtained under these simplifying assumptions will serve as a foundation for work on more complex objects presented later in this chapter in Sections 7.11–7.13.

Each line L joining two junctions in the drawing can be assigned

1. A semantic label from the six possible labels $+, -, \leftarrow, \rightarrow, \Leftarrow, \Rightarrow$ according to the form of the 3D edge E projecting into L ;
2. A pair of numerical labels m, n representing the number of surfaces m lying between the viewpoint and E and the number of surfaces n lying behind E .

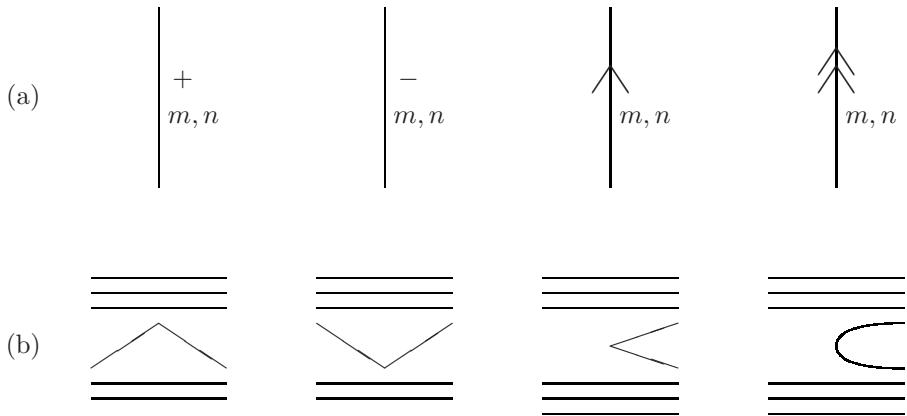


Figure 7.1: (a) The four distinct line labels; (b) cross-sections of the corresponding 3D edges.

The meaning of the semantic labels $+$, $-$, \rightarrow , \Rightarrow is demonstrated by their cross-sections shown in Figure 7.1. In the example cross-sections of Figure 7.1(b), $m = 3$ and $n = 2$ for the lines labelled ' $+$ ' and ' $-$ '; $m = 3$ and $n = 3$ for the lines labelled ' \rightarrow ' and ' \Rightarrow '. The label ' $+$ ' means that the two surfaces meeting at E subtend an angle greater than π when observed from the viewpoint. The label ' $-$ ' means that this angle is less than π .

We say that an edge E is visible if there is no surface lying between E and the viewpoint. An edge which is not visible is called hidden. A 3D edge E is called convex (concave) if the two object faces which intersect to form E subtend an angle less than π (greater than π) in the interior of the object. If a line L labelled ' $+$ ' (' $-$ ') is the projection of a visible edge E , then E is a convex (concave) edge. Note that this is not necessarily the case for hidden edges. In fact, the label $(+, m, n)$ represents a convex edge if m is even and a concave edge if m is odd. Similarly, the label $(-, m, n)$ represents a concave edge if m is even and a convex edge if m is odd.

The semantic label \rightarrow means that the corresponding edge is the intersection of two object faces both of which project onto the right-hand side of the line as we follow the direction of the arrow. The label \rightarrow represents a surface-normal discontinuity, i.e. the intersection of two non-tangential object surfaces, whereas the label \Rightarrow represents a viewpoint-dependent edge (extremal edge) which is the locus of points at which the line of sight is tangential to an object surface. Again the object surface projects onto the right-hand side of the line as we follow the direction of the arrow. For example, a sphere projects into a circle labelled \Rightarrow . Note that, purely for typographical reasons, extremal edges are labelled by a double-headed arrow in the figures but by the symbol \Rightarrow in the text.

Figures 7.2(a) and (c) show two wireframe drawings of objects with holes. Figures 7.2(b) and (d) are drawings of the corresponding 3D solid opaque objects in which only visible edges are shown. Note that line AB in Figure 7.2(a) and

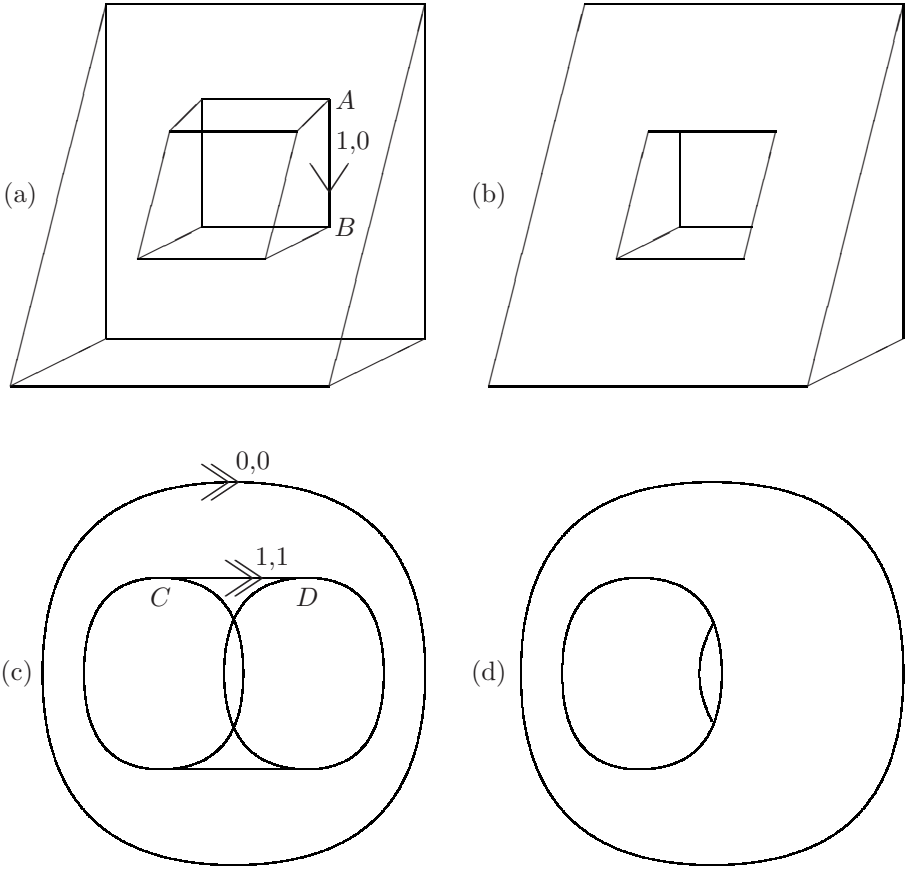


Figure 7.2: (a),(c) Two wireframe drawings of the 3D opaque objects illustrated in (b),(d), respectively.

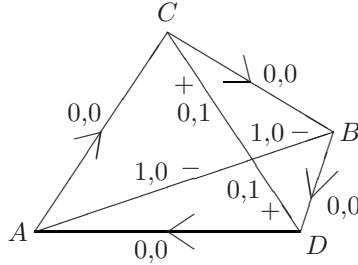


Figure 7.3: A simple wireframe projection labelled with both semantic and numerical labels.

line CD in Figure 7.2(c) both represent boundaries of holes and it is the hole which lies to the right of the line as we follow the direction of the arrow.

Figure 7.3 shows a simple wireframe projection with a legal labelling of each line segment. According to the labelling of Figure 7.3, we can deduce that the 3D edge CD lies in front of the 3D edge AB . Note that line AB in Figure 7.3(b) is labelled ‘-’ even though it is the projection of a convex 3D edge since its depth $m = 1$ is odd. Junctions in a wireframe projection can be classified as follows. Let J be a junction at which three lines L_1, L_2, L_3 meet non-tangentially and let E_1, E_2, E_3 be the 3D edges which project into L_1, L_2, L_3 . Assume that the lines are numbered so that the angles between L_1 and L_2 and between L_2 and L_3 are both less than π . If the angle between L_3 and L_1 is greater than π , then J is a W junction, otherwise J is a Y junction. If E_1 lies behind the plane of E_2, E_3 , then J is classified as a W(+) or Y(+) junction, otherwise a W(-) or Y(-) junction. When two lines cross, they form an X junction. At 2-tangent and 3-tangent junctions, two or three lines meet tangentially.

Figure 7.4 shows a labelled wireframe projection involving curved lines. In this figure, junctions have been identified by their junction-type (2-tangent, 3-tangent, W(+), W(-), etc.). Although a labelled wireframe projection is still ambiguous, since we do not know the depth of any point on any object surface, the labels provide valuable local shape information at each edge and vertex as well as topological information concerning object faces.

7.3 Realizability

When three surfaces meet non-tangentially at a point to form a 3D vertex, their tangents divide space into eight octants. Each octant may be empty or filled with matter. Figure 3.1 shows the six possible trihedral vertices thus obtained by this classic technique [20, 75]. (There is, in fact, a seventh vertex which is not shown since it is simply a reflected version of vertex C). The viewpoint may be situated in any of the eight octants, including those filled with matter. By exhaustion, we obtained the list of all semantically and numerically labelled projections of the vertices in Figure 3.1 from all possible viewpoints. For example, vertex B in

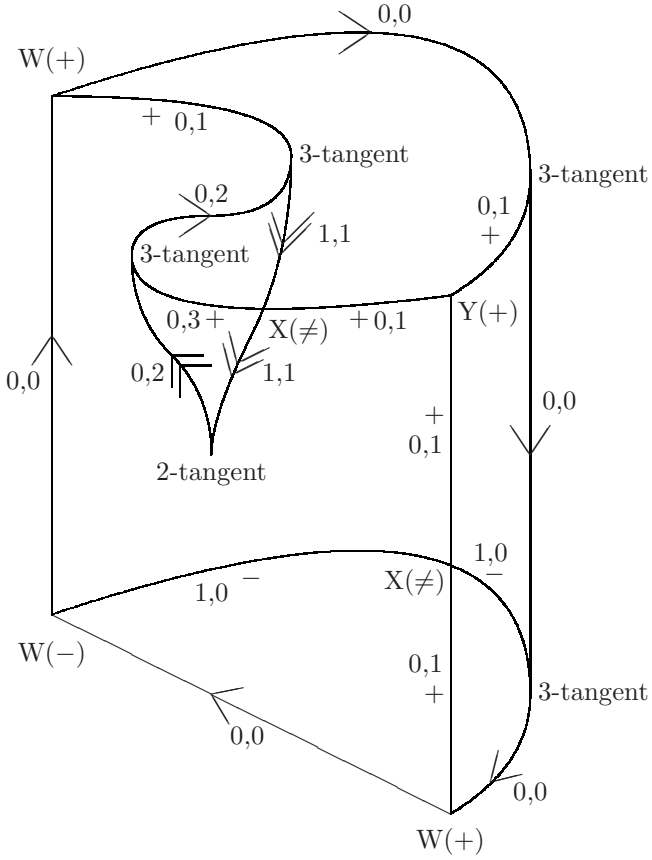


Figure 7.4: An example of a labelled wireframe projection.

Figure 3.1 projects into the eight labelled junctions of Figure 7.5(a). To obtain the numerical labels, we assume that any number of object surfaces can lie in front of or behind the vertex. Thus the numbers m, n are arbitrary non-negative integers. A physical realization of each of these eight labellings is shown in the wireframe projection of a holed cube shown in Figure 7.5(b). The complete catalogue of labelled junctions representing projections of trihedral vertices is given in Figures 7.6 and 7.7 under the headings $W(+)$, $W(-)$, $Y(+)$, $Y(-)$, $X(=)$, snowflake.

Even if the drawing contains only a single object, one edge may pass in front of another without intersecting it in 3D space. The resulting junction is known as an $X(\neq)$ junction (as opposed to an $X(=)$ junction, which is the projection of a vertex at which two edges intersect, such as vertex C in Figure 3.1). An example of an $X(\neq)$ junction is shown in Figure 7.4. We call the projection of vertex D in Figure 3.1 a snowflake junction. Note that certain workers have chosen not to include the vertices which project into $X(=)$ and snowflake junctions in the list of possible vertices.

The distinction between $W(+)$ and $W(-)$ junctions and between $Y(+)$ and $Y(-)$ junctions was first made by Parodi and Torre [127]. We can classify, for example, a W junction as $W(+)$ or $W(-)$ if we have information about the relative directions of the corresponding 3D edges, obtained from vanishing points. If such information is not available, then the set of labellings for a W junction is simply the union of the sets of labellings for $W(+)$ and $W(-)$ junctions. Purely for compactness of presentation, we have omitted from the list of legal labellings for Y , X and snowflake junctions those labellings that can be obtained by a simple rotation of those given in Figures 7.6 and 7.7. After incorporating these rotated versions, a $Y(+)$ junction, for example, has exactly the same number of labellings as a $W(+)$ junction.

Finally, curved surfaces give rise to viewpoint-dependent junctions involving projections of extremal edges. Examples of the resulting types of junction, known as 2-tangent and 3-tangent, are illustrated in Figure 7.4. The legal labellings of 2-tangent and 3-tangent junctions are listed in Figure 7.8. At a 3-tangent junction J , two of the lines meeting at the junction form a C^3 curve L which passes through J , whereas the third line L' terminates at J and exhibits a discontinuity of curvature with L . Line L' is shown as a horizontal straight line in Figure 7.8 but may be curved [32, 109]. Again purely for compactness of presentation, we have also omitted a reflected version of the 3-tangent junction in which line L' goes off to the right instead of to the left. In the catalogue of labelled junctions in Figures 7.6–7.8, the numbers m and n are non-negative integers. A further unary constraint exists on each line label (s, m, n) . This unary constraint, which follows directly from the reasonable assumption that the viewpoint lies outside all objects and that no object is of infinite extent, can be stated as follows:

Parity constraint: If $s = '+'$ or $'-'$, then $m + n$ is odd; if $s = '→'$ or $'⇒'$, then $m + n$ is even.

If we assume that the wireframe projection is a projection of the whole 3D

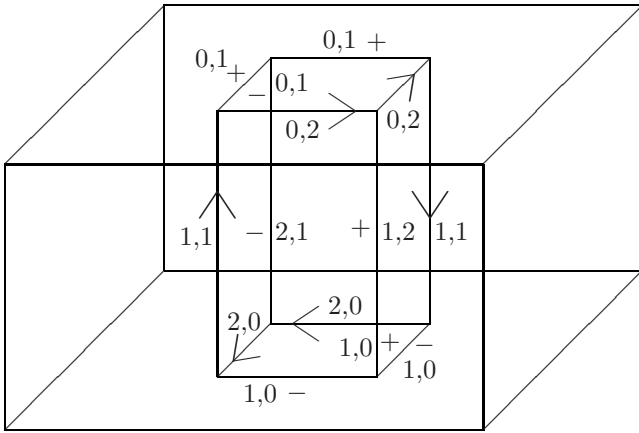
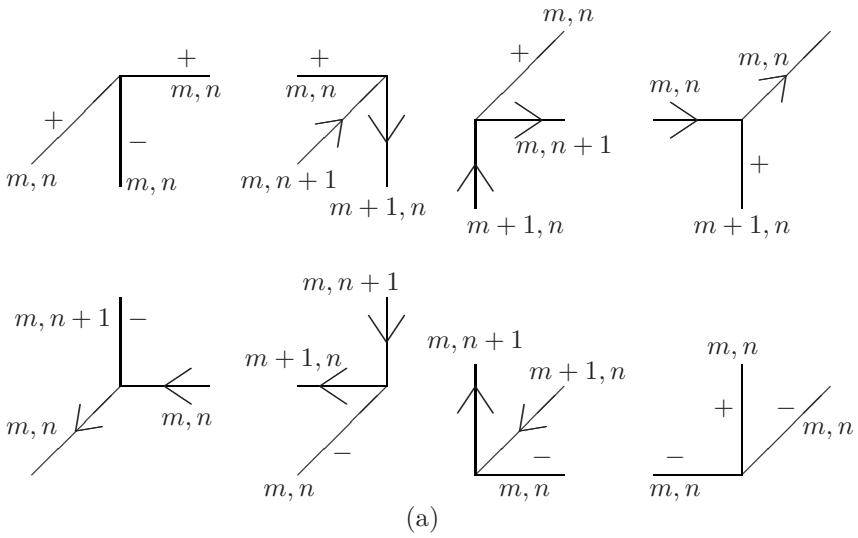


Figure 7.5: (a) The labelled junctions which can occur in a wireframe projection of vertex B in Figure 3.1; (b) a wireframe projection (of an object with a vertical hole) demonstrating the physical realizability of each of these labellings.

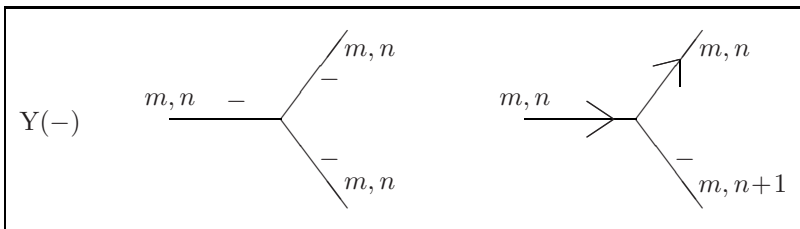
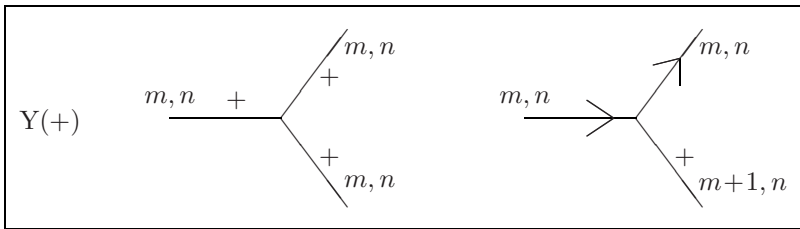
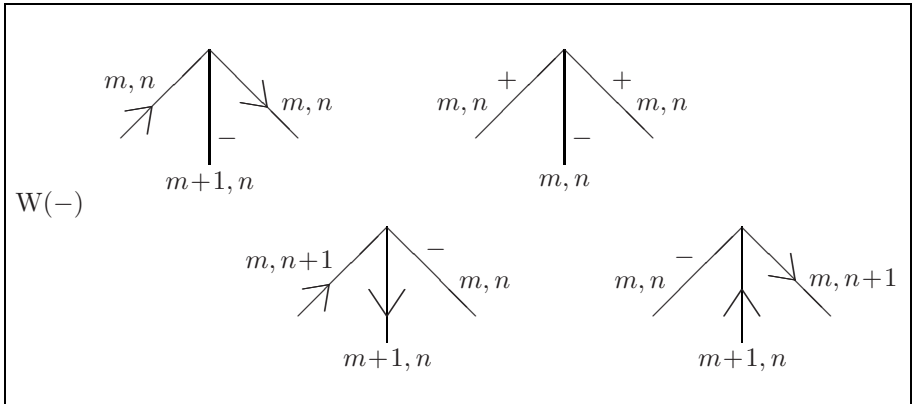
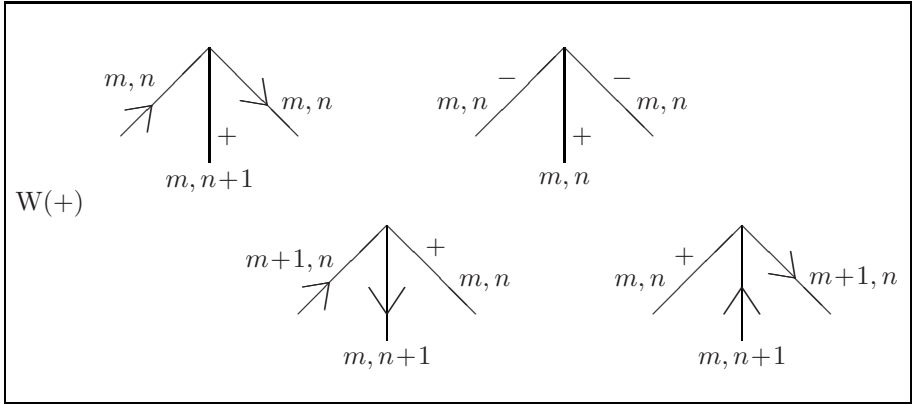


Figure 7.6: Catalogue of labelled W and Y junctions in wireframe projections.

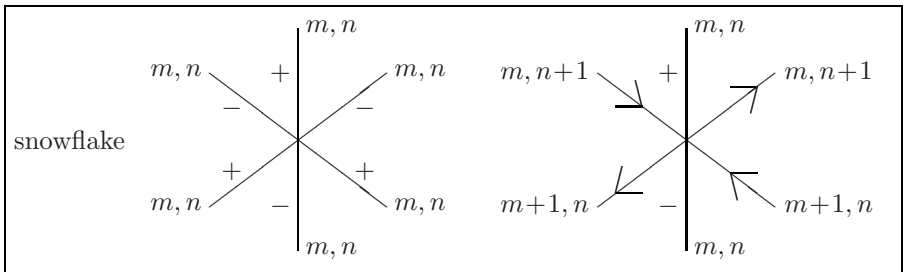
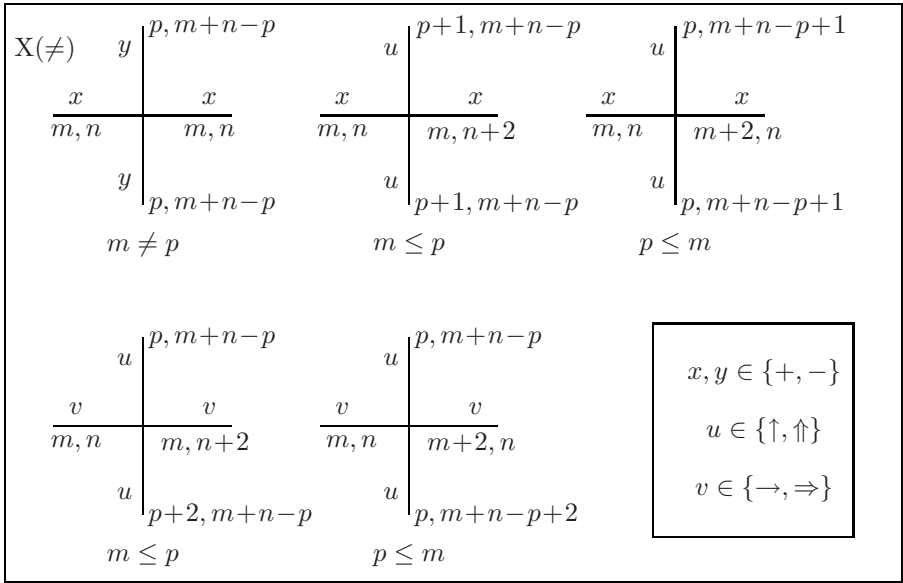
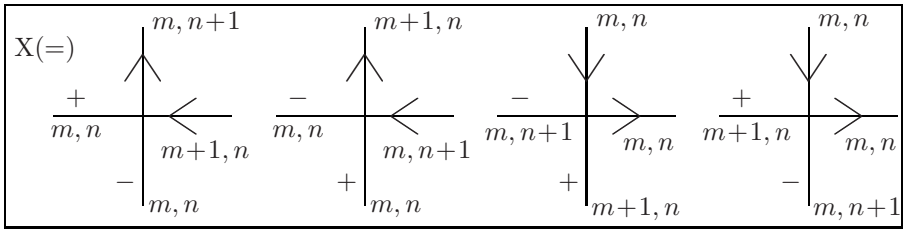


Figure 7.7: Catalogue of labelled X and snowflake junctions in wireframe projections.

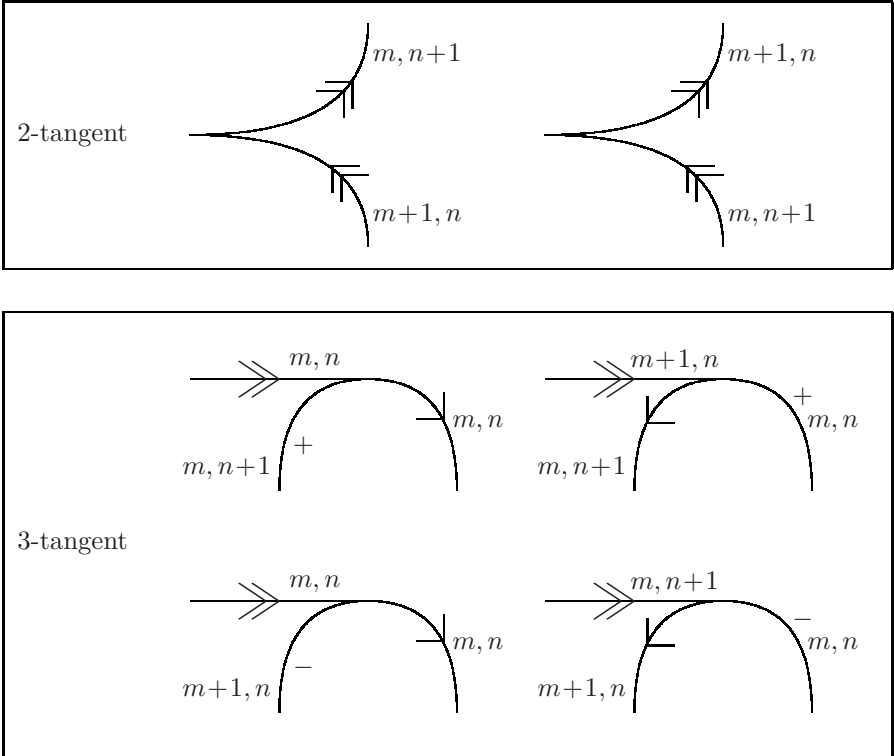


Figure 7.8: Catalogue of labelled 2-tangent and 3-tangent junctions in wireframe projections.

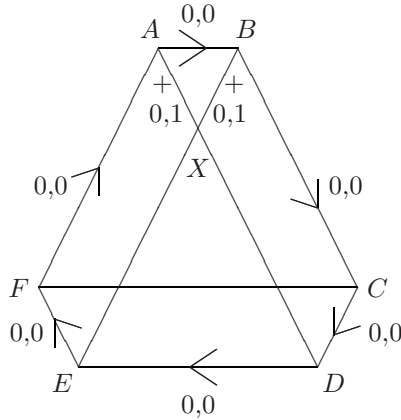


Figure 7.9: An example of an impossible wireframe projection.

scene, in the sense that all 3D edges and surfaces are completely visible and not clipped by the picture boundary, then we have the following stronger constraint.

Outer-boundary constraint: all lines on the outer boundary of the wireframe projection are labelled $(\uparrow, 0, 0)$ or $(\uparrow, 0, 0)$, since they correspond to occluding edges of objects.

As an example of the strength of these constraints, consider the wireframe projection in Figure 7.9. Its boundary has been labelled in accordance with the outer-boundary constraint. From the list of labellings for W junctions, we see that each of the lines AD, BE, CF must have the label $(+, 0, 1)$ or $(-, 1, 0)$, and hence two of these lines must have the same label. Without loss of generality, suppose that lines AD and BE are both labelled $(+, 0, 1)$. But then the labelling of junction X is illegal according to the catalogue of Figure 7.7. Similarly, each of the wireframe projections in Figure 2.16(a),(c),(d) are also physically impossible, since none of them has a semantic and numerical labelling satisfying the above constraints. There is also another constraint which is necessary when a wireframe projection can be decomposed into several connected components. To state this new constraint, we require the following definition.

Definition 7.1 *In a wireframe projection (considered as a planar graph G), let R be a connected region (i.e. face of G) containing holes H_1, \dots, H_g (i.e. connected components of G entirely surrounded by R). The frontier of R is defined as the list of line segments forming the outer perimeter of R (visited in clockwise order) together with the lists of line segments forming the perimeters of H_1, \dots, H_g (each visited in anticlockwise order).*

Region constraint: If line segments L_1, L_2 form part of the frontier of the same connected region R , then $f_R(L_1) = f_R(L_2)$, where $f_R(L)$ is the number of surfaces projecting into the region to the right of the line L and is given by:

$$f_R(L) = m + n + 1 \text{ if } L \text{ is labelled } (+, m, n) \text{ or } (-, m, n);$$

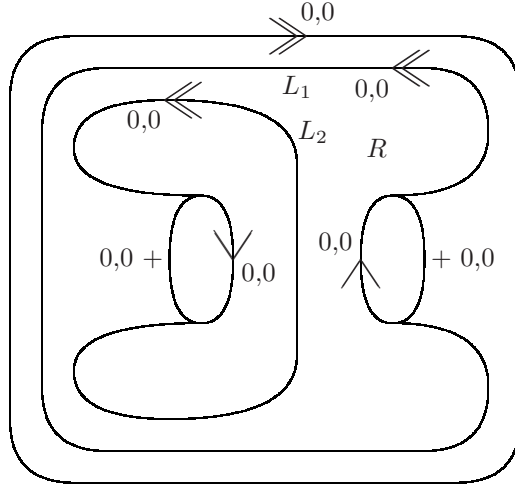


Figure 7.10: An impossible wireframe projection. For example, the labelling shown violates the region constraint on lines L_1, L_2 of region R .

$$\begin{aligned}
 f_R(L) &= m + n + 2 && \text{if } L \text{ is labelled } (\uparrow, m, n) \text{ or } (\uparrow, m, n); \\
 f_R(L) &= m + n && \text{if } L \text{ is labelled } (\downarrow, m, n) \text{ or } (\downarrow, m, n).
 \end{aligned}$$

Note that the region constraint is redundant if L_1 and L_2 belong to the same connected component of the wireframe projection. This is because, by construction of our catalogue from projections of physically realizable vertices, the region constraint holds for lines L_1 and L_2 meeting at a junction (and forming part of the frontier of the same region) and hence, by induction, for such lines L_1 and L_2 joined by a sequence of junctions. Nevertheless, the region constraint is essential. Figure 7.10 shows an example of an impossible wireframe projection whose impossibility would not be detected if the region constraint were not applied.

Note that it can easily be shown that the parity constraint is redundant if the outer-boundary and the region constraints are both applied.

Definition 7.2 *A labelling of a wireframe projection is legal if each junction labelling occurs in the catalogue of Figures 7.6–7.8 and the outer-boundary and region constraints are also satisfied.*

Theorem 7.3 *A line drawing of curved objects is realizable as the wireframe projection of a 3D scene satisfying conditions 1, 2, 3, 4 (given in Section 7.2) if and only if it has a legal labelling.*

Proof: The catalogue was constructed by listing all types of 3D vertices allowed by conditions 1, 2, 3, 4 and studying all possible projections from different viewpoints. It follows immediately that each junction in a wireframe projection must have a labelling in the catalogue. This shows that the catalogue represents

a necessary condition for realizability. Similarly, the necessity of the outer-boundary and region constraints have also been shown above.

To show that the existence of a legal labelling is a sufficient condition for realizability, note firstly that all labellings have been included in the catalogue because they are projections of allowed vertices. Therefore, taken separately, all labelled junctions are realizable as projections of 3D vertices satisfying conditions 1, 2, 3, 4. It remains to show that we can join these vertices together to form a 3D object. Consider the drawing as a partition of the plane into non-intersecting regions. For each region A , calculate N_A , the number of surfaces which project into A . This number is well defined by the region constraint. Lay N_A rubber sheets of the same 2D shape as A on top of each other and on region A in the drawing. For each line segment L in the drawing, create a convex, concave, occluding or extremal edge according to the semantic label s of L at the depth given by the numerical label m, n of L as follows: if s is '+' or '-', then this means creating a convexity or concavity in the rubber sheet which lies at depth $m + 1$; if s is '↑' or '↑↑', then this means joining the rubber sheets lying at depths $m + 1$ and $m + 2$ in the region to the right of L to create either a surface-normal discontinuity edge or an extremal edge. The rubber sheets partition 3D space into non-overlapping subsets. It only remains to specify which subsets should be filled with matter and which left empty. For any region R of the drawing, there will be matter between sheets at depth $m = 2i$ and $m = 2i + 1$ (for all i). The parity constraint ensures that the resulting 3D scene constructed from a wireframe projection will be of finite depth. ■

Not only does Theorem 7.3 generalize Sugihara's groundbreaking work [151] to curved objects, but we have also considerably simplified the expression of his original constraints. As an example of the use of Theorem 7.3, consider the wireframe projection of Figure 7.11(a) (adapted from an example given by Ernst [61]). The given labelling is legal, and hence this is a physically realizable wireframe projection. The corresponding opaque object is shown in Figure 7.11(b), and a method of constructing it from a flexible tube with triangular cross-section is shown in Figure 7.11(c).

In a wireframe projection of an entire but isolated object (or a set of objects each of which is wholly visible) we can easily identify the outer boundary B . Given a semantic labelling of the drawing, we can calculate for each region A the number of surfaces N_A which project into A by the following algorithm. Trace a straight-line segment L from A to any point on B , choosing the angle of L so that L does not pass through any junction. Without loss of generality, suppose that L leaves A horizontally to the right. Let N_{down} (respectively N_{up}) be the number of lines that L crosses which are labelled ↓ or ↓↓ (respectively ↑ or ↑↑). Then

$$f_R(L) = 2(N_{down} - N_{up}).$$

It follows that in a complete semantic and numerical labelling of the drawing, the value of n in the numerical labelling m, n of any line is redundant since it can be deduced from m and the semantic labelling. This generalizes a result first proved

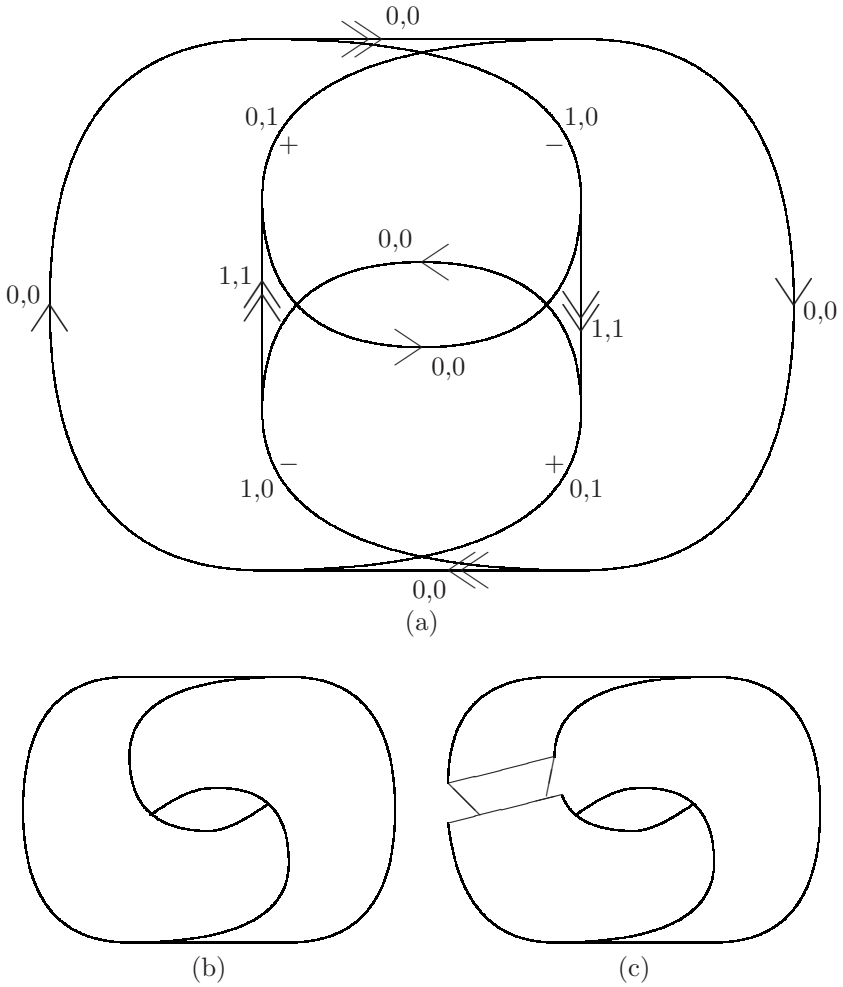


Figure 7.11: (a) A legally labelled wireframe projection, (b) the corresponding opaque object, (c) constructing the object.

by Williams [177] for wireframe projections of smooth curved objects with no surface-normal discontinuity edges. The labelling of wireframe projections using a combination of semantic labels and the single numerical depth label m was first suggested by Huffman in his seminal paper [75].

7.4 All Wireframes Are Ambiguous

It is well known that wireframe projections can be ambiguous [112, 118]. In this section we prove a negative result concerning the ubiquity of ambiguity, which will help us to put into perspective the positive results of later sections.

Theorem 7.4 (*Depth-reversal theorem*) *Consider a legal labelling of a wireframe projection P . If we have no depth information (such as the identification of a W junction as a $W(+)$ or a $W(-)$ junction), then another legal labelling of P can be obtained by the following depth-reversal operation: (1) change all ‘+’ labels to ‘-’ and vice versa and (2) change all numerical labels m, n to n, m .*

Proof: The proof is trivial by exhaustion over all labellings in the catalogue. For example, the first labelling for the $W(+)$ junction in Figure 7.6 is transformed into the first labelling for the $W(-)$ junction by the above operation. It is trivial to check that the outer-boundary and region constraints cannot be invalidated by the depth-reversal operation. ■

When the object represented by the wireframe projection is a cube, then this result corresponds to the famous Necker cube ambiguity. Interestingly, no junction labelling in our catalogue can be transformed into itself by the depth-reversal operation. This leads to the following result.

Lemma 7.5 *The only physically possible wireframe projections which are not subject to a depth-reversal ambiguity are those involving no junctions and in which all lines are labelled ‘→’ or ‘⇒’.*

Theorem 7.6 *All physically possible wireframe projections are ambiguous.*

Proof: Lemma 7.5 tells us that the only case to consider is when the wireframe projection contains no junctions. But in this case the outer boundary must consist of one or more closed curves. A closed curve C may be labelled ‘→’ or ‘⇒’. For example, a circle may be the projection of a sphere or a disk-shaped object. ■

Imagine a drawing consisting of n mutually intersecting circles. There are at least $2^n n!$ interpretations of the drawing as a wireframe projection of a 3D scene, since each circle could be the projection of either a sphere or a disk and there are $n!$ possible depth orderings of the n objects. There is another form of systematic ambiguity in wireframe projections which we call matter/space ambiguity. This occurs when it cannot be determined whether a subset of 3D

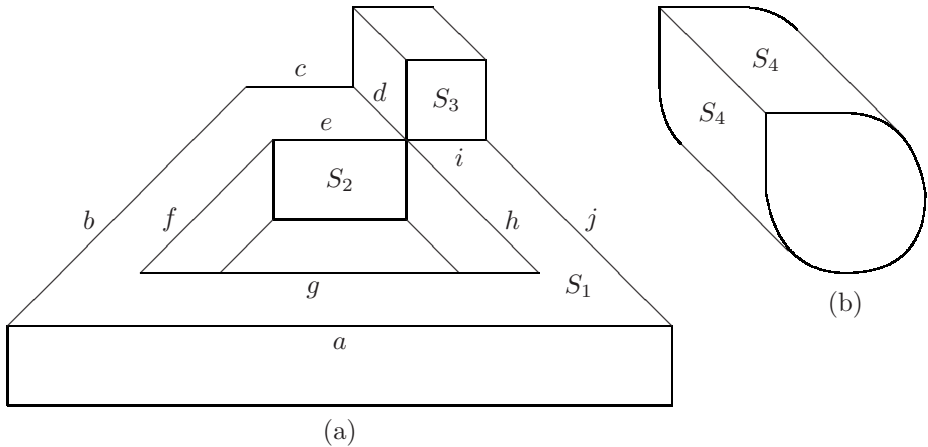


Figure 7.12: S_1, S_2, S_3, S_4 are examples of faces.

space bounded by a set of surfaces represents an object or a hole. For example, in a wireframe projection consisting of just two concentric circles, the inner circle could represent a sphere in front of or behind another larger sphere, or it could represent a hole inside an outer sphere.

7.5 Identifying Faces

An essential part of the interpretation of a wireframe projection is the identification of object faces. First of all we require a formal definition of a face.

From our assumptions on object shape in Section 7.2, a 3D edge E is the intersection of two 3D surfaces with distinct surface normals at every point of E . Each edge has a direction. As we walk along E in this direction and on the exterior of the object, let $S_L(E)$ represent the surface on our left and $S_R(E)$ the surface on our right.

Definition 7.7 Let $B = (E_1, \dots, E_r)$ be a circuit of 3D edges such that, for $i = 1, \dots, r$, V_i is the endpoint of E_i and the start point of E_{i+1} (where E_{r+1} is understood to mean E_1). Then B is a 3D face boundary if, for $i = 1, \dots, r$, E_{i+1} is the first edge leaving V_i , which lies to the right of E_i , on $S_R(E_i)$.

By this definition, $S_R(E_i) = S_R(E_{i+1})$ at each V_i , and hence B is a boundary of the face which lies to its right.

Definition 7.8 A face of a 3D object is a closed connected C^3 surface patch P bounded by non-intersecting 3D face boundaries B_1, \dots, B_h for some $h \geq 0$.

If the face is planar, then one of B_1, \dots, B_h is the outer boundary and the others are boundaries of holes. For curved surfaces, no such distinction can be

made. For example, the curved surface of a cylinder has two circular 3D face boundaries, neither of which is the outer boundary. A sphere has a single face with no 3D face boundary (i.e. $h = 0$ in Definition 7.8). Note that a face may touch itself at a point (as does surface S_1 in Figure 7.12(a)) or along an edge (as does surface S_4 in Figure 7.12(b)). According to Definition 7.8, surfaces S_2 and S_3 of Figure 7.12(a) cannot be merged to constitute a single face. Furthermore, the 3D face boundary of S_1 is necessarily $(a, b, c, d, e, f, g, h, i, j)$; it is not possible to interpret (e, f, g, h) as a hole boundary and (a, b, c, d, i, j) as an outer boundary.

Definition 7.9 *A face circuit in a wireframe projection is a circuit of line segments which are the projection of the 3D face boundary of an object face.*

The identification of face circuits is a major step in converting a wireframe model into a surface-based model such as a B-rep [103, 147, 170, 171]. In fact, we will show that face circuits can be unambiguously deduced from the labelling of a wireframe projection. Each labelled junction in Figures 7.6–7.8 is the projection of a 3D vertex. (In fact, each labelled junction corresponds to two distinct vertices due to the matter/space ambiguity, but such ambiguity does not affect face circuits.) By reconstructing a 3D vertex projecting into each labelled junction, we deduced the face-circuit information given in Figures 7.13–7.15. In these figures, thick lines represent lines present in the wireframe projection, whereas thin lines represent fragments of face circuits. The face-circuit line drawn on the left (right) of a line L labelled ‘+’ or ‘−’ corresponds to the face projecting into the region to the left (right) of L . For a line labelled ‘↑’, the two face-circuit lines are both drawn to the right of L : the face-circuit line which is nearer to (farther from) L corresponds to the face which is nearer to (farther from) the viewpoint. For brevity of presentation, $X(\neq)$ junctions are not given in Figure 7.14. The presence of a $X(\neq)$ junction on a line L has no effect on the face-circuit lines of L . Note that since extremal lines (i.e. lines labelled ↑) are not projections of 3D edges, they do not belong to any face circuit. This explains why we do not need to include 2-tangent junctions in Figure 7.15. For clarity of presentation, the numerical labels are not shown on the lines in Figures 7.13–7.15, but can be read off directly from the corresponding labelled junction in Figures 7.6–7.8.

Theorem 7.10 *Given a legal labelling of a wireframe projection, we can uniquely determine the face circuits of the corresponding 3D scene in linear time.*

Proof: Denote the wireframe projection by P . Each junction in a legal labelling of P uniquely determines face-circuit fragments, as illustrated by Figures 7.13–7.15. It is then possible to determine complete face circuits by concatenating these fragments. There is no possible ambiguity in the face circuits in the construction. In the case of curved objects, Theorem 7.3 guarantees that a 3D scene exists which projects into P . ■

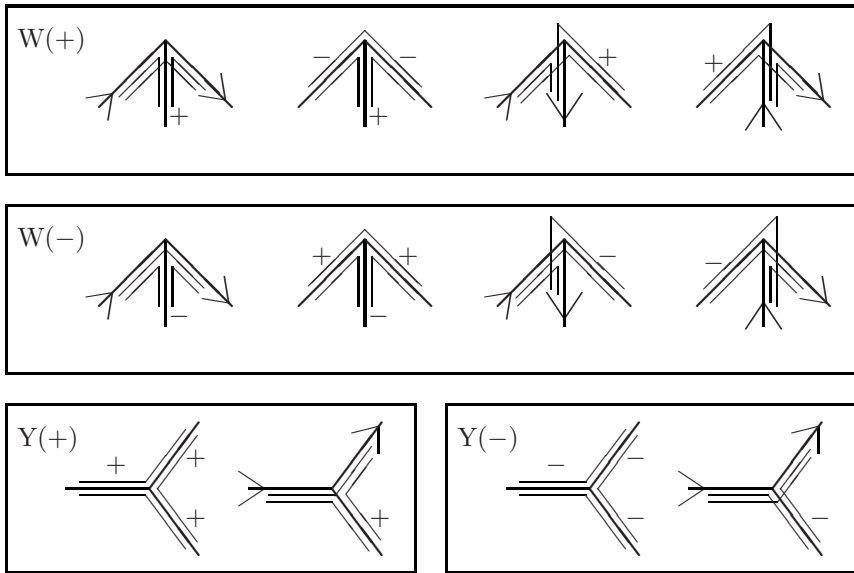


Figure 7.13: Catalogue of W and Y junctions with face-circuit fragments.

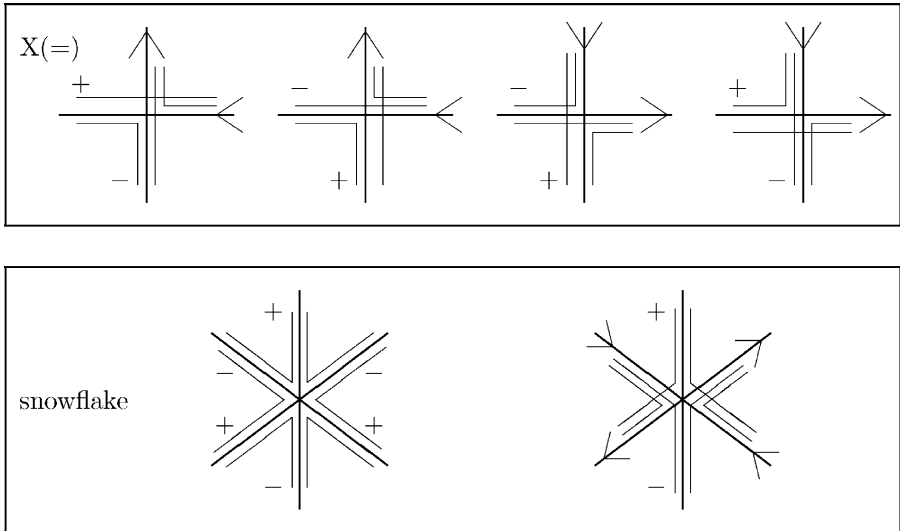


Figure 7.14: Catalogue of X and snowflake junctions with face-circuit fragments.

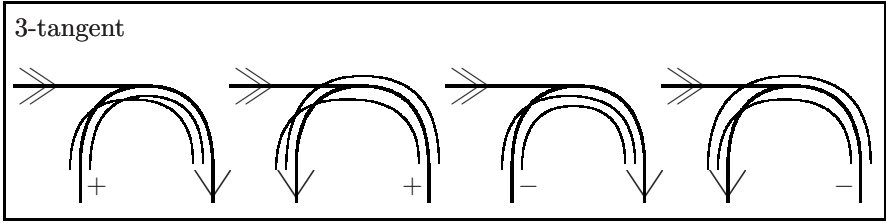


Figure 7.15: Catalogue of 3-tangent junctions with face-circuit fragments.

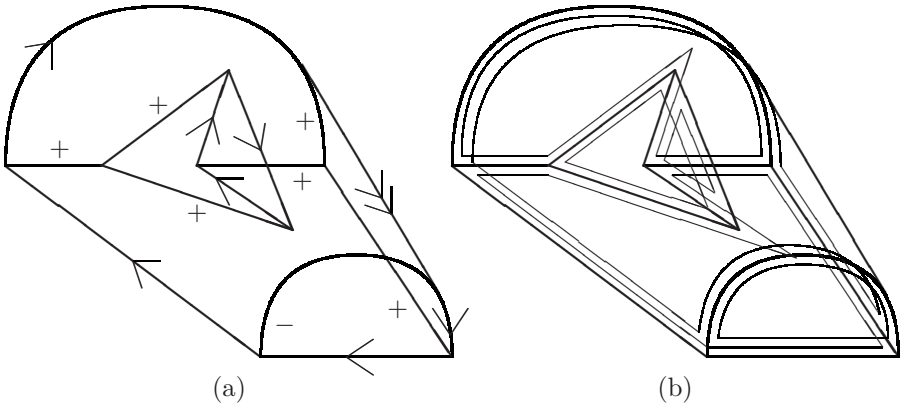


Figure 7.16: (a) A legally labelled wireframe projection; (b) its face circuits derived from the catalogue of Figures 7.13–7.15.

Figure 7.16 shows an example of the determination of face circuits of a wireframe projection. The wireframe projection in Figure 7.16(a) has only two legal labellings. One legal labelling is shown in Figure 7.16(a), and the other is its depth reversal. For clarity of presentation, the numerical labels have been omitted (although they are essential in eliminating certain physically impossible labellings). Using the face-circuit constraints given in Figures 7.13–7.15, we easily obtain the face circuits shown in Figure 7.16.

As another example, consider again the wireframe projection of Fig. 7.11(a). From the legal labelling given in the figure, the catalogue of Figure 7.15 allows us to deduce that the corresponding object has just two 3D face boundaries. In fact, as can be seen in Figure 7.11(c), this object has a single face and is a version of the Möbius strip with a triangular cross-section.

Our approach consists in finding all possible semantic and numerical labellings of a wireframe projection (of which there may be an exponential number) and then determining the face circuits for each such legal labelling. This can be compared with the traditional face-based approach [112, 147, 170, 171]. For example, Shpitalni and Lipson [147] determine a set of possible face circuits and then find maximal consistent sets of face circuits. Their algorithm

has worst-case time complexity which is exponential in the number of putative face circuits, which is itself a potentially exponential function of the size of the drawing. We have thus reduced a doubly exponential complexity to a simple exponential complexity. Liu et al. [105] eliminate many physically impossible face circuits using properties derived from the planarity of faces, but this means that their system can only identify faces in a drawing of a curved object after the user has provided a drawing of a polygonal approximation of the object.

However, Definition 7.8 tells us that the projection of a face is, in fact, a set of face circuits. In order to identify projections of faces, rather than just face circuits, we use the same construction as in the proof of Theorem 7.3. Suppose that we are given a legally labelled wireframe projection with regions R_1, \dots, R_t . For each region R_i we can easily determine the total number of faces $f(R_i)$ projecting into R_i . We can then, for each R_i , create $f(R_i)$ copies $R_{i,1}, R_{i,2}, \dots$ of R_i , which we call patches. To group together patches belonging to the same face it suffices to apply the following rule until convergence: patches $R_{i,u}, R_{j,v}$ are merged whenever (1) R_i, R_j are adjacent regions separated by a line L with numerical label (m, n) and $u = v \leq m$ or $f(R_i) - u = f(R_j) - v < n$ (i.e. $R_{i,u}, R_{j,v}$ correspond to the same face which either passes in front of or behind the 3D edge projecting into L) or (2) $i = j, u = m + 1, v = m + 2$, where region R_i has a line L with label (\Rightarrow, m, n) on its boundary (with R_i to the right of L as we follow the direction of the \Rightarrow arrow). Each patch $R_{i,u}$ is a subset of a face (the projection of a face into region R_i). The set of patches thus represents an oversegmentation of the 3D object surfaces. Two adjacent patches separated by a line L can be merged if L is the projection of a 3D edge lying on neither of the patches (case (1) above). Two patches can also be merged if they correspond to the same 3D face which folds over itself to form an extremal edge (case (2)).

7.6 Common-Surface Constraints

Instead of calculating face circuits from legal labellings, it is sometimes possible to deduce face-circuit information directly without exhausting over all legal labellings. We will show in this section that face-circuit fragments can be directly determined given information concerning pairs of adjacent junctions. This information may be obtained, for example, from vanishing points or by applying local consistency operations (Section 8.3) to the labelling constraints of Section 7.3.

Remember that W and Y junctions can be classified as $+$ or $-$ depending on the relative 3D orientations of the edges which meet at the vertex which projects into junction J . Knowledge of the vanishing points of the three lines which meet at J is sufficient to classify J as $+$ or $-$ [127]. The following theorem tells us that knowing whether two W/Y junctions joined by a line are classified as the same or opposite sign allows us to determine two distinct triples of consecutive lines in face circuits.

Consider a pair of junctions (J_1, J_2) joined by a line where each of J_1, J_2 is either a W or a Y, as illustrated in Figure 7.17. We call this a *W/Y junction*

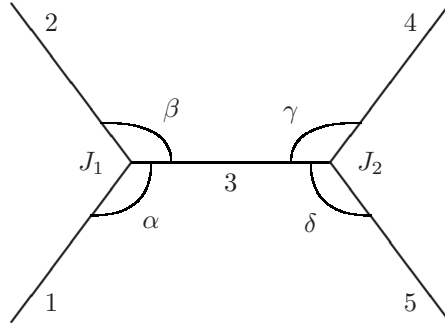


Figure 7.17: A generic W/Y junction pair (J_1, J_2) . Note that lines 1-5 may be curved and the angles $\alpha, \beta, \gamma, \delta$ may also be acute or reflex.

pair. Each of the lines 1-5 may be straight or curved and there may be any number of $X(\neq)$ or 3-tangent junctions on line 3 (but no $X(=)$ junctions). Let $n_{refl}(J_1, J_2)$ be the number of the angles $\alpha, \beta, \gamma, \delta$ which are reflex (i.e. greater than π); let $n_{3t}(J_1, J_2)$ be the number of 3-tangent junctions lying on line 3; let $n_+(J_1, J_2)$ be the number of the junctions J_1, J_2 which are + (i.e. $W(+)$ or $Y(+)$).

Theorem 7.11 (*Common-surface constraint*) *Let (J_1, J_2) be a W/Y junction pair in a wireframe projection. If $p = (n_{refl}(J_1, J_2) + n_{3t}(J_1, J_2) + n_+(J_1, J_2)) \bmod 2$, then both $(1, 3, 5 - p)$ and $(2, 3, 4 + p)$ are triples of consecutive lines in a face circuit, where line numbers are as in Figure 7.17.*

Proof: The result follows by simple exhaustion over all legal labellings of junction pairs followed by reconstruction of the corresponding surfaces as in the proof of Theorem 7.10. ■

It is important to note that, by our assumption that object edges are surface-normal discontinuities, we have implicitly disallowed a concave/convex transition on an edge. An example of such a transition is point P in Figure 7.18. In fact, the faces which intersect to form the edge are tangential to P [34]. Theorem 7.11 is no longer valid if $+/-$ transitions can occur on line 3 of Figure 7.17.

7.7 Coplanarity Constraints

The presence of straight lines in a wireframe projection can provide information about the 3D positions of object vertices, provided we make the following assumption about object shape.

Straight-edge formation assumption: Any straight object edge is formed by the intersection of two locally planar surfaces.

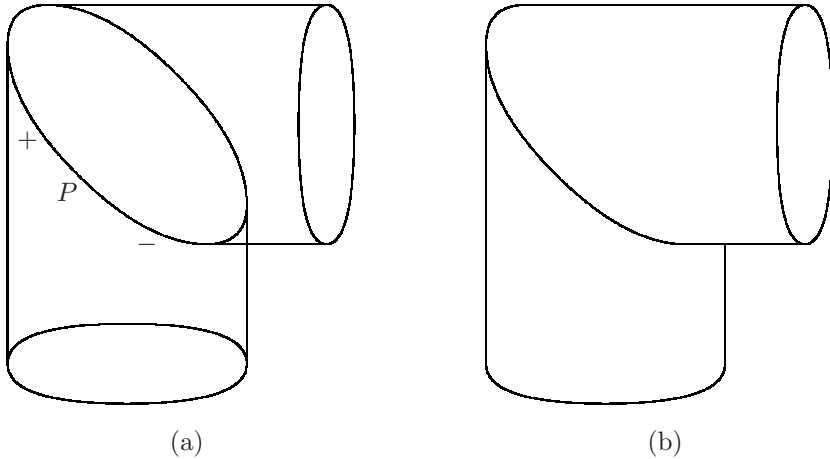


Figure 7.18: (a) A wireframe projection involving a $+/-$ semantic label transition at P ; (b) a view of the corresponding opaque object.

This assumption was first stated in [37] in the context of the interpretation of line drawings of opaque objects. It is clearly not always valid in man-made objects, but it allows us to extend techniques developed for polyhedral objects to a large class of curved objects. Under this assumption, neither 3-tangent junctions nor $+/-$ transitions (as in Figure 7.18) can occur on straight lines. Most importantly, together with the general viewpoint assumption that we have already made, this new assumption means that straight lines in a wireframe projection can provide coplanarity constraints on edges. For example, the straight-edge formation assumption implies that if line segments L_1, L_2, L_3 are consecutive line segments in a face circuit and L_1, L_2, L_3 are straight lines, then the 3D edges E_1, E_2, E_3 projecting into L_1, L_2, L_3 are coplanar. (Even if L_1 and L_3 are curved, we can still obtain a constraint, but this time involving the tangents to E_1 and E_3 at the vertices where they meet E_2 . See Chapter 6 for details.)

Thus Theorem 7.11 immediately provides coplanarity constraints provided that line 3 in Figure 7.17 is a straight line. Note, however, that we can say more: the coplanarity of a set of 3D edges may be detected even when the edges do not lie on the same face circuit. Figure 7.19 shows an example. In this wireframe projection, knowing that A and B are both $W(+)$ junctions allows us to deduce that lines 1, 3, 4 are coplanar (as are lines 2, 3, 5). By exhausting all cases, we discover that the presence of any number of $X(=)$, $X(\neq)$ or snowflake junctions on line 3 in the W/Y junction pair of Figure 7.17 does not invalidate the coplanarity constraint, which we state formally as follows.

Theorem 7.12 (Coplanarity constraint) *Suppose that lines 1–5 of the W/Y junction pair (J_1, J_2) in Figure 7.17 are straight lines and there are any number of X or snowflake junctions on line 3. If $p = (n_{refl}(J_1, J_2) + n_{3t}(J_1, J_2) + n_+(J_1, J_2)) \bmod 2$, then both $(1, 3, 5 - p)$ and $(2, 3, 4 + p)$ are triples of coplanar*

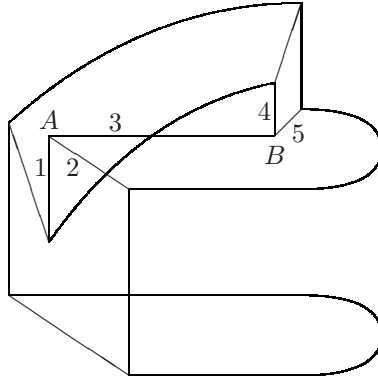


Figure 7.19: The coplanarity of lines 1, 3, 4 can be deduced despite the fact that they do not lie on the same face.

lines.

As an example of the strength of the coplanarity constraint, consider the wireframe projection shown in Figure 7.20(a). The wireframe projection has two legal labellings, one of which is the depth reversal of the other. The bottom left corner of one of these legal labellings is illustrated in Figure 7.20(b). The visible lines of this labelling corresponds to the famous Penrose triangle [128] shown in Figure 7.20(c). The physical impossibility of the wireframe projection follows from the coplanarity constraint. In both legal labellings, junction pairs B, C and C, D are both of opposite sign. It follows from Theorem 7.12 that both A, B, C, D and B, C, D, E are projections of sets of coplanar points. Hence A, B, C, D, E are projections of coplanar points. By symmetry, E, D, F, G, H are projections of coplanar points, as are H, G, I, B, A . These three planes should then meet at a point in 3D space, which implies that lines AB , ED and HG , when extended, should meet at a point in the drawing. Since this is clearly not the case, the wireframe projection is physically impossible.

To obtain necessary and sufficient conditions for the realizability of a wireframe projection of curved objects involving straight lines, we require not only the coplanarity constraints deduced from Theorem 7.12, but also inequality constraints expressing the fact that the nearer line passes in front of the distant line at each $X(\neq)$ junction and that the two faces meeting at each concave (convex) edge subtend an angle less (greater) than π (Chapter 6). This leads to a linear programming problem P . However, a realizable wireframe projection may give rise to a problem P which has no solution due to rounding errors or small user errors in the positions of junctions. Different solutions have been proposed to overcome this problem of superstrictness [155, 135, 37]. It should be noted that testing the realizability of a wireframe projection still involves solving a linear programming problem for each of a possibly exponential number of legal labellings.

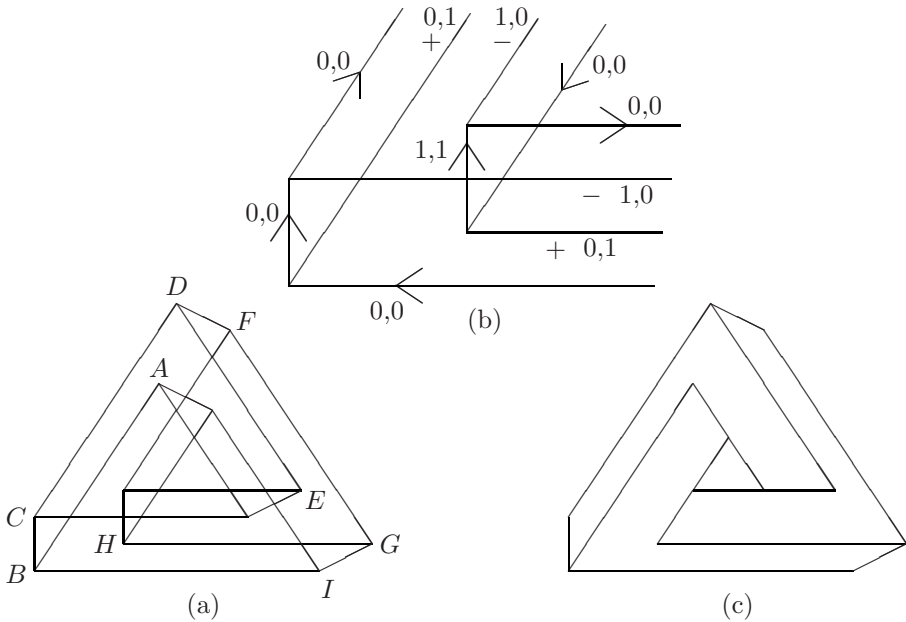


Figure 7.20: (a) A wireframe projection; (b) a part of a legal labelling; (c) the visible lines corresponding to this labelling.

7.8 Unambiguous Wireframes

Although all physically realizable wireframe projections are ambiguous by Theorem 7.6, we show in this section that the presence of extra information, together with some further restrictions on the class of 3D scenes, can eliminate all ambiguity either in the determination of the face-circuits or in the complete interpretation of the wireframe.

Definition 7.13 A vertex V is simple if none of the edges which meet at V continues through V . Thus vertices which project into $X(=)$ or snowflake junctions are not simple.

Definition 7.14 A wireframe projection is simple polyhedral if it is a projection of a collection of polyhedral objects with simple vertices satisfying conditions 1–4 (of Section 7.2).

Theorem 7.15 Let P be a simple polyhedral wireframe projection. If for each W/Y junction pair (J_1, J_2) we know whether J_1, J_2 are of the same or of opposite signs, then the face circuits of P can be unambiguously identified in linear time.

Proof: Since P is simple polyhedral, it only contains junctions of type $W, Y, X(\neq)$. $X(\neq)$ junctions have no effect on face circuits. The common-surface constraint of Theorem 7.11 allows us to construct without ambiguity the face

circuits of P . This can clearly be achieved in linear time, by extending face-circuit fragments until a closed circuit is obtained and restarting this procedure until each line is has been placed in two distinct face circuits. ■

Corollary 7.16 *Let P be a simple polyhedral wireframe projection. If for each line L in P we know the position of the vanishing point of L , then the face circuits of P can be unambiguously identified in linear time.*

Proof: By the assumption of a polyhedral scene, all edges are straight lines. The position of the vanishing point of each line L in P allows us to determine whether each W/Y junction J of P is $+$ or $-$ [127]. Thus, by Theorem 7.15, the face circuits of P can be determined in linear time. ■

Sugihara [151, 155] and Alevizos [5] both study a specific type of wireframe projection in which the user specifies, for each line L , whether L is of even or odd depth. In other words, if (m, n) represents the numerical label of L , the user must specify whether m is even or odd. We say that such a wireframe projection has depth-parity information. Consider a W/Y junction pair as illustrated in Figure 7.17. Recall that $n_{refl}(J_1, J_2)$ is the number of the angles $\alpha, \beta, \gamma, \delta$ in Figure 7.17 which are greater than π . We say that a W/Y junction pair has even (odd) *angle parity* if $n_{refl}(J_1, J_2)$ is even (odd). Given the depth parity of each of lines 1, 2, 4, 5 in a W/Y junction pair, we can also clearly determine the parity of the sum s of the depths of lines 1, 2, 4, 5; we say that a W/Y junction pair has even (odd) *depth parity* if s is even (odd). Having established the necessary notation, we can now state the following lemma.

Lemma 7.17 *Let (J_1, J_2) be a W/Y junction pair in a wireframe projection formed under assumptions 1, 2, 3, 4 of Section 7.2. If the angle parity of (J_1, J_2) is the same as its depth parity, then J_1, J_2 are of the same sign; otherwise J_1, J_2 are of opposite sign.*

Proof: The proof is simple, although tedious, by exhaustion over all possible labellings of all possible configurations of W/Y junction pairs. ■

Thus, depth-parity information for each line allows us to deduce for each W/Y junction pair (J_1, J_2) , whether J_1, J_2 are of the same or opposite signs. Theorem 7.15 then provides us with a simple proof of the following result first stated by Alevizos [5].

Corollary 7.18 *Let P be a simple polyhedral wireframe projection with depth-parity information. Then the face circuits of P can be unambiguously identified in linear time.*

Under perspective projection, the vanishing points of all lines provide sufficient information to identify the sign of W/Y junctions. Under orthographic

projection, on the other hand, the sign of W/Y junctions is inherently ambiguous, as illustrated by the depth-reversal phenomenon (Section 7.4) in which all W/Y junctions change sign. However, we can use the fact that, under orthographic projection, if two lines L_1, L_2 in the wireframe projection are parallel, then, by the general viewpoint assumption, L_1, L_2 are projections of parallel lines in 3D. If, in the generic W/Y junction pair shown in Figure 7.17, all lines are straight lines and lines i and j are parallel (where $i \in \{1, 2\}$ and $j \in \{4, 5\}$), then it follows that lines $i, 3, j$ are coplanar and hence, by the straight-edge formation assumption, that $i, 3, j$ is a face-circuit fragment.

Corollary 7.19 *Let P be a simple polyhedral wireframe projection formed by orthographic projection. If all lines of P are parallel to one of only three directions, then the face circuits of P can be uniquely determined in linear time.*

Proof: Since all lines of P are parallel to one of only three directions, at each W/Y junction pair there must be two pairs of parallel lines (either (1, 4), (2, 5) or (1, 5), (2, 4) where the numbers refer to the lines in Figure 7.17). From the discussion above, at each W/Y junction pair, we can thus determine two face-circuit fragments (either (1, 3, 4), (2, 3, 5) or (1, 3, 5), (2, 3, 4)). The result then follows by the same argument as in the proof of Theorem 7.15 ■

Theorem 7.20 *Let P be a simple polyhedral wireframe projection. If for each vertex V in P we know the 3D coordinates of the point which projects into V (i.e. we have a 3D wireframe model), then the semantic and numerical labelling of P is unique. Furthermore, the physical realizability of P can be checked and the semantic and numerical labelling of P can be found in quadratic time.*

Proof: From the 3D positions of vertices we can determine for each W/Y junction J in P whether J is + or -. Theorem 7.15 then tells us that we can uniquely determine the face circuits of P . Since we know the 3D coordinates of all vertices, we can then easily determine the 3D face boundaries projecting into the face circuits of P . In order to identify faces, it only remains to determine for each 3D face boundary B whether B is the outer boundary of a face or the boundary of a hole. Consider only those 3D face boundaries B_1, \dots, B_r which lie in the same plane Q_B as B . Let H_B be a half-line from a point on B to infinity within the plane Q_B such that H_B does not intersect any object vertex. Then B is an outer boundary (hole boundary) if the number of intersections of H_B with the 3D face boundaries B_1, \dots, B_r is even (odd). (Note that here we do not need to consider any intersections of H_B with faces which are not coplanar with B , which renders this operation much simpler than when it is performed as a means of identifying face circuits [82, 112].)

For each line L in P , it is then trivial to determine the semantic label of L from the resulting set of faces. Furthermore, by tracing a ray R from the viewpoint through an arbitrary point on the 3D edge E_L which projects into L and counting the number of faces which R intersects in front of and behind E_L allows us to deduce the numerical label (m, n) of L . To check the physical

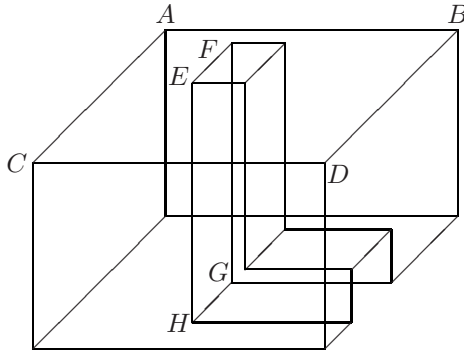


Figure 7.21: An example of an impossible 3D wireframe model of a polyhedral object.

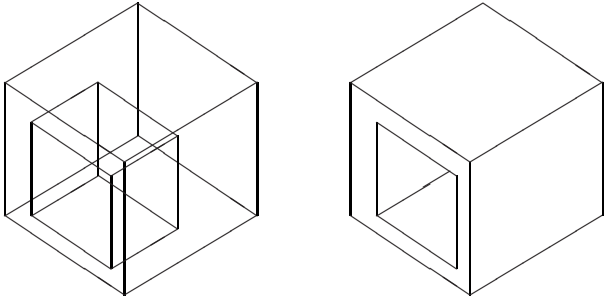
realizability of P we only need to determine for each pair of faces F_1, F_2 , whether F_1, F_2 intersect. If they do, then this intersection must correspond to an edge on the 3D face-boundaries of both F_1 and F_2 . ■

As an example, Figure 7.21 shows a physically impossible 3D wireframe model. It is illegal because the two faces $ABCD$ and $EFGH$ intersect along an edge whose projection is not present in the model. Note, on the other hand, that this wireframe model is physically realizable if surface $ABCD$ may be curved.

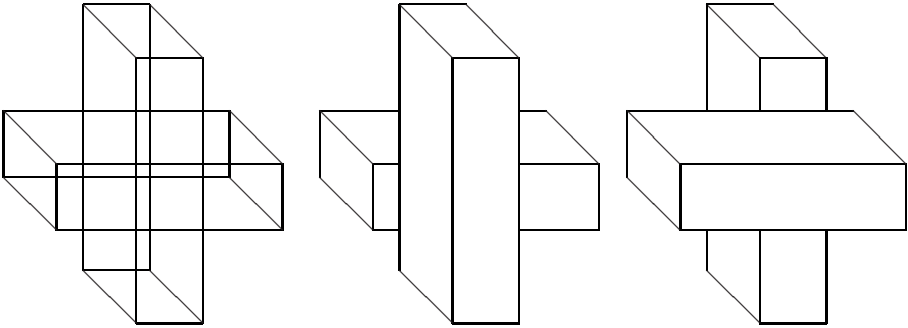
7.9 Residual Ambiguity

It is worthwhile studying wireframe projections in which the assumptions of the uniqueness theorems of Section 7.8 have been relaxed in order to see which kinds of ambiguity can occur. The simple polyhedral wireframe projection on the left-hand side of Figure 7.22(a) is an orthographic projection of a pair of cubes. All lines are parallel to one of just three directions. By Corollary 7.19, all face circuits can be uniquely determined. However, we cannot determine whether one cube lies in front of, behind or inside the other. Thus the numerical labelling of lines is not unique. Furthermore, neither the semantic labelling nor the faces are unique since if one cube lies inside the other, then the left-hand side of the inner cube may be a face or a hole (as illustrated on the right-hand side of Figure 7.22(a)).

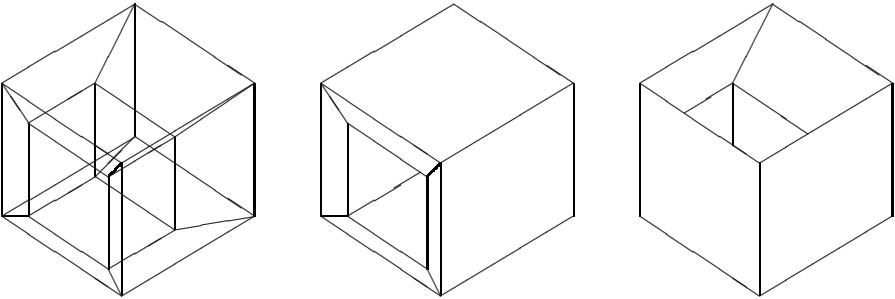
Under orthographic projection, the face circuits of the wireframe projection shown on the left-hand side of Figure 7.22(b) can be uniquely determined (by Corollary 7.19) provided $X(=)$ junctions cannot occur. This is despite depth-reversal ambiguity as well as relative-depth ambiguity between the two objects. However, if $X(=)$ junctions may occur, then the wireframe projection may be interpreted as either of the two distinct cross-shaped objects (formed by sticking two rectangular objects together) shown on the right-hand side of Figure 7.22(b).



(a)



(b)



(c)

Figure 7.22: Examples of ambiguous wireframe projections.

These two interpretations, as well as a third interpretation as two separate objects, all give rise to distinct sets of face circuits.

Finally, the wireframe projection shown on the left-hand side of Figure 7.22(c) is a classic example [112, 118] of an ambiguous 3D wireframe model with non-trihedral vertices. Two possible interpretations are shown on the right-hand side of Figure 7.22(c). These interpretations clearly do not have the same face circuits.

The main source of ambiguity in wireframe projections remains the missing depth dimension. Different heuristics have been proposed to choose among different interpretations of a wireframe projection, favouring commonly occurring features of man-made objects such as planarity, symmetry, orthogonality, equality of angles or lengths [111, 99, 103]. Leclerc and Fischler [99] state an interesting condition for evaluating the psychological plausibility of a reconstructed object O , namely that the same object O be produced by the recovery algorithm from a wireframe projection of O from a different viewpoint. We advocate the use of a rich discrete labelling scheme and complete search, as discussed in detail in Chapters 3–5 for visible-line drawings, before embarking on an incomplete search over real-valued parameters.

The complexity of testing the existence of a valid semantic and numerical labelling of a wireframe projection is an open problem (although it is likely to be NP-complete). A similar remark holds for testing the realizability of a wireframe projection.

A catalogue of labelled junctions is insufficient to disambiguate drawings of complex objects with tetrahedral vertices and curved surfaces. In the rest of this chapter, we therefore introduce new constraints between distant lines based on the local planarity of surfaces and introduce a richer labelling scheme which allows us to identify locally planar surfaces of curved objects. We show how the valued constraint satisfaction framework can be used to express preferences for interpretations involving common features such as locally-planar surfaces, orthogonal edges and pairs of parallel 3D edges.

7.10 Constraints Between Distant Lines

It is well known that low-order constraints often exist between distant lines in drawings of polyhedral objects. In Chapter 3, we gave two generic constraints, the cyclic-path constraint and the parallel-lines constraint, which extended previously published constraints, in particular, Huffman's cut-set rule based on reasoning in dual space [77]. Unfortunately, these constraints only apply to drawings in which only visible lines are shown. In this section, we merge and generalize these constraints so that they can be applied to wireframe projections. In the next section, we use this new wireframe-path constraint to generate the catalogue of labelled projections of tetrahedral vertices.

The cyclic-path and parallel-lines constraints (Chapter 3) generate a constraint on the labels that can be assigned to a set of lines intersected by a path in a visible-line drawing; this path necessarily lies on visible surfaces. In

wireframe projections we need to specify on which surface the path lies by giving the depth m of the surface at each point of the path. We assume throughout this section that the objects depicted in the drawings are polyhedral. We show in Section 7.13 how to apply the wireframe-path constraint to drawings of curved objects with some planar faces.

Given a path Π joining two lines L_1, L_2 , which are projections of edges E_1, E_2 which are known to intersect in 3D, we can place restrictions on the labels which can be simultaneously assigned to the lines intersected by Π . Edges E_1, E_2 meet in 3D at a point P which projects into a point Q , known as the anchor point. We may know that E_1, E_2 intersect for any of the following reasons:

- Under orthographic projection, L_1 and L_2 are parallel lines, which under the general viewpoint assumption implies that E_1 and E_2 are parallel and hence meet at a point at infinity P .
- Under perspective projection, E_1 and E_2 have been deduced to be parallel since L_1 and L_2 meet at a vanishing point Q .
- L_1 and L_2 meet at a viewpoint-independent junction Q , which implies that E_1 and E_2 meet at a vertex P .
- $L_1 = L_2$ in which case $E_1 = E_2$ and Q can be chosen to be any point on L_1 .

Let S_1, \dots, S_t be the set of surfaces through which path Π passes and let Q be the anchor point. We write $S_i \geq_Q S_j$ if S_i passes behind or intersects S_j at Q (with the inequality being strict if the surfaces do not intersect at Q). Similarly, we write $E_i \geq_Q S_j$ ($S_i \geq_Q E_j$) if E_i (S_i) passes behind or intersects S_j (E_j) at Q . The wireframe-path constraint simply says that it is impossible to have $E_1 \geq S_1 \geq_Q S_2 \geq_Q \dots \geq_Q S_t \geq_Q E_2$ if there is at least one strict inequality, since we know that E_1 and E_2 intersect at Q .

Figure 7.23 shows the six distinct configurations at which a path can begin, and Figure 7.24 shows the seven distinct configurations at which a path can terminate. For brevity of presentation, we have omitted from both of these figures the reflected versions of the six configurations, which nevertheless must be included in a complete list of start and end configurations. As an example, the reflected version of Figure 7.23(a) is identical except that path Π leaves line L_1 vertically upwards rather than vertically downwards.

The inequalities given in Figure 7.23 (such as $p \leq m$ in Figure 7.23(c)) guarantee that at Q the surface S_1 in which path Π begins either passes through or in front of edge E_1 . If S_1 passes strictly in front of E_1 at Q , then the start configuration is known as *strictly positive*. It is tedious but easy to check that the start configurations in Figure 7.23(c)–(f) are strictly positive if:

- In Figure 7.23(c), we also have $q > n$;
- In Figure 7.23(d), we also have $q > n + 1 + m' - m$;
- In Figure 7.23(e), we also have $p > m$;

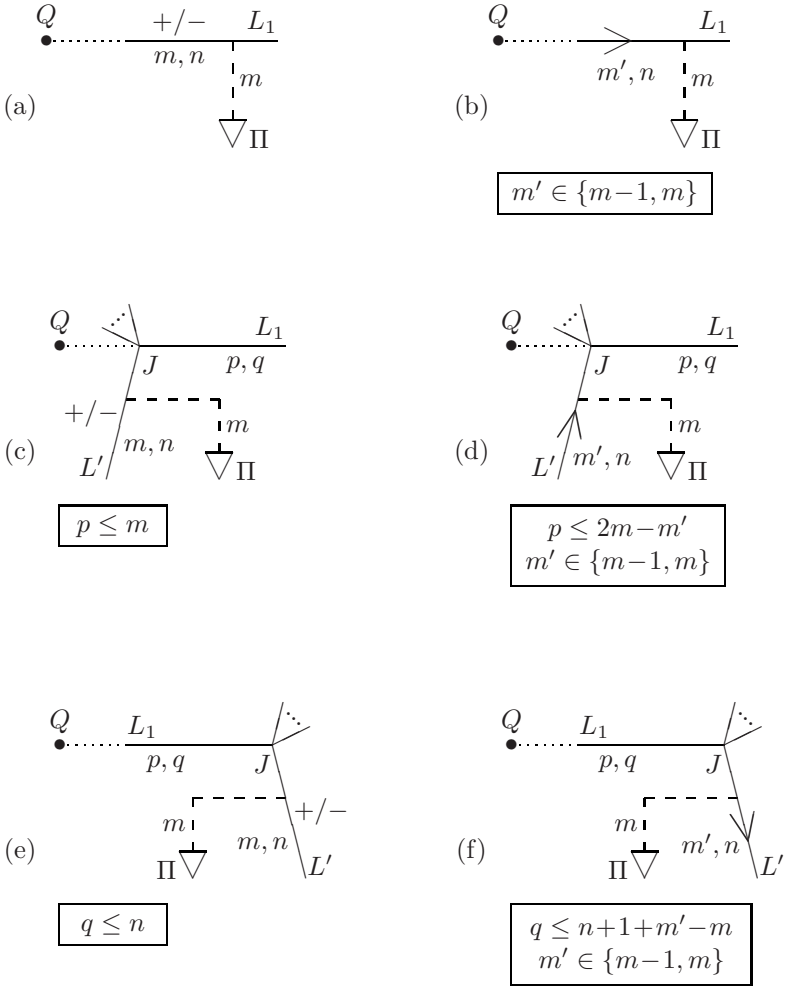


Figure 7.23: How paths start. (To this list we must also add the reflected version of each of the above configurations.)

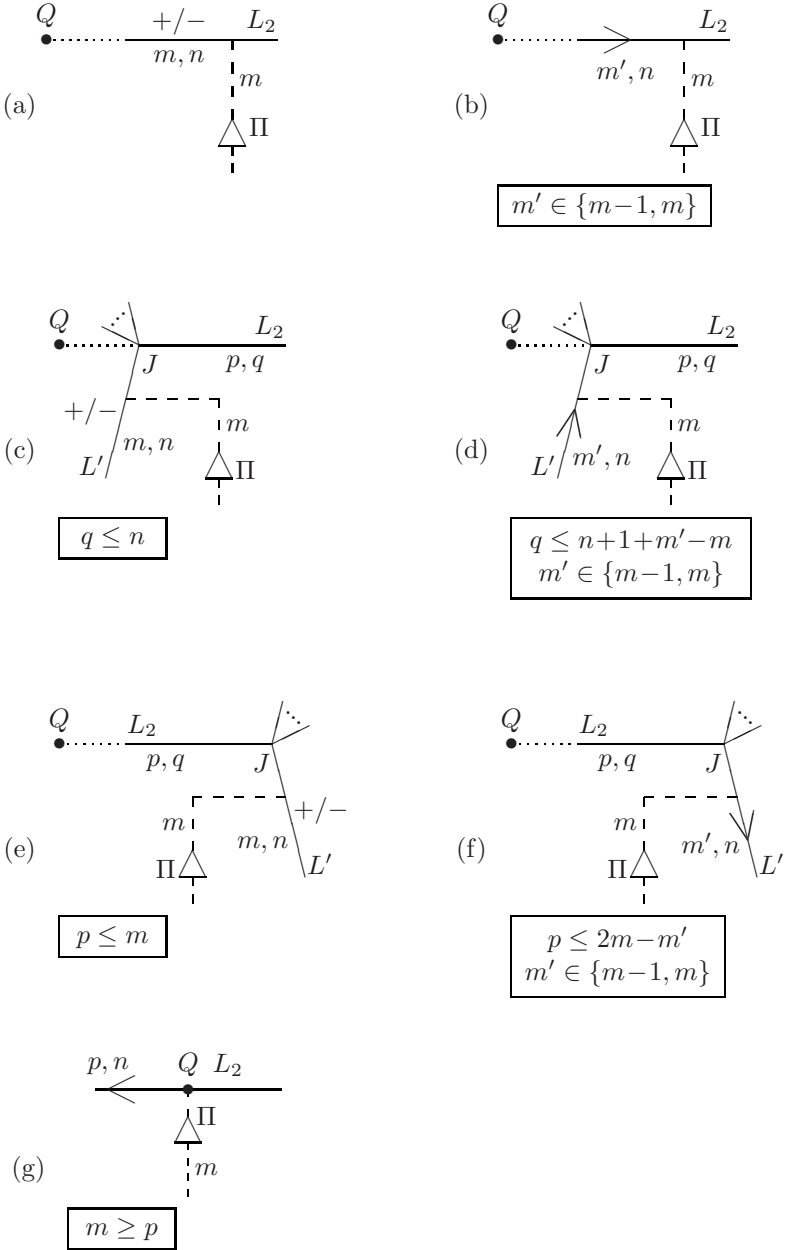


Figure 7.24: How paths end. (To this list we must also add the reflected version of each of the above configurations.)

- In Figure 7.23(f), we also have $p > 2m - m'$.

Similarly, the inequalities given in Figure 7.24 guarantee that at Q the final surface S_t of Π either passes through or behind edge E_2 . The end configuration is known as *strictly positive* if S_t passes strictly behind E_2 at Q . The end configurations in Figure 7.24(c)–(f) are strictly positive if:

- In Figure 7.24(c), we also have $p > m$;
- In Figure 7.24(d), we also have $p > 2m - m'$;
- In Figure 7.24(e), we also have $q > n$;
- In Figure 7.24(f), we also have $q > n + 1 + m' - m$.

The end configuration in Figure 7.24(g) is always strictly positive provided the occluding label for L_2 implies a depth discontinuity at Q (which is the case, for example, in a wireframe projection of a single manifold object).

Definition 7.21 *A path in a wireframe projection consists of an anchor point Q and a locus Π of points, together with a depth label m at each point of Π . The depth label m remains constant between intersections. Locus Π begins in one of the start configurations in Figure 7.23, contains any number of strictly positive or null intersections (as given in Figure 7.25) and terminates in one of the end configurations in Figure 7.24, where lines L_1, L_2 are projections of edges which meet at a 3D point P which projects into Q .*

In the strictly positive intersections shown in Figure 7.25(a), the reference point Q lies anywhere in an open half-plane defined by line L (i.e. strictly above or below L , as indicated). It is easy to verify that, in each of the four cases in Figure 7.25(a), if path Π passes from surface S_i to surface S_{i+1} as it crosses the edge, then S_i is behind S_{i+1} at Q . If Q is collinear with L , then the intersection becomes a null intersection. In the null intersections shown in Figure 7.25(b), the reference point Q can lie above, lie below or be collinear with line L , since Π lies on a surface which is either in front of or behind the edge projecting into L .

Definition 7.22 *A path in a wireframe projection is strictly positive if it begins and/or ends at a strictly positive configuration and/or contains at least one strictly positive intersection.*

The **wireframe-path constraint** simply says that strictly positive paths are disallowed in wireframe projections of polyhedral objects.

A path Π is given not by the actual locus of points, but rather by the sequence of line segments it intersects. We define its length to be the total number of lines involved in its start and end configurations and any intersections on Π . The number of paths of length n is an exponential function of n . Clearly, from a practical point of view we must restrict ourselves to applying the wireframe-path constraint to paths of limited length, such as $n = 5$, for example.

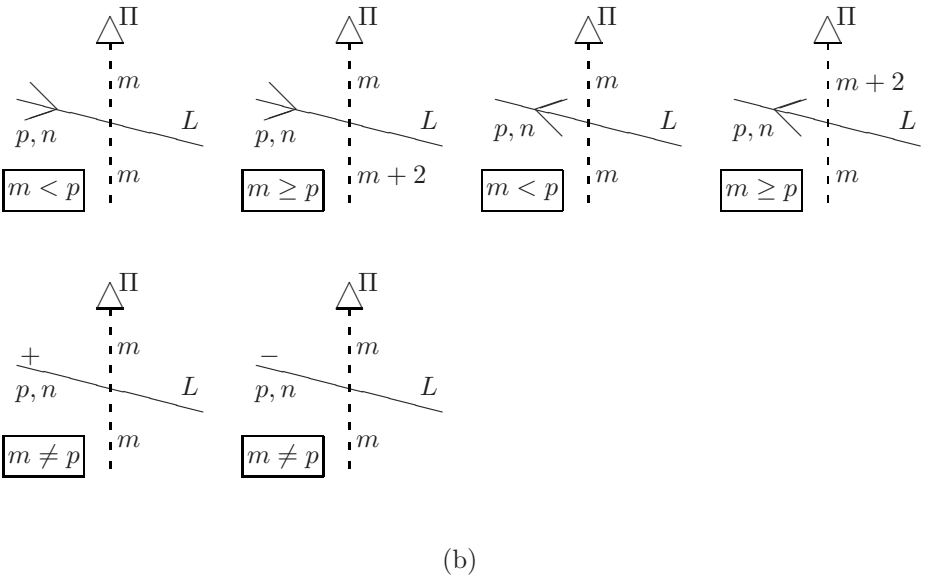
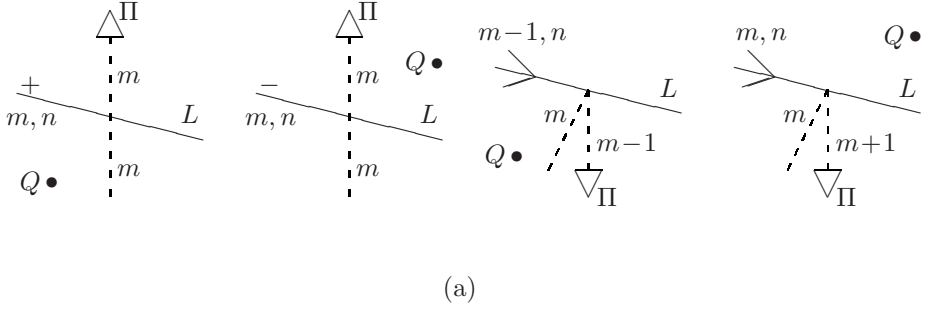


Figure 7.25: (a) Strictly positive intersections; (b) null intersections. The intersections in (a) become null when Q is collinear with L .

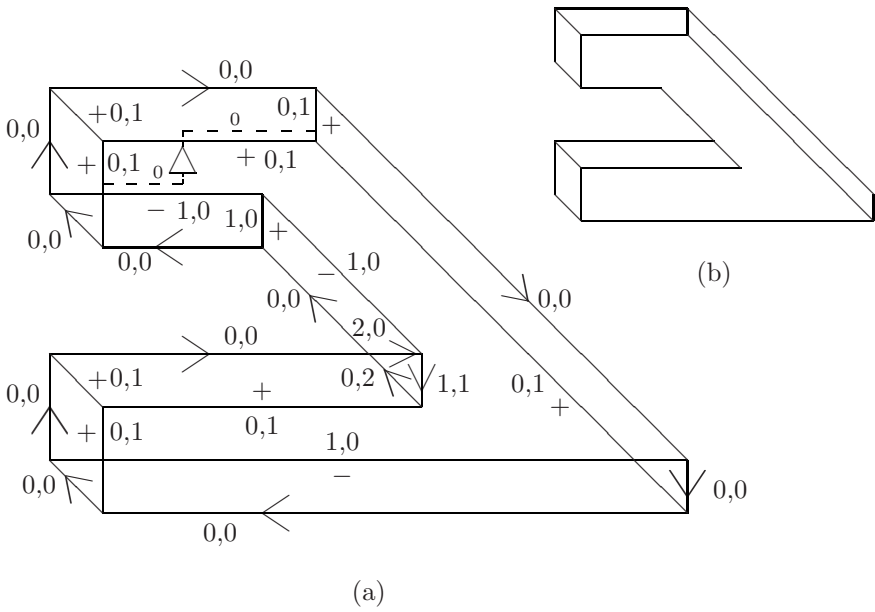


Figure 7.26: (a) A legally labelled wireframe projection; (b) the corresponding solid object (shown half-size).

Figure 7.26(a) shows an example of a legally labelled wireframe projection. Figure 7.26(b) shows only lines at depth $m = 0$ and hence illustrates the solid object corresponding to this labelling. We can see that this labelling is theoretically impossible, under the general viewpoint assumption, since it contains the strictly positive path shown in Figure 7.26(a) as a dashed line.

Consider an instance of the wireframe-path constraint along a path Π , beginning and ending at lines L_1, L_2 , respectively, where L_1, L_2 are postulated to be projections of parallel edges E_1, E_2 . Suppose that $L_1, L_2 \in S$, where S is a set of lines which are postulated to be projections of parallel 3D edges, because, for example, under orthographic projection their angles all lie within the same small interval $[\theta - \epsilon, \theta + \epsilon]$. In the valued constraint framework, this instance of the wireframe-path constraint can be coded as the combination of the strict constraint

$$par_S \Rightarrow \text{the wireframe-path constraint holds on } \Pi$$

and the unary valued constraint which incurs a given finite cost $c > 0$ if $par_S = \text{false}$ (and cost 0 if $par_S = \text{true}$), where par_S is true iff all lines in S are projections of parallel 3D edges. This valued constraint formulation avoids superstrictness problems which result, for example, if we assume that all parallel lines in an orthographic projection are projections of parallel 3D edges. In the case of Figure 7.26(a), it allows us to accept the interpretation shown in Figure 7.26(b) and deduce that the vertical lines in the wireframe projection are not all projections of parallel edges in 3D. This is indeed the most likely interpretation of this wireframe projection. Of course, if $L_1 = L_2$ or if L_1, L_2 are projections of lines which meet at a vertex, then the wireframe-path constraint is a strict constraint.

Our main interest in the wireframe-path constraint lies not in the possibility of detecting or correcting impossible pictures, but rather in the possibility of disambiguating otherwise ambiguous pictures. Consider the wireframe projection in Figure 7.27(a) (adapted from a drawing in Liu et al.'s test set [105]). We see this object as the union of a triangular wedge and a parallelepiped, but other interpretations are possible. The combination of labels shown in Figure 7.27(a) violates the wireframe-path constraint on the path Π shown as a dashed line in the figure (joining two parallel lines). It follows that the wireframe-path constraint invalidates this interpretation, which is illustrated in Figure 7.27(b) (where only visible lines are shown). In the valued constraint formulation, described above, this implies that the interpretation shown in Figure 7.27(b) is non-optimal since it incurs a non-zero cost due to the fact that parallel lines in 2D are not projections of parallel 3D edges. The triangular wedge plus parallelepiped interpretation has zero cost. It is worth noting that if lines AB and CD were not parallel, then this wireframe projection would be genuinely ambiguous with different interpretations having different sets of face circuits. This example shows that it is essential to use parallel lines in the interpretation of wireframe projections if the system is to reliably reconstruct the object actually intended by the user.

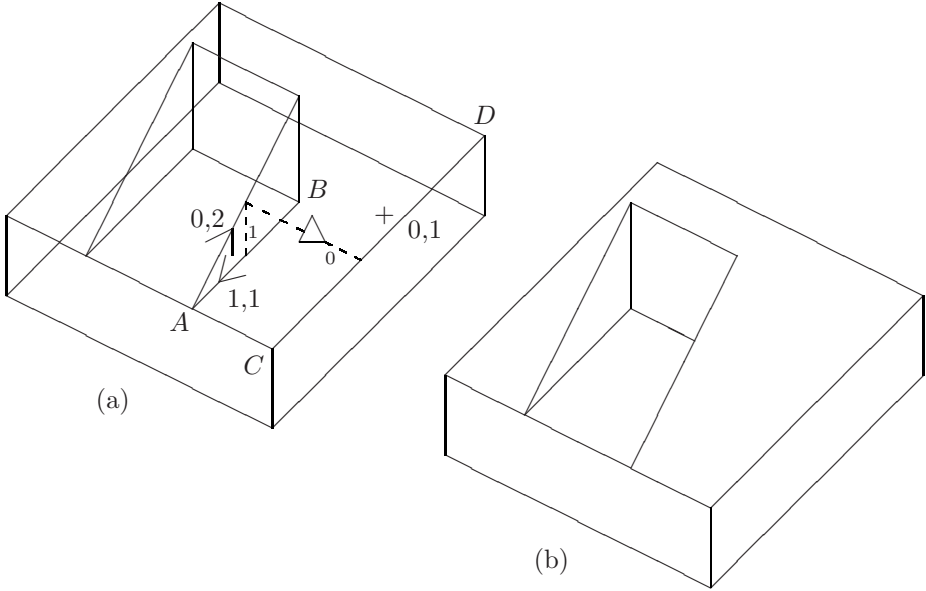


Figure 7.27: Example of the use of the wireframe-path constraint: (a) a strictly positive path which invalidates the interpretation shown in (b).

Our interest in constraints between distant lines lies in their ability to disambiguate otherwise ambiguous pictures such as in Figure 7.27(a). This is particularly important when we enlarge the class of objects to include objects with tetrahedral vertices, as we will see in the following section.

7.11 Tetrahedral Vertices

In Figures 7.6–7.8 we gave the catalogue of legally labelled trihedral junctions. In this section we extend this catalogue to include all vertices formed by the intersection of four edges (some of which may be collinear) and four planar faces F_1, \dots, F_4 . We assume that the object is a 3D manifold and that its surface S is manifold in the neighbourhood of vertex V . This means that the intersection of S with a sphere centred at V and of radius ϵ forms a simple closed curve C . We can classify the vertices into different types according to the topology of the projection C' of C in the image plane. The *signature* of a projection of a vertex tells us where C' turns back on itself (to form an occluding line) and the relative depth order of the faces F_1, \dots, F_4 . By exhaustive search, using the simple constraints that faces intersect only at edges and that a planar face cannot overlap itself, it is simple to discover that there are essentially only eleven non-equivalent signatures for tetrahedral vertices. These are shown in Figure 7.28. We consider two signatures to be *equivalent* if they can be put into correspondence by some combination of rotation, reflection and depth reversal.

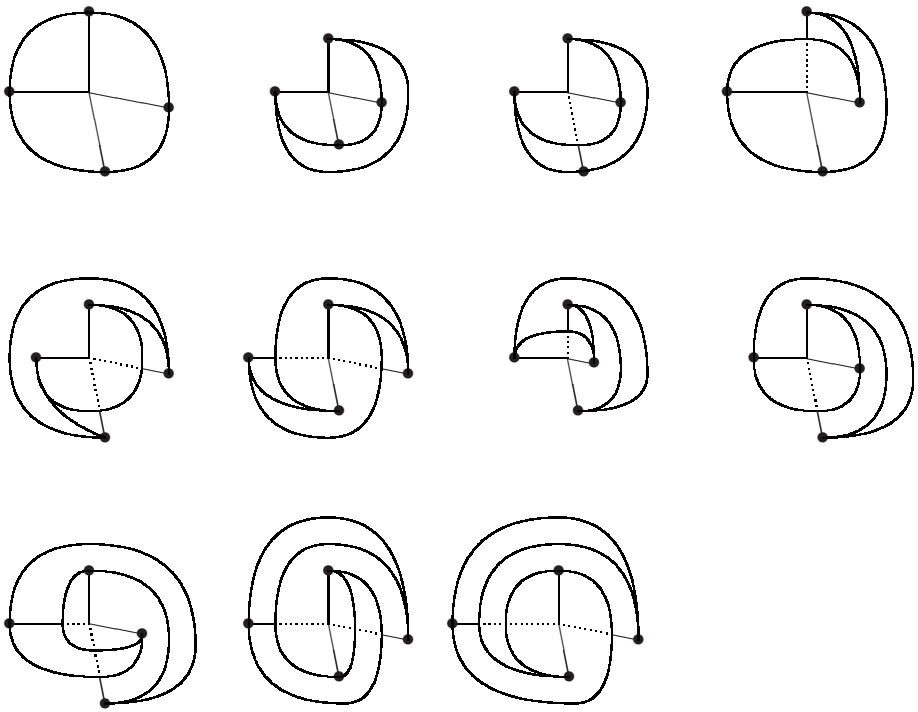


Figure 7.28: The eleven non-equivalent legal signatures of tetrahedral junctions.

Each signature may correspond to a number of different junction labellings. The junction type may be Multi, Peak, K, Ψ or X depending on the angles between the four lines [43, 165]. Furthermore, each non-occluding edge may be either concave or convex. However, starting from the list of 11 non-equivalent signatures in Figure 7.28 allowed us, by hand, to find all legally-labelled junctions formed by the wireframe projection of tetrahedral vertices. Given any two faces F_i, F_j which meet at a vertex, their projections in the picture plane cannot overlap over an angle greater than π (otherwise F_i and F_j would pass through each other, thus creating an edge which is not 3D-manifold). Those labellings which did not satisfy this property or the wireframe-path constraint were eliminated. It turned out that each of the remaining labellings was realizable, which we checked by hand by constructing a vertex which projected into the actual labelled junction.

Huffman [75] and Clowes [20] used a different exhaustive technique to establish the catalogue of labelled projections of trihedral vertices: at the vertex, space is divided into eight octants which may be filled with matter or be empty; each manifold vertex can then be viewed from each of the eight octants. The technique we employed for tetrahedral vertices, based on signatures, requires exhaustion over a smaller number of cases than this traditional technique.

Figures 7.29 and 7.30 give the resulting catalogue of labelled junctions which are projections of tetrahedral vertices. To obtain the complete catalogue of labelled junctions we must not forget to add all labellings which can be obtained by any combination of rotation, reflection and depth-reversal. This means that the number of distinct legal labellings for Multi, Peak, K and Ψ junctions are, respectively, 70, 56, 24 and 24 (once the numerical label m, n of one line is fixed). In the analysis of wireframe projections, allowing tetrahedral vertices does not increase the number of legal labellings for W, Y, 2-tangent, 3-tangent, X and snowflake junctions.

The construction of a catalogue of legal labellings could be automated for higher-order vertices. We briefly describe an effective procedure for doing so. Consider a vertex V at which N faces meet. Suppose that V is an isolated vertex in the sense that no other faces lie in front of or behind V . Then the total number of faces which lie in front of or behind any of the edges E meeting at V is clearly bounded above by $N - 1$. The numerical label m_E, n_E for the projection of E thus satisfies $0 \leq m_E + n_E \leq N - 1$. Hence there is only a finite number of possible numerical and semantic labellings which need to be checked for validity, for each different junction type involving N lines. (Generic numerical labellings involving two variables m and n , representing the number of surfaces in front of and behind V , are easily obtained by adding m, n to the numerical label of each edge E meeting at V .) We described in [42] how to uniquely identify faces given a labelled wireframe projection:

1. For each region R in the drawing, calculate, from the semantic and numerical label of one of the lines bounding R , the number N_R of surfaces projecting into R .
2. Generate an imaginary stack $Stack(R)$ of N_R copies of region R .

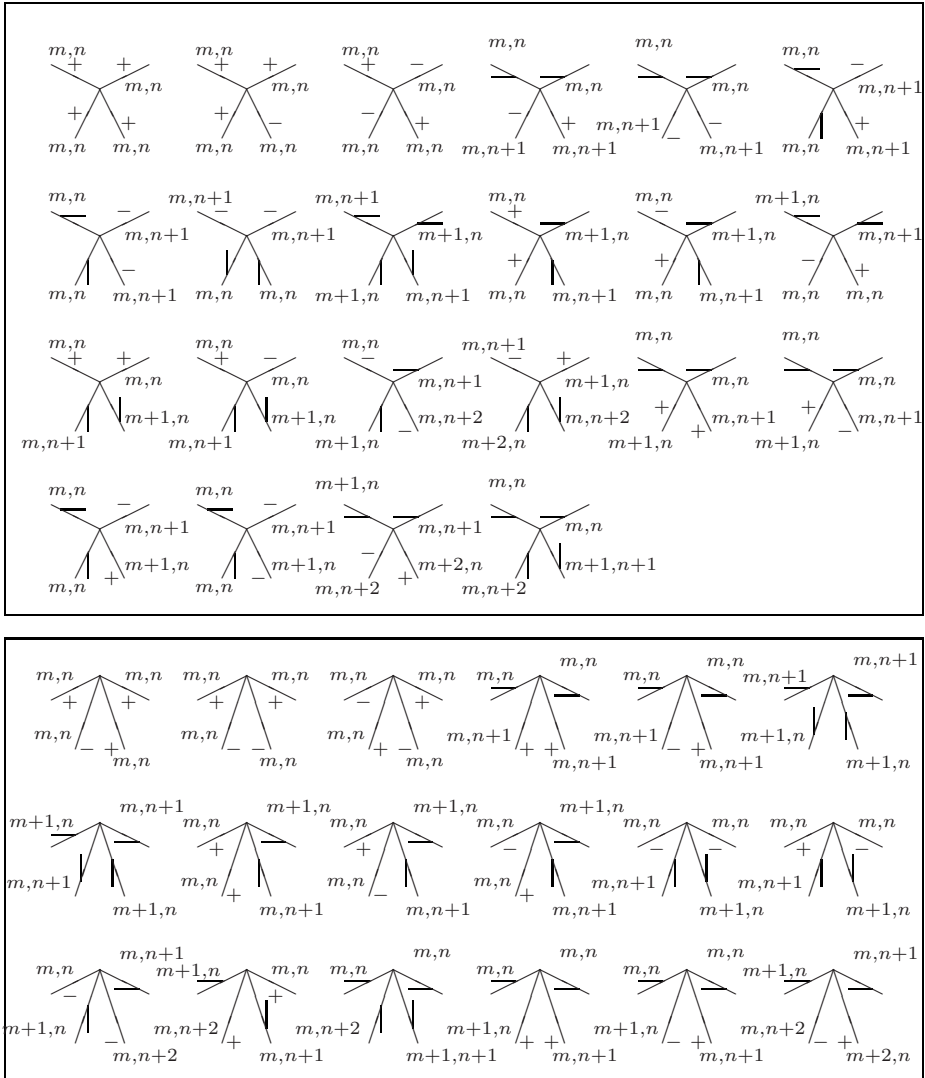


Figure 7.29: Catalogue of Multi and Peak junction labellings (to which must be added all labellings obtained by reflection and/or depth reversal).

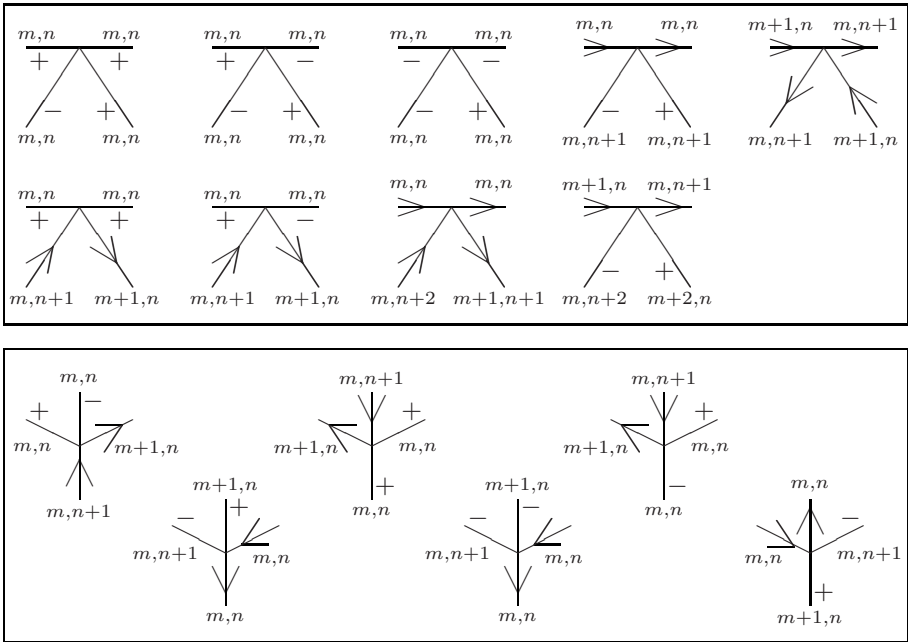


Figure 7.30: Catalogue of K and Ψ junction labellings (to which must be added all labellings obtained by reflection and/or depth-reversal).

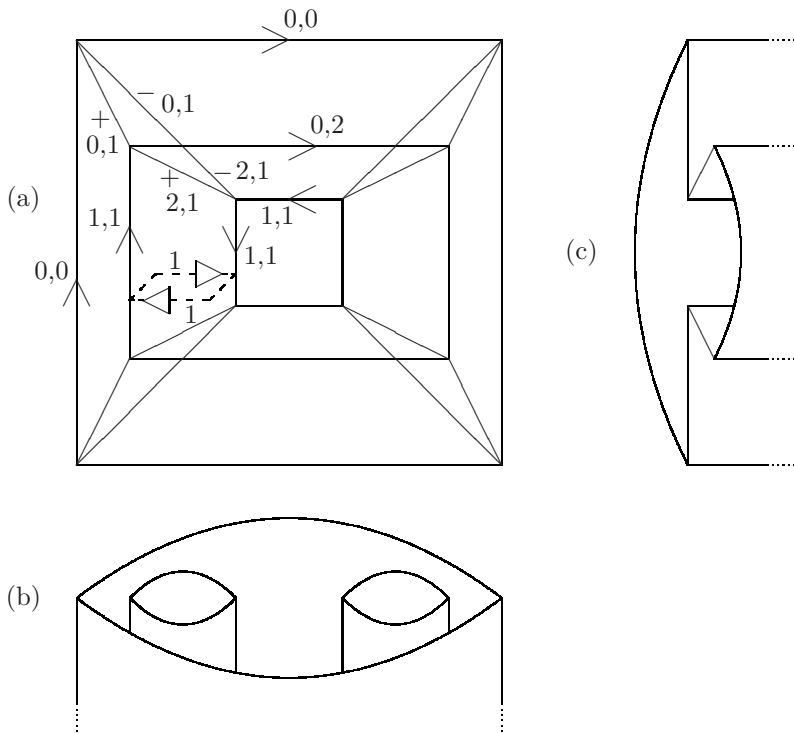


Figure 7.31: (a) A labelling which is illegal by the wireframe-path constraint; (b),(c) horizontal and vertical cross-sections through the middle of a curved object corresponding to this labelling.

- For each line L (which without loss of generality we suppose is vertical) with numerical label m, n separating a pair of regions R_{left}, R_{right} : if the semantic label of L is $+$ or $-$, then join the depth- m surfaces in $Stack(R_{left})$ and $Stack(R_{right})$; if the semantic label of L is \uparrow or \uparrow , then join the depth- m and depth- $(m+1)$ surfaces in $Stack(R_{right})$; if the semantic label of L is \downarrow or \downarrow , then join the depth- m and depth- $(m+1)$ surfaces in $Stack(R_{left})$.

This technique can clearly be applied to the wireframe projection of vertex V . The validity of the resulting face structure can then be checked by Sugihara's classic linear programming technique for drawings of polyhedral objects [155]. Assuming that object edges and surfaces are C^3 and that they meet nontangentially at V , in the neighbourhood of V the object can be approximated by the polyhedral vertex formed by the tangent planes to its surfaces. Furthermore, no problems of superstrictness can occur in a drawing of a single isolated vertex.

Definition 7.23 *Two semantic and numerical labellings of a wireframe projection are topologically equivalent if the set of faces they define are identical.*

For example, a labelling and its depth reversal are necessarily topologically equivalent. More generally, changing some semantic labels from $+$ to $-$ (and some from $-$ to $+$) does not alter the faces of an object but may not produce a legal labelling.

Figure 7.31(a) shows a typical example of a wireframe projection of a planar-faced object with tetrahedral vertices. There are several legal labellings of this wireframe projection which satisfy both the junction constraints and the wireframe-path constraint. These different labellings are all topologically equivalent since they all represent a torus with a triangular cross-section. If we do not apply the wireframe-path constraint, then many more labellings become legal. Each of these labellings can be physically realized as a possible but highly improbable object with curved surfaces. For example, the labelling given in Figure 7.31(a) is physically realizable as an object with curved surfaces whose horizontal and vertical cross-sections are shown in Figure 7.31(b) and (c), respectively. If we assume, on the other hand, that all faces are planar, then we can apply the wireframe-path constraint: a strictly positive wireframe path is shown as a dashed line in Figure 7.31(a). We can see from this example that the wireframe-path constraint applied to paths involving only two lines allows us to impose the well-known geometrical constraint that two distinct planar faces cannot share two non-collinear edges. Moreover, we can apply this constraint even to faces with holes, which is not the case of systems based on finding face circuits [106].

7.12 Tangential Edges and Surfaces

Another way of relaxing our assumptions on object shape is to allow object edges and/or surfaces to meet tangentially. In the case of visible-line drawings, a complete catalogue of labelled junctions has already been drawn up [34]. This section gives a similar catalogue for wireframe projections.

When two surfaces intersect or merge along a simple curve, we call this a TIC (tangential intersection curve). In order to avoid studying increasingly large numbers of highly unlikely vertices, we consider only vertices formed by the intersection of at most three surfaces S_1, S_2, S_3 and disallow unnecessary coincidences (such as a TIC itself being tangential to another edge). As proved in [34], this leaves five new types of vertices, where E_{ij} represents the intersection of the surfaces S_i, S_j :

1. E_{ij} tangential to S_k for all $i, j, k = 1, 2, 3$, and E_{ij} tangential to E_{ik} for all $i, j, k = 1, 2, 3$.
2. S_1 kisses S_2 at a point.
3. S_1 tangential to S_2 , and they also intersect along another edge (which may or may not be a tangential edge).

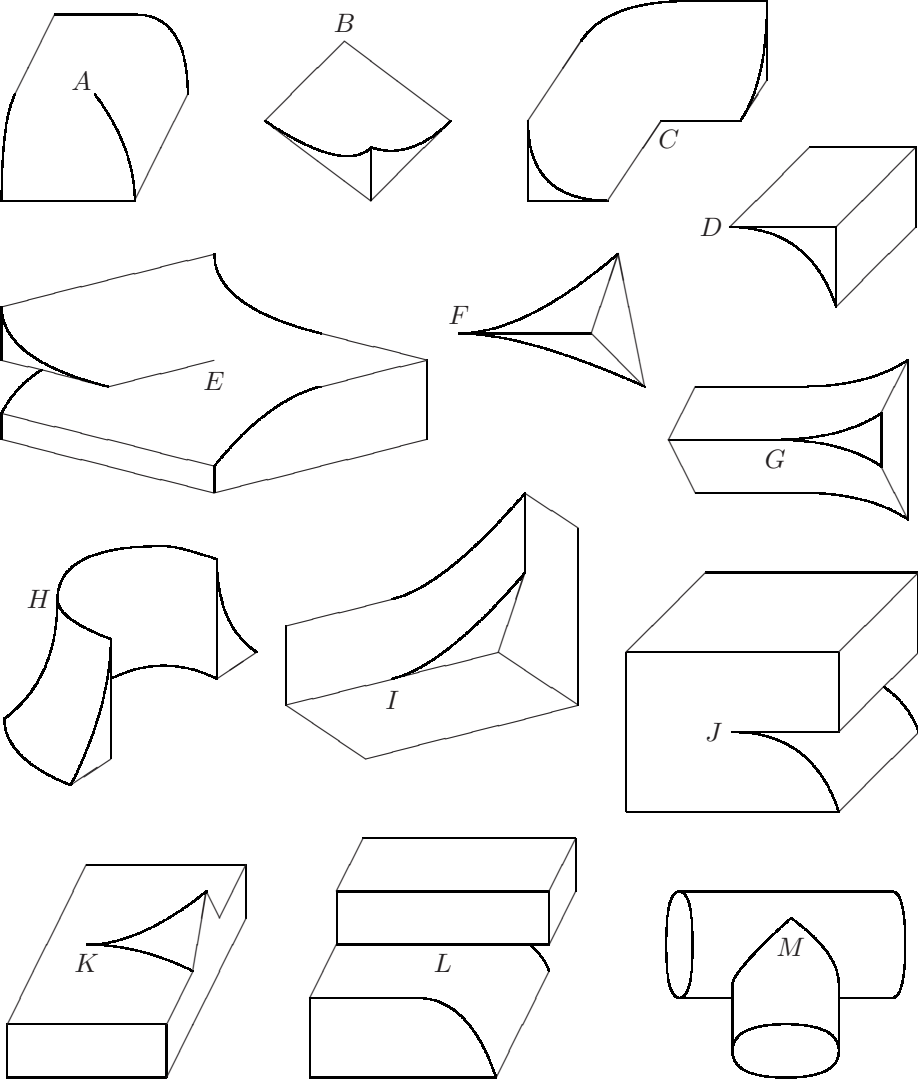


Figure 7.32: Examples of vertices formed by tangential edges and/or surfaces.

4. S_1 tangential to S_2 , and their TIC intersects S_3 .
5. S_i tangential to S_j for all $i, j = 1, 2, 3$.

Junctions can also be formed by the intersection of a TIC with an extremal edge: in other words, the two surfaces meeting at the TIC are tangential to the viewing direction.

Examples of type 1 vertices, formed when three edges are all tangential, include vertices F, G, I, K in Figure 7.32. Examples of type 2 vertices, formed when two surfaces kiss, include vertex M of Figure 7.32 and vertex P of Figure 7.18. Examples of type 3 vertices, formed when two surfaces are tangential and also intersect along another edge, include vertices B, C of Figure 7.32. Examples of type 4 vertices, formed when a surface intersects the tangential intersection curve of two other surfaces, include vertices D, J, L of Figure 7.32. Examples of type 5 vertices, formed when three surfaces are tangential, include vertices A, E of Figure 7.32. Vertex H in Figure 7.32 is an example of a viewpoint-dependent vertex formed when the tangential intersection curve of two surfaces is tangential to the viewing direction.

By exhaustion over all possible viewpoints, for each possible vertex of each different type described above, we obtained the list of junction labellings given in Figure 7.33. For brevity of presentation, we have not included in this figure those labellings that can be obtained from labellings in Figure 7.33 by any combination of rotation, reflection and depth reversal. For example, a terminal junction may also be labelled $-$, which is obtained by depth reversal of the labelling given in Figure 7.33. At a terminal junction a single line terminates. At a 4-tangent junction, there is no discontinuity of curvature between the two viewpoint-independent lines (labelled \rightarrow and \leftarrow), but there is a discontinuity of curvature between these two lines and the two extremal lines. At Y_0, W_0 and W_{00} junctions, the 0 indicates an angle of zero degrees between the tangents to two lines meeting at the junction. As pointed out in [34], at a Y_0 junction either all three lines have identical curvature or there is a discontinuity of curvature between each pair of lines; therefore, in theory, there is no possible confusion with a 3-tangent junction. In practice, of course, the legal labellings of a 3-tangent junction as shown in Figure 7.8, its reflected version and a Y_0 junction should all be merged. In the valued constraint formulation of the problem, we can penalize each such misclassification of a junction by some small positive cost.

When two lines meet at a junction J and have different curvatures, J is known as a curvature-L junction [109]. When two face patches of different curvatures meet along a tangential intersection curve, they create what is known as a smooth edge. In wireframe projections of objects with smooth edges, discontinuities of curvature can occur on any type of line (concave, convex, occluding or extremal), but this does not provoke a label transition. In other words, the two lines meeting at a curvature-L junction have identical semantic and numerical labels; it is for this reason that we do not need to include curvature-L junctions in our catalogue of labelled junctions.

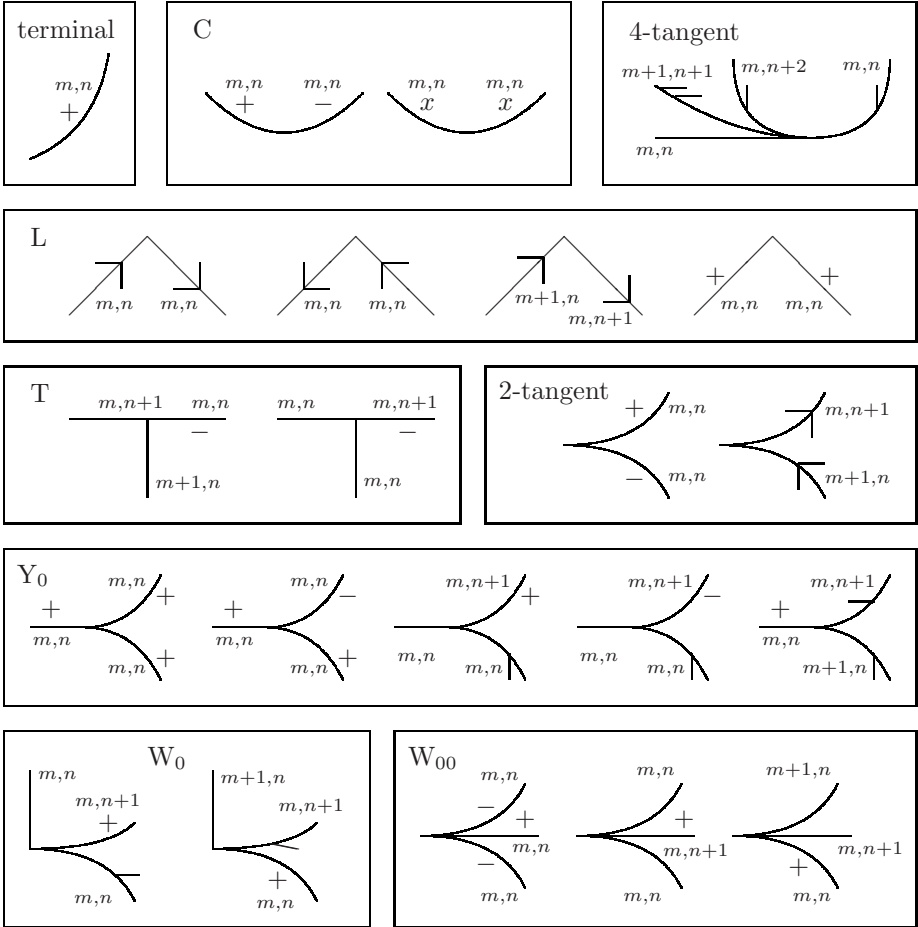


Figure 7.33: Labeled wireframe projections of vertices involving tangential edges and/or surfaces (to which must be added all labellings obtained by any combination of rotation, reflection and depth reversal). In the labellings of a C junction, x is any semantic label.

It is interesting to note that Y_0 , W_0 and W_{00} junctions do not have the same set of labellings as Y or W junctions in the standard trihedral catalogue given in Figure 7.6. Another important point is that, apart from 2-tangent junctions, none of the junctions in Figure 7.33 can occur in wireframe projections without tangential edges and/or surfaces. As with the tetrahedral catalogue, this catalogue therefore tends to complement, rather than dilute, our trihedral catalogue. Unfortunately, due to vertices created by two surfaces kissing at a point, as illustrated in Figure 7.18, transitions between convex and concave labels can occur at any point on any curved line. Fortunately, the straight-edge formation assumption (which says that all straight edges in 3D are formed by the intersection of locally planar surfaces) in conjunction with the general viewpoint assumption, excludes $+/-$ transitions on straight lines. By expressing the labelling problem as a valued constraint satisfaction problem, we can assign a cost $c > 0$ to each $+/-$ transition in order to find the labelling with the least number of such transitions.

7.13 Rich Labelling Scheme

In Chapter 5 we described a rich labelling scheme for visible-line drawings, in which the aim is not only to assign semantic labels to lines but also to identify locally planar surfaces as well as their gradient directions. This section extends this labelling scheme to wireframe projections. The basic principles remain the same, but certain constraints have to be adapted since in wireframe projections all lines are visible.

The planarity constraints, given in Figure 5.5, still hold except for the constraints involving C junctions and curvature- L junctions since these two types of junctions do not occur in wireframe projections. In the rest of this section we assume that drawings have been produced by orthographic projection. If a plane has equation

$$z = px + qy + r,$$

then (p, q) is known as its *gradient* [107]. It is well known that gradient space analysis provides necessary but not sufficient conditions for a drawing to be realizable as an orthographic projection of a polyhedral scene, due to the fact that the third parameter r is ignored [155]. The *gradient direction* of a plane P is the direction in the image plane of the projection of the normal to P . It is given by $\tan^{-1}(\frac{q}{p})$ and is hence even less informative than the gradient (p, q) . Although gradient directions provide an incomplete description of planes, we have shown in Chapter 5 how they can be incorporated into a discrete labelling scheme to enrich the classical semantic labels (convex, concave, occluding, etc.).

The gradient-direction constraints given in Chapter 5 have to be rewritten, since in wireframe projections all three lines are always visible at a viewpoint-dependent vertex. The new constraints are given in Figure 7.34. These constraints only hold under the assumption that the surface-normal discontinuity edges L_1, L_2 are orthogonal. The top four constraints in Figure 7.34 simply translate the following simple geometrical observation: if three straight edges

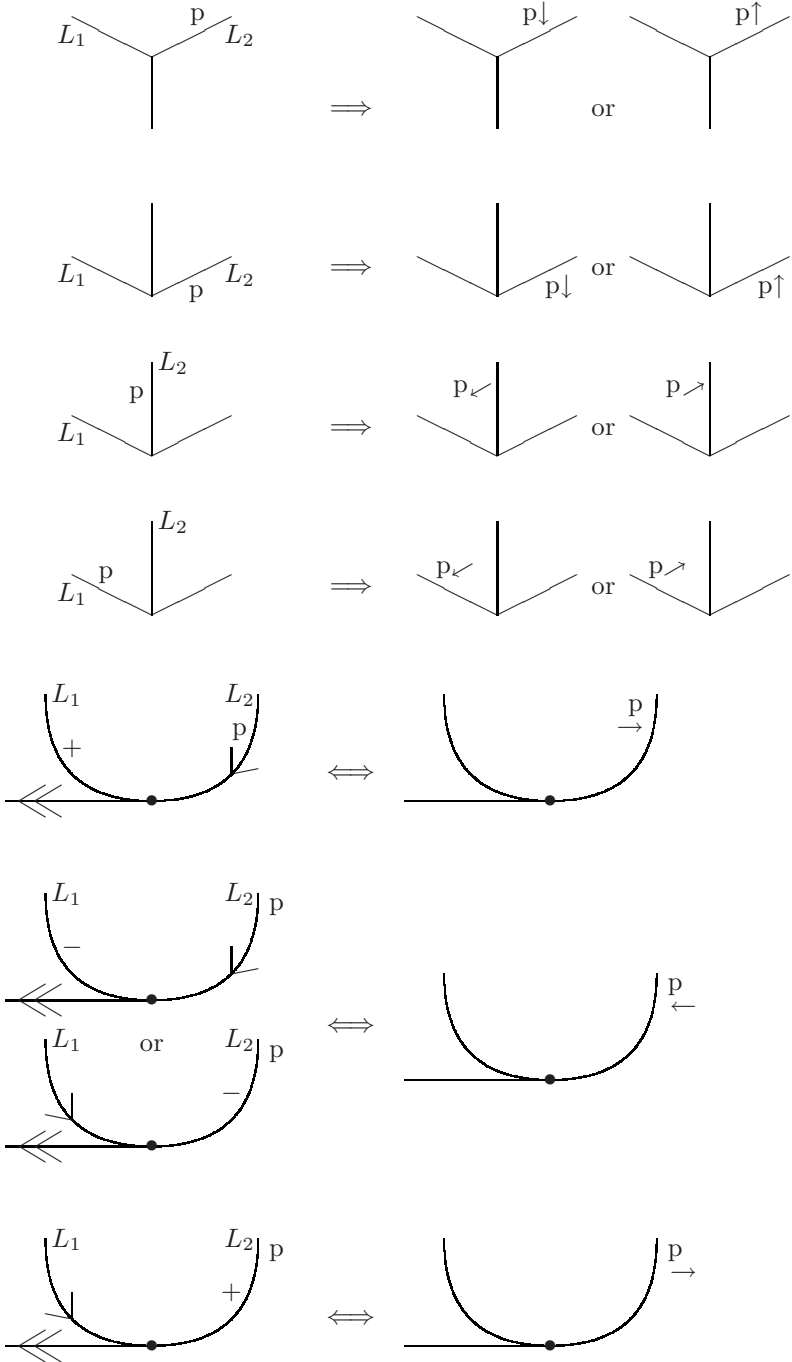


Figure 7.34: Gradient-direction constraints in wireframe projections assuming that the surface-normal discontinuity edges L_1 , L_2 are orthogonal. The \Leftarrow implications are consequences of the general viewpoint assumption.

E_1, E_2, E_3 meet at a vertex V , where edges E_1, E_2 are orthogonal, then the normal \mathbf{n} at V to the surface S passing through E_1, E_2 is parallel to E_3 . We symbolize the gradient direction (the direction of the projection of \mathbf{n}) by a short arrow next to the ‘p’ on the right-hand side of Figure 7.34. The constraints still hold even if any number of the three lines L_1, L_2, L_3 meeting at the junction are curved (under our assumptions that object surfaces are piecewise C^3 and that edges and surfaces meet non-tangentially), in which case E_1, E_2, E_3 represent the tangents to the edges at V .

The bottom three constraints shown in Figure 7.34, reading the implication from left to right, all concern a labelled 3-tangent junction formed by the projection of a curved orthogonal edge E which is the intersection of a curved surface S_c with a planar surface S_p . Let P represent the 3D point on E at which the tangent to S_c passes through the viewpoint, and let T_P represent this tangent. If \mathbf{n} is the normal to the planar surface S_p , then by orthogonality of E , \mathbf{n} is parallel to T_P . It follows that the projection of \mathbf{n} in the drawing is parallel to the extremal edge (which is the projection of T_P). Note that, since the lines L_1, L_2 are projections of the same edge E , planarity and gradient-direction labels propagate through 3-tangent junctions.

Each side of each line L has a label representing the gradient direction of the corresponding surface. Consider a line L joining two junctions J_1, J_2 . The gradient direction of each side of L is either parallel to (the tangent to) one of the lines meeting at J_1 or J_2 , or otherwise is assigned the label ‘other’. Hence, gradient directions also lie in finite domains and can be incorporated into our valued constraint satisfaction formulation and found by complete intelligent branch-and-bound search [115].

The gradient-direction/semantic-label constraints, given in Figure 5.8, remain valid for wireframe projections. Perkins [130] observed that only certain Y and W junctions can be projections of cubic corners (trihedral vertices at which three surfaces meet orthogonally). The most succinct statement of Perkins’s rules is that when the three lines meeting at a Y or W junction J (a projection of a cubic corner) are extended through J , then no three of the resulting six half-lines lie within an angle of $\frac{\pi}{2}$ [103]. When we try to interpret the three lines meeting at such a junction J as projections of orthogonal edges, the combination of the gradient-direction constraints and the gradient-direction/semantic-label constraints leads to a contradiction. Hence by applying our constraints we automatically apply Perkins’s rule.

As in the case of visible-line drawings, each of the constraints given in Figures 5.5, 7.34, 5.8 is a strict constraint. They must be combined with soft unary constraints expressing a preference for locally planar surfaces, planar faces and orthogonal edges. For example, each ‘c’ line label and each non-orthogonal line should be penalized by a non-zero cost.

We now demonstrate how it is possible to apply the wireframe-path constraint to drawings of curved objects by stipulating that path Π lies in locally planar surfaces.

Definition 7.24 *Let ϵ be a distance which is small compared to all line lengths*

in a wireframe projection. A face-circuit fragment in a semantically and numerically labelled wireframe projection is a depth-labelled locus of points Π , such that:

1. Every point P of Π lies at a distance ϵ from some line L_P . If L_P is labelled $(+, m, n)$ or $(-, m, n)$, then the depth-label m_P of Π at P satisfies $m_P = m$ and Π lies either to the right or left of L_P ; if L_P is labelled (\uparrow, m, n) (respectively (\downarrow, m, n)), then $m_P \in \{m, m + 1\}$ and Π lies to the right (left) of L_P .
2. All intersections on Π are null intersections, of the form shown in Figure 7.25(b).

A *face circuit* is a face-circuit fragment which is a closed curve. A face-circuit fragment Π is *locally planar* if at every point P of Π the object surface at depth m_P is locally planar at L_P . Note that when line L_P is labelled (\uparrow, m, n) and the depth label of Π at P is $m + 1$, then, by our convention, the planarity label ‘p’ is written on the opposite side of line L_P to Π . In all other cases, the planarity label ‘p’ lies on the same side of L_P as Π .

A face-circuit fragment Π follows a sequence of lines in a wireframe projection. Let L_1, L_2 be two consecutive lines in this sequence. They necessarily meet at a junction J . Let E_1, E_2 be the edges which project into L_1, L_2 , let V be the vertex which projects into J and let S be the surface on which E_1 and E_2 both lie. Suppose that, for $i \in \{1, 2\}$, S has a constant tangent plane T_i along the length of E_i (i.e. S is locally planar along E_i). Assuming that S is C^3 at V , and that E_1, E_2 are not tangential, the tangent planes T_1, T_2 must be identical. We can now state a constraint which allows us to propagate gradient-direction labels.

Gradient-direction propagation constraint: In an orthographic wireframe projection of an object composed of C^3 surfaces/edges meeting non-tangentially, gradient-direction is constant along a locally planar face-circuit fragment Π .

The wireframe-path constraint, as stated earlier in Section 7.10, only applies to drawings of polyhedral objects. We can easily generalize it to drawings of objects with some curved faces (involving only C^3 surfaces/edges meeting non-tangentially) by stipulating that path Π must lie on locally planar surfaces. Path Π must be a sequence of locally planar face-circuit fragments joined together at strictly positive or null intersections.

Figure 7.35(a) and (b) are two examples of wireframe projections of objects involving some curved and some planar surfaces. Applying the outer-boundary constraint and the catalogue of legal labellings provides a unique semantic and numerical labelling (modulo depth reversal) for both of these drawings. These labellings are shown in Figure 7.35(a),(b). Also applying the planarity constraints, the gradient-direction constraints, the gradient-direction/semantic-label constraints and the gradient-direction propagation constraint allows us to determine the optimal interpretations of these two drawings. Figure 7.35(a) has two equally likely interpretations, shown in Figure 7.35(c),(e). In order to

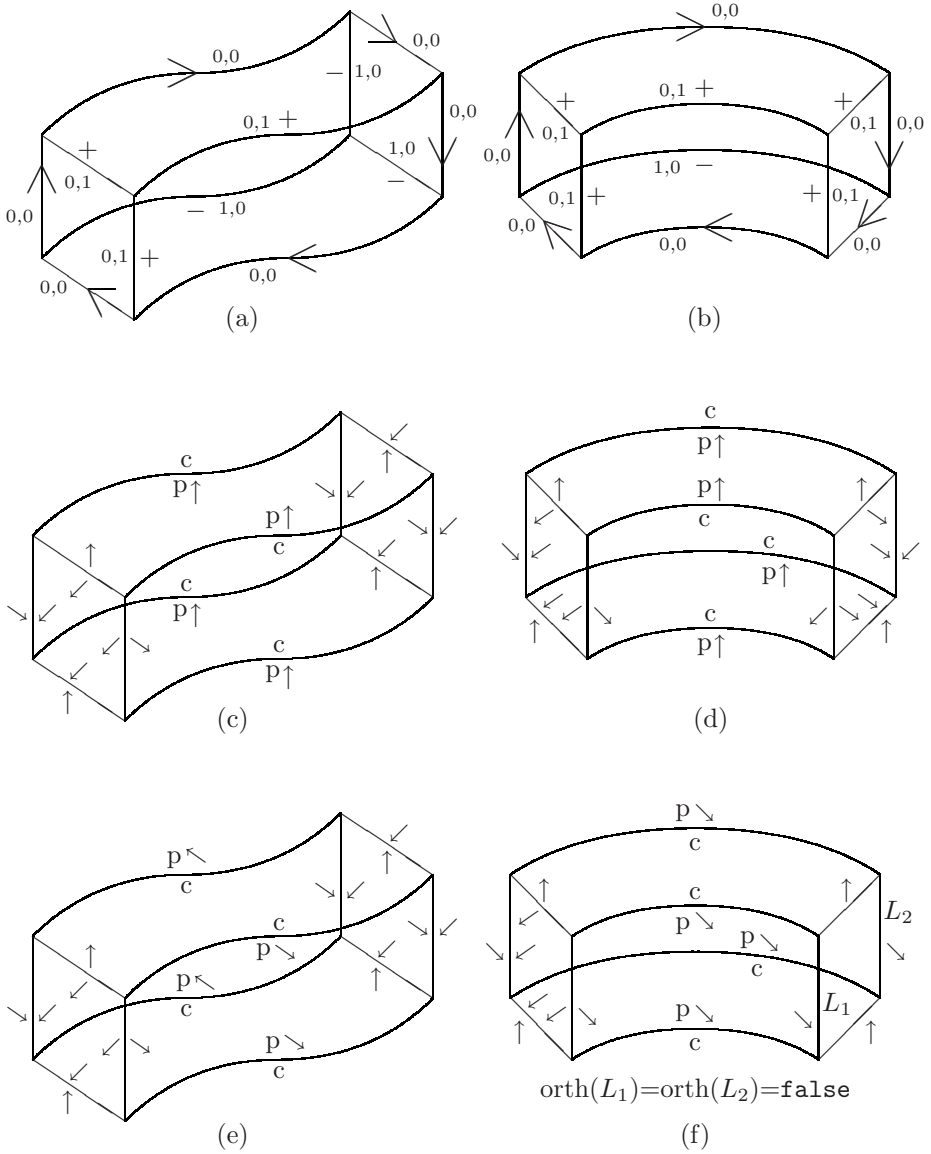


Figure 7.35: (a), (b) Two wireframe projections with their semantic and numerical labellings; (c), (e) two equally likely interpretations of the wireframe projection in (a); (d), (f) two interpretations of the wireframe projection in (b), (d) being the most likely since all edges are orthogonal.

avoid cluttering up the figures, we have omitted ‘p’ labels on straight lines. In both interpretations (c) and (e), all edges are orthogonal, the ambiguity lying in whether it is the top and bottom faces which are planar or the front and back faces. Figure 7.35(b) has a unique optimal labelling, shown in Figure 7.35(d), in which all edges are orthogonal. Figure 7.35(f) is an example of a non-optimal interpretation, involving two non-orthogonal edges. In (d) the top and bottom faces are planar, whereas in (f) it is the front and back faces which are planar.

7.14 Discussion

We tested our labelling scheme on the 11 wireframe projections in Liu et al.’s test set [105]. We applied the outer-boundary constraint so that all labels on the outer boundary of the drawing were $(\uparrow, 0, 0)$ or $(\uparrow, 0, 0)$. To avoid depth-reversal ambiguity, we pruned the set of possible labellings of one junction on the outer boundary so that it contained no pair of labellings such that one was the depth reversal of the other. We prefer interpretations involving as much regularity and structure as possible, since these are typical properties of objects which are likely to be depicted in the drawing. Thus

- Each ‘c’ label incurs a cost c_1 , in order to maximize the number of locally planar surfaces in the interpretation;
- Each line L with $\text{orth}(L) = \text{false}$ incurs a cost c_2 , in order to maximize the number of orthogonal edges;
- Each $+/-$ semantic label transition between the two ends of a line L incurs a cost c_3 ;
- Each $\text{par}_S = \text{false}$ (meaning that the set of lines S which are parallel in the drawing, to within a certain tolerance, are not projections of parallel 3D edges) incurs a cost c_4 .

The choice of the relative values of c_1, c_2, c_3, c_4 was not found to be critical, since different types of constraints rarely enter into conflict with each other. We therefore arbitrarily chose $c_1 = c_2 = c_3 = c_4 = 1$ in our trials.

We found that for polyhedral objects with only trihedral vertices, the junction catalogue was sufficient to produce an unambiguous labelling. However, for objects involving tetrahedral vertices, the wireframe-path constraint was necessary in order to avoid highly unlikely labellings as illustrated in Figure 7.31. Although in each case the tetrahedral junction catalogue, together with the wireframe-path constraint, was sufficient to unambiguously determine face circuits, much ambiguity nevertheless remained, particularly in deciding whether edges were concave or convex.

As an illustration of this, consider the wireframe projection shown in Figure 7.36(a). Applying the wireframe-path constraint allowed us to correctly identify all faces of the depicted object. It also allowed us to deduce that if CG is convex then EF is concave. However, even assuming that CG is convex, all of

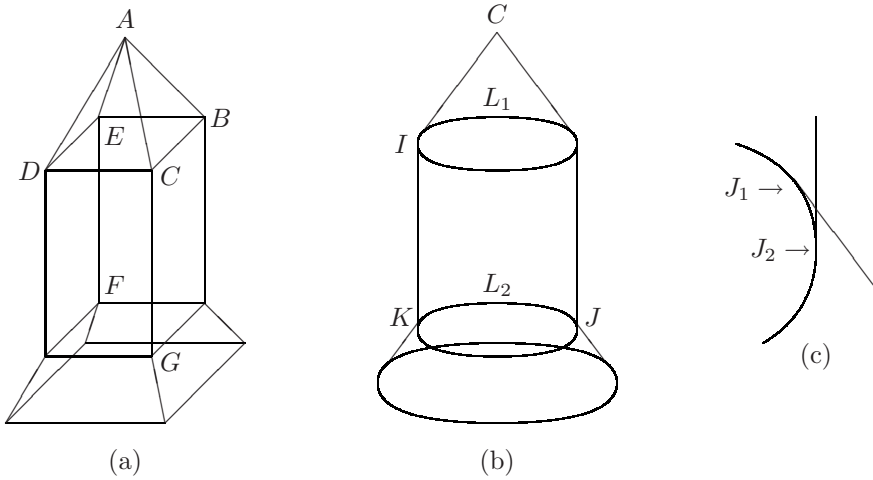


Figure 7.36: (a),(b) Examples of wireframe projections; (c) a close-up of junction J in (b).

the lines AE, DE, EB, BC, CD could be either concave or convex due to the fact that the depth of A relative to the rest of the object is unknown. Liu et al. [106] found Marill’s Minimum Standard Deviation of Angles [111] very effective in finding the correct interpretation of this drawing. An alternative approach which would work very well in this example is the search for symmetries [132].

Figure 7.36(b) is a circular version of the object depicted in Figure 7.36(a). We need to make two important changes to our labelling scheme in order to be able to interpret this drawing. Firstly, vertex H is the apex of a cone and hence is disallowed by our assumption that object faces are C^3 surfaces, since surface curvature is undefined at H . We can nonetheless easily accommodate apices of cones by adding a (\uparrow, \downarrow) labelling to the list of legal labellings of L junctions given in Figure 7.33. Secondly, junctions I and J do not appear in our catalogue. This is not surprising, since both I and J involve a pair of 3-tangent junctions which are so close together in the drawing that they appear to be a single junction. Figure 7.36(c) shows a close-up of J , which shows that what we see as a single junction J in Figure 7.36(b) is in fact a triangle composed of two 3-tangent junctions J_1, J_2 and one X junction. In freehand sketches, pairs of 3-tangent are often merged in this way. Any 4-line junction at which two tangential lines appear to meet two other tangential lines can simply be decomposed into

- Two 3-tangent junctions if the junction is a Peak-type junction (such as I in Figure 7.36(b)),
- Two 3-tangent junctions and an X junction, as shown in Figure 7.36(c), if the junction is a Multi-type junction.

The wireframe-path constraint provides no useful information when applied to the drawing in Figure 7.36(b) since most surfaces of the object depicted are curved. There is no ambiguity in terms of face identification, but our junction catalogue does not allow us to recognize that L_1 is concave if and only if L_2 is convex. The independent depth reversal of the top and the bottom of this drawing can be avoided by augmenting our rich labelling scheme with a relative-depth label for straight extremal edges, as described in Chapter 4 for polyhedral objects. Assuming that L_1, L_2 are projections of orthogonal edges, if I is nearer the viewer than K , then L_1 is concave and L_2 convex, otherwise L_1 is convex and L_2 concave.

7.15 Conclusion

A wireframe projection provides a convenient means of representing a 3D object from a single view. However, it is well known that, even given complete depth information about lines, a wireframe model of a polyhedron is ambiguous. We have shown, however, that a 3D wireframe model of a polyhedron with simple trihedral vertices is unambiguous.

When a wireframe projection is human-entered, it is important to test its physical realizability. In the case of curved objects, we have given necessary and sufficient conditions for the physical realizability of a wireframe projection. However, a wireframe projection of an object with curved surfaces and tetrahedral vertices often has a large number of physically realizable but highly unlikely interpretations. This has led us to the introduction of constraints between distant lines based on planarity and the extension of traditional line labels to include extra local information.

Our notion of interpretation is thus based on a labelling scheme involving

- The number of surfaces in front of and the number of surfaces behind each edge;
- The classification of edges as convex, concave, occluding or extremal;
- The identification of locally-planar surfaces;
- The identification of orthogonal edges;
- The identification of surface-normal directions of locally planar faces.

The set of constraints presented in this chapter is not exhaustive. For example, under perspective projection, the identification of vanishing points provides strong constraints on the labellings of trihedral junctions [127], and under orthographic projection, constraints exist between pairs of junctions involving parallel lines [43].

A possible extension of the labelling scheme would be to allow extra lines representing discontinuities of surface curvature, as has already been done for line drawings of opaque objects [32, 34]. However, perhaps the most important

future challenge remains the selection of the most likely 3D curved object among the infinite family of objects which project into a given wireframe. A rich labelling (as described in this chapter) is a good starting point but only provides an incomplete description of a curved 3D object. We revisit the problem of complete 3D reconstruction in Chapter 10.

Chapter 8

Simplification of Combinatorial Problems

8.1 Transformations of Combinatorial Problems

When faced with an instance I of a theoretically intractable problem, one strategy is to transform I into an equivalent but simplified instance I' . Examples of such simplifying reductions are

Consistency: elimination of values or tuples which cannot be extended to a global consistent labelling,

Soft consistency: shifting of weights among different cost functions in order to obtain a better lower bound,

Substitution operations: elimination of domain values which are inessential in the sense that they can always be replaced by another value in any global solution.

Reduction operations may be applied just once as a preprocessing step before embarking on a complete search, or at every node of the search tree. Clearly, such operations should only be applied if, on average, the time gained by the resulting decrease in the size of the search tree outweighs the time required to actually apply the reduction operations.

The constraint satisfaction problem (CSP) is a generic combinatorial problem with many applications in artificial intelligence [137] and operations research. It provides an ideal framework for expressing the line drawing labelling problem when all constraints are crisp. It has been generalized to the valued constraint satisfaction problem (see Definition 8.16, below) which allows us to mix crisp constraints and soft preference constraints.

Definition 8.1 *A constraint satisfaction problem (CSP) is a tuple $\langle N, D, C \rangle$ where $N = \{X_1, \dots, X_n\}$, each variable $X_i \in N$ has a domain d_i of possible*

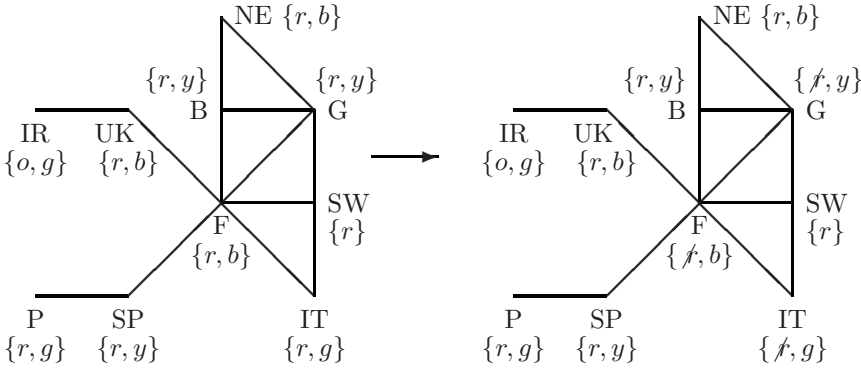


Figure 8.1: Example of local reduction operations applied to a CSP.

values, and C is a set of constraints. Each constraint $\langle I, R_I \rangle \in C$ is defined over a set of variables $M \subseteq N$ (its scope) as a relation R_M of tuples of values that may be simultaneously assigned to the variables in M .

Notation: Given an assignment $t : N \rightarrow d_1 \times \dots \times d_n$ to the variables N and a subset $M = \{X_{i_1}, \dots, X_{i_m}\} \subseteq N$, we write $t[M]$ to represent $\langle t(X_{i_1}), \dots, t(X_{i_m}) \rangle$. To avoid cumbersome notation, we write $t[i]$ or simply t_i , instead of $t(X_{i_m})$, to denote the value assigned to the i th variable. For the same reason, we use R_{ij} as a synonym of $R_{\{i,j\}}$.

An assignment t satisfies a constraint $\langle M, R_M \rangle$, if $t[M] \in R_M$. A solution to a CSP is an assignment to its variables that satisfies all its constraints.

Example 8.2 Consider the problem of colouring a map of western Europe so that each country is assigned a colour among the main colours on its national flag (excluding black and white). This problem can be expressed as the CSP shown on the left-hand side of Figure 8.1, in which the domain of the ten variables are shown as the initial letters of the colours that can be assigned to the corresponding country and a line between adjacent countries represents a *not-equal-to* constraint. The fact that Switzerland must be assigned the colour red implies that we can eliminate red from the domain of its neighbours: Italy, France and Germany. These eliminations propagate. For example, we can now eliminate the colour blue from the domain of the United Kingdom and the colour yellow from the domain of Belgium, which, in turn, implies that red can be eliminated from the domain of the Netherlands. The resulting reduced problem is said to be arc consistent.

If we require only one solution, we can eliminate domain elements which are not essential for finding a solution. For example, in any colouring of the map in which Spain is assigned red, we can change this colour to yellow to obtain another valid colouring since yellow is consistent with all possible colours in the

domains of Spain's neighbours. Eliminating red from the domain of Spain is an example of a neighbourhood substitution operation. Such eliminations can also propagate. \square

8.2 When Local Reductions Suffice

Certain combinatorial problems can be solved by reduction operations alone, in the sense that in the subsequent search no backtracking is required. To illustrate this, we describe the classical tractable problem HORNSAT.

An instance of SAT is given by a conjunction of clauses (disjunctions of literals), where a literal is either a variable or the negation of a variable. In an instance I of SAT, if a clause only contains one literal l_1 , we can clearly assign the value **true** to l_1 . Knowing that $l_1 = \mathbf{true}$, we can now replace each clause of the form $(\neg l_1 \vee l_2 \vee \dots \vee l_k)$ by $(l_2 \vee \dots \vee l_k)$ and simply delete each clause of the form $(l_1 \vee l_2 \vee \dots \vee l_k)$ (since it is clearly satisfied). This operation is called *unit propagation*. These reduction operations can propagate (if $k = 2$). If we produce an empty clause, this is a contradiction, which implies that I has no solution. Otherwise, we end up with a reduced instance I' equivalent to I .

Definition 8.3 *A Horn clause is a clause containing at most one positive literal.*

The following are examples of Horn clauses: (X_1) , $(X_1 \vee \neg X_2)$, $(X_1 \vee \neg X_2 \vee \neg X_3)$, \dots , $(\neg X_1)$, $(\neg X_1 \vee \neg X_2)$, \dots

Definition 8.4 *HORNSAT is the set of instances of SAT in which each clause is a Horn clause.*

HORNSAT can be solved in polynomial time by applying unit propagation until convergence and then, if no contradiction (empty clause) has been produced, by assigning the value **false** to all unaffected variables. With the use of appropriate data structures, HORNSAT can be solved in time which is linear in the number of clauses [58].

Horn clauses have been generalized to non-boolean domains (Section 9.1.2). Certain other restrictions on the types of constraints that can occur in a CSP also give rise to tractable problems. For example, CSPs with 0/1/all constraints (a generalization of 2SAT to non-boolean domains, described in detail in Section 9.1.1) can be solved by establishing consistency on all 3-variable subsets of the variables [46, 91]. In fact, the classes of constraints for which local consistency ensures global consistency have been completely characterized [85, 65]. The now substantial body of work on tractable classes of hard constraints has recently been surveyed by Cohen and Jeavons [25].

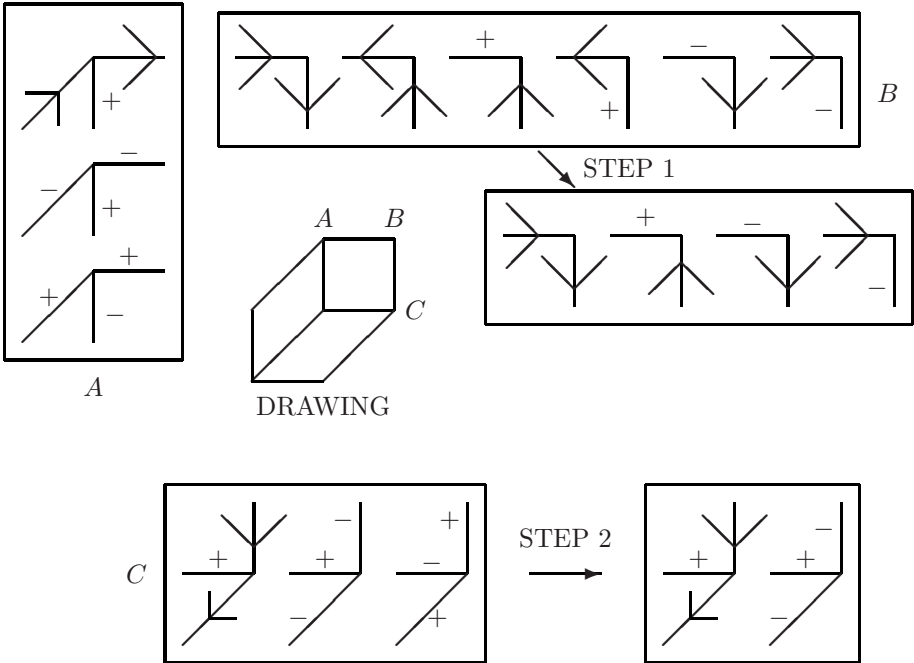


Figure 8.2: Arc consistency operations applied to a simple line drawing labelling problem. The drawing is shown (in the centre) together with the sets of labellings for the three junctions A , B , C .

8.3 Arc Consistency

Figure 8.2 shows an example of the use of arc consistency on a simple line drawing labelling problem. Each of the six junctions of the drawing is a variable to which we have to assign a legal labelling so that adjacent junctions assign the same semantic label ($+$, $-$, \rightarrow , \leftarrow) to the line between them. The drawing is shown in the centre, and the lists of possible labellings of each of the three junctions A , B , C are given within rectangular boxes. We assume that these junctions are projections of trihedral vertices. Two of the legal labellings of B can be immediately eliminated since in the possible labellings of A , line AB cannot be labelled with a left-pointing arrow (STEP 1 in Figure 8.2). Because of these eliminations from the list of labellings of B , we can deduce line BC cannot be labelled $+$ (convex). This, in turn, reduces the set of possible labellings for junction C (STEP 2 in Figure 8.2). We can now deduce that the central line of junction C must be convex, since it is labelled $+$ in the two remaining labellings of C . After applying all possible arc consistency operations, we can deduce that three convex edges meet at the central vertex of the object depicted in the drawing.

Notation: If R is a relation on variables X_1, \dots, X_r , and $S \subseteq \{X_1, \dots, X_r\}$, then $\pi_S R$ represents the projection of R onto the variables S :

$$\pi_S R = \{\langle x_{i_1}, \dots, x_{i_s} \rangle : \exists \langle x_1, \dots, x_r \rangle \in R\}.$$

When S is a singleton, we use the simpler notation $\pi_i R$ instead of $\pi_{\{X_i\}} R$.

Definition 8.5 *A binary CSP is arc consistent if each element of each domain is compatible with at least one element of each of the other domains:*

$$\forall i, j \quad (d_i \subseteq \pi_i(R_{ij} \bowtie d_j)),$$

where d_i is the domain of variable X_i , R_{ij} is the list of compatible assignments to the pair of variables $\langle X_i, X_j \rangle$, and \bowtie is the standard relational join operator.

An arc-consistent CSP does not necessarily have a legal global solution. Consider a graph-colouring problem on a triangle, in which each domain is $\{r, g\}$. This CSP is arc consistent but has no solution.

Different algorithms have been proposed for establishing arc consistency in binary CSPs. In Figure 8.3 we give a version of the algorithm **AC-2001** [10]. For simplicity of presentation, we suppose that each domain $d_i = \{1, 2, \dots, d\}$. An essential ingredient of this algorithm is the data structure $MinSupp(X_j, b, X_i) = \min\{a \in d_i : (a, b) \in R_{ij}\}$, which, for each $b \in d_j$, records the smallest element of domain d_i which is compatible with the assignment $X_j = b$. The total number of iterations of lines (1) and (2) is $O(ed^2)$, where e is the number of binary constraints in the CSP. The time complexity of **AC-2001** is therefore $O(ed^2)$. This is optimal in the sense that we need ed^2 operations to read the constraints of the problem instance.

```

Initialization() ; (* MinSupp( $X_j, b, X_i$ ) = 1,  $Q = \{X_1, \dots, X_n\}$  *)

while  $Q \neq \emptyset$  do
  extract  $X_i$  from  $Q$  ;
  for each  $j$  such that  $\exists$  constraint on  $\{i, j\}$  do
(1)   for each  $b \in d_j$  do
       $a := \text{MinSupp}(X_j, b, X_i)$  ;
      while ( $a \notin d_i$  or  $(a, b) \notin R_{ij}$ ) and  $a \leq d$  do
(2)          $a := a + 1$  ;
      if  $a > d$  (*  $b$  has no support in  $X_i$  *)
      then  $d_j := d_j - \{b\}$  ;  $Q := Q \cup \{X_j\}$  end;
      else  $\text{MinSupp}(X_j, b, X_i) := a$  ;

```

Figure 8.3: The algorithm AC-2001.

When a CSP contains non-binary constraints, we can use generalized arc consistency instead of arc consistency. Recall that each constraint of a CSP is a pair $\langle M, R_M \rangle$, where the *constraint scope* M is a subset of variables and the *constraint relation* R_M is the list of all compatible assignments to the variables in M .

Definition 8.6 A CSP $\langle N, D, C \rangle$ is generalized arc consistent if for each constraint $\langle M, R_M \rangle \in C$ and for each variable $X_i \in M$ each element of the domain d_i can be extended to an assignment belonging to R_M .

Suppose that there is a constraint $f(X_1, \dots, X_k) = \text{vrai}$ on the variables X_1, \dots, X_k . The basic operation in establishing generalized arc consistency is the following consistency test: for $a \in d_1$

$$\exists (a_2, \dots, a_k) \in d_2 \times \dots \times d_k \text{ such that } f(a_2, \dots, a_k) = \text{vrai}?$$

In the most general case, this consistency test is exponential in k . Nevertheless, for certain types of constraints this test can be performed in time which is a polynomial function of k [163]. For example, if f is a SAT clause, such as $X_1 \vee \dots \vee X_i \vee \neg X_{i+1} \vee \dots \vee \neg X_k$, then the consistency test is an $O(k)$ operation. In SAT, generalized arc consistency can be established in time which is linear in the size of the original instance by applying unit propagation until convergence.

Another common example is the AllDiff constraint [133], which is equivalent to $\forall i, j \in \{1, \dots, k\} X_i \neq X_j$. In this case, there is an $O(k^{1.5}d)$ algorithm for the consistency test:

1. Establish a bipartite graph $\langle V_1 \cup V_2, E \rangle$ with a node in V_1 for each variable X_i , a node in V_2 for each domain value v and an edge $\{X_i, v\} \in E$ if $v \in d_i$.
2. Using Hopcroft and Karp's maximal matching algorithm, find a maximal matching M , i.e. a maximal set of non-intersecting edges $\{X_i, v\}$.

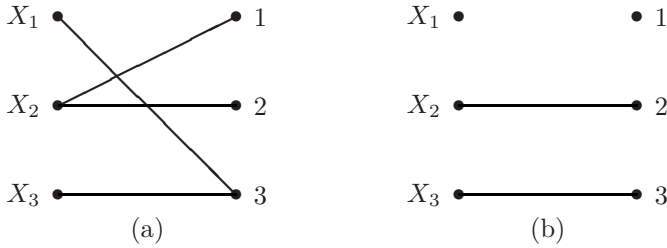


Figure 8.4: Example of a consistency test in an AllDiff constraint using maximal matching.

3. Check that $|M| = k$.

As a simple example, consider the 3-variable problem with domains $X_1 \in \{3\}$, $X_2 \in \{1, 2\}$, $X_3 \in \{3\}$ and the constraint $\text{AllDiff}(X_1, X_2, X_3)$. The corresponding bipartite graph is shown in Figure 8.4(a). Since the size of the maximal matching (shown in Figure 8.4(b)) is only 2, we can deduce that this problem is inconsistent.

8.4 Neighbourhood Substitution

Arc consistency is ubiquitous in constraint processing. However, another, lesser-known technique exists for domain reduction: neighbourhood substitution. Substitution reductions guarantee the conservation of at least one solution of a CSP (if a solution exists) but do not conserve the set of all solutions.

Definition 8.7 A value $a \in d_i$ is substitutable for $b \in d_i$ if in every solution (global consistent labelling) replacing the assignment $X_i = b$ with $X_i = a$ produces another solution. The corresponding substitution operation is the elimination of b from domain d_i .

Detecting a valid substitution according to Definition 8.7 is NP-hard. We therefore turn our attention to a local version, first defined by Freuder [66].

Definition 8.8 A value $a \in d_i$ is neighbourhood substitutable for $b \in d_i$ if for each binary constraint $\langle\langle i, j \rangle, R_{ij}\rangle$ and for each $z \in d_j$

$$(b, z) \in R_{ij} \implies (a, z) \in R_{ij}.$$

We write $b \rightarrow a$.

Example 8.9 Consider the 4-variable graph-colouring problem on the left of Figure 8.5(a). The domain of the each variable is specified by listing the initial letter of the colours that can be assigned to the variable. During the first step

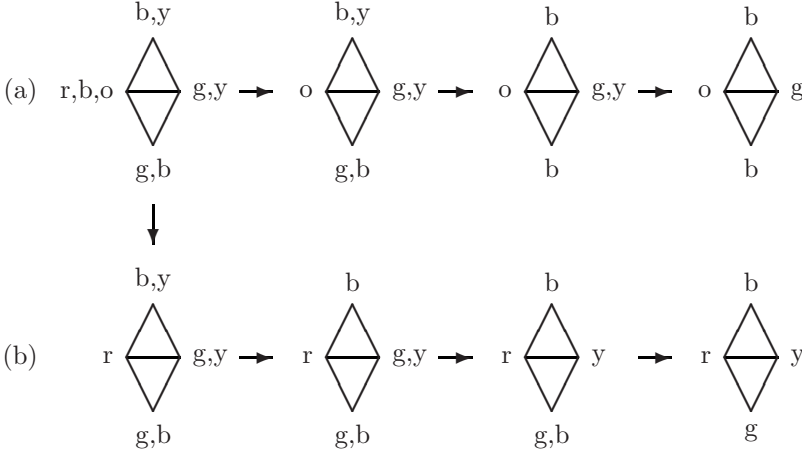


Figure 8.5: Two convergent sequences of neighbourhood-substitution operations on the same graph-colouring problem.

we apply the neighbourhood substitutions $r \rightarrow o$ and $b \rightarrow o$, to eliminate red and blue from the domain of the leftmost node. Only after these eliminations can we apply, in the second step, the neighbourhood substitutions $y \rightarrow b$ and $g \rightarrow b$ at the top and bottom nodes, respectively. These eliminations then trigger the third and final step, in which we apply $y \rightarrow g$ at the rightmost node.

Unlike arc consistency, the result of applying neighbourhood substitutions until convergence is not unique. For example, Figure 8.5(b) shows another valid sequence of eliminations by neighbourhood substitution applied to the same graph-colouring problem. \square

The following theorem, however, tells us that there is no best convergent sequence of neighbourhood-substitution operations; the resulting CSPs are all isomorphic in the sense that they can be made identical by renaming the elements of each domain.

Theorem 8.10 [33] *Two convergent sequences of neighbourhood-substitution operations applied to a CSP produce two isomorphic CSPs.*

An algorithm to apply neighbourhood-substitution operations until convergence exists with time complexity $O(d^3e)$ [33]. It employs the same basic optimization techniques as for arc consistency but has a greater complexity since it has to test all pairs (a, b) of domain elements to see whether $a \rightarrow b$.

If we want to apply both arc consistency and neighbourhood substitution to the same problem, arc consistency should be established first. Indeed, an elimination by neighbourhood substitution cannot destroy arc consistency, whereas an arc consistency elimination can trigger new eliminations by neighbourhood substitution. In fact, this result also holds for higher-order forms of consistency [33].

Let E be the set of neighbourhood substitutions applied to a CSP.
 Let Sol be the set of solutions to the reduced CSP.

```

 $L := Sol ;$ 
while  $L \neq \emptyset$  do
  Extract a solution  $x = (x_1, \dots, x_n)$  from  $L ;$ 
  for  $i = 1$  to  $n$  do
    for each  $x'_i$  such that  $(x'_i \rightarrow x_i) \in E$  do
      if  $x' = (x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)$  satisfies
        all constraints on variable  $X_i$  and  $x' \notin Sol$ 
      then add  $x'$  to  $L$  and to  $Sol$ 
  end while

```

Figure 8.6: The algorithm **Reconstruct** to build the set of all solutions to a CSP after having solved a reduced version resulting from applying a set of neighbourhood-substitution operations.

In combinatorial problems the variables can have two very distinct meanings. In resource allocation and planning problems, for example, variables are user-controlled. On the other hand, in vision or diagnostic problems, variables represent predefined unknowns whose values the user is trying to discover. Neighbourhood substitution can accelerate the search for a single solution (which is clearly useful in problems in the first category) and also the search for the existence of at least one solution (which can be applied to all types of problems). What is less obvious is that, even after having applied neighbourhood-substitution operations, we can still efficiently recuperate all solutions. It suffices to repeatedly try to apply all neighbourhood substitutions in reverse to all global consistent labellings. The resulting algorithm **Reconstruct** is given in Figure 8.6. Its time complexity is $O(N(de + n^2))$, where N is the total number of solutions to the CSP and n the number of variables (provided we use a tree data structure to store Sol) [33].

8.5 Simplification of Soft Constraint Problems

The valued constraint satisfaction problem (VCSP) is a generic optimization problem over finite domains with many applications in areas as diverse as artificial intelligence, operations research and biomathematics [115]. We have already seen in Chapters 3, 4, 5, 7 how the problem of finding the most likely labelling of a line drawing can be expressed as a VCSP.

Soft consistency operations transform a VCSP into an equivalent problem by shifting weights between cost functions of different arities. The principal aim is to produce a lower bound on the cost of any solution to the VCSP. A good lower bound is an essential ingredient of a branch-and-bound search. For example, establishing a directional version of soft arc consistency at every node

of the search tree allowed us to completely exhaust over search spaces of size 10^{350} in optimal planning problems [47]. In sketch-based modelling systems, the user is unlikely to enter a sketch involving thousands of lines. We can therefore envisage the possibility of an exhaustive search for the most likely labelling of the line drawing derived from the sketch.

The notion of arc consistency, which has proved an invaluable tool in solving CSPs, can be extended to the soft constraint framework in several distinct ways. This is because, in general, an optimization problem does not have a unique soft arc consistency closure and hence different compromises are possible between computation time and strength of the resulting closure. Stronger notions of soft arc consistency are obtained:

1. By sending weights in the same direction,
2. By splitting integer weights into fractional parts,
3. By simultaneously performing several operations at once,
4. By decomposing the original problem into the sum of subproblems.

The remainder of this chapter describes several different notions of soft arc consistency based on these ideas.

8.6 Valuation Structures

Crisp yes/no constraints in the CSP are replaced by cost functions in the VCSP [140]. A cost function returns a valuation (a cost, a weight or a penalty) for each combination of values for the variables in the scope of the function. Crisp constraints can still be expressed by, for example, assigning an infinite cost to inconsistent tuples. The following definition of a valuation structure [140] captures the minimal set of properties that a set of valuations must satisfy.

Definition 8.11 *A valuation structure is a tuple $\langle E, \oplus, \geq \rangle$ such that*

- *E is a set, whose elements are called valuations, which is totally ordered by \geq , with a maximum element denoted by \top and a minimum element denoted by \perp ;*
- *E is closed under a binary operation \oplus that satisfies:*
 - $\forall \alpha, \beta \in E, \alpha \oplus \beta = \beta \oplus \alpha;$ *(commutativity)*
 - $\forall \alpha, \beta, \gamma \in E, \alpha \oplus (\beta \oplus \gamma) = (\alpha \oplus \beta) \oplus \gamma;$ *(associativity)*
 - $\forall \alpha, \beta, \gamma \in E, \alpha \geq \beta \Rightarrow (\alpha \oplus \gamma) \geq (\beta \oplus \gamma);$ *(monotonicity)*
 - $\forall \alpha \in E, \alpha \oplus \perp = \alpha;$ *(neutral element)*
 - $\forall \alpha \in E, \alpha \oplus \top = \top.$ *(annihilator)*

Note that in the semi-ring based constraint satisfaction problem (SCSP), valuations satisfy the same properties except that they are only partially ordered [11].

Definition 8.12 *An operator \oplus is strictly monotonic if $\forall \alpha, \beta, \gamma \in E, (\alpha > \beta) \wedge (\gamma \neq \top) \Rightarrow \alpha \oplus \gamma > \beta \oplus \gamma$.*

Examples of strictly monotonic aggregation operators include addition in the non-negative reals with infinity, multiset union in the case of the leximin version of the Fuzzy CSP [63] and the operator $p \oplus q = 1 - (1 - p)(1 - q)$ in the probabilistic CSP [62]. We call a valuation structure strictly monotonic if its operator is strictly monotonic. The simplest possible valuation structure is $\langle \{\perp, \top\}, \oplus, \geq \rangle$ (which is isomorphic to $\langle \{0, \infty\}, +, \geq \rangle$). This valuation structure is also strictly monotonic but only allows us to express crisp constraints.

Define the addition-with-ceiling operator $+_m$ as follows:

$$\forall a, b \in \{0, 1, \dots, m\} \quad a +_m b = \min\{a + b, m\}.$$

Then for $m > 1$, $S_m = \langle \{0, 1, \dots, m\}, +_m, \geq \rangle$ is a valuation structure in which $+_m$ is not strictly monotonic, since $(m - 1) < (m - 1) +_m (m - 1) = m = \top = m +_m (m - 1)$. In a bounded version of MAX-CSP, penalties lie in the valuation structure S_m [96]. It is a version of MAX-CSP in which all solutions which violate m or more constraints are considered equally bad. This is a situation which applies, for example, at a node of a branch-and-bound search tree on a MAX-CSP problem where m is the number of constraints violated by the best solution found so far.

Definition 8.13 [41] *A valuation structure $\langle E, \oplus, \geq \rangle$ is discrete if for each $\alpha \in E$ such that $\alpha < \top$ there is a finite number of $\beta \in E$ such that $\beta \leq \alpha$.*

Clearly, a valuation structure is discrete if and only if it is countable.

Definition 8.14 [50] *In a valuation structure $\langle E, \oplus, \geq \rangle$, if $\alpha, \beta \in E, \alpha \leq \beta$ and there exists a valuation $\gamma \in E$ such that $\alpha \oplus \gamma = \beta$, then γ is known as a difference of β and α .*

The valuation structure is fair if for any pair of valuations $\alpha, \beta \in E$, with $\alpha \leq \beta$, there exists a maximum difference of β and α . This unique maximum difference of β and α is denoted by $\beta \ominus \alpha$.

For example, if \oplus is addition in $\mathbb{R}^+ \cup \infty$, then \ominus is subtraction of real numbers (with $\infty \ominus \infty = \infty$). If \oplus is max, then \ominus is also max [139]. If \oplus is $+_m$, then \ominus is $-_m$ given by $\forall \alpha, \beta \in \{0, 1, \dots, m\}$ such that $m > \alpha \geq \beta, \alpha -_m \beta = a - b$ and $\forall \beta \in \{0, 1, \dots, m\}, m - \beta = m$ [96, 97].

We use the following notation:

$$m\alpha = \underbrace{\alpha \oplus \dots \oplus \alpha}_m.$$

Definition 8.15 *A valuation structure $S = \langle E, \oplus, \geq \rangle$ is rational if, for each $\beta \in E$ and for each integer $n \geq 1$, there is a maximum element $\alpha \in E$ such that $n\alpha = \beta$. We write $\alpha = \frac{1}{n}\beta$.*

Let β be any element of a valuation structure S . Then, by the definition of a valuation structure, $m\beta \in E$ for each non-negative integer m (with 0β being understood to be \perp). Hence a rational valuation structure contains all non-negative rational multiples $\frac{m}{n}\beta$ of its elements.

The valuation structure $\langle N \cup \{\infty\}, +, \geq \rangle$, where N is the set of non-negative integers, can be embedded in the rational valuation structure $\langle \mathbb{Q}^+ \cup \{\infty\}, +, \geq \rangle$, where \mathbb{Q}^+ represents the set of non-negative rational numbers. Similarly, the valuation structure S_k can be embedded in the rational valuation structure $\langle \mathbb{Q}_k \cup \{\infty\}, +_k, \geq \rangle$, where \mathbb{Q}_k is the set of rational numbers α satisfying $0 \leq \alpha < k$.

Embedding a valuation structure in a rational valuation structure has the advantage of enriching the set of available soft arc consistency operations, since we can now project and extend fractional weights.

8.7 Valued Constraint Satisfaction

Definition 8.16 [50] *A valued constraint satisfaction problem (VCSP) is a tuple $\langle N, D, C, S \rangle$, where N is a set of n variables $N = \{X_1, \dots, X_n\}$, each variable $X_i \in N$ has a domain of possible values $d_i \in D$, C is a set of constraints and $S = \langle E, \oplus, \geq \rangle$ is a valuation structure. Each constraint $\langle I, c_I \rangle \in C$ is defined over a set of variables $I \subseteq N$ (its scope) as a function c_I from the cartesian product of the domains $d_i(X_i \in I)$ to E .*

Purely for notational convenience, we suppose that no two constraints have the same scope. This allows us to identify C with the set of scopes I of constraints c_I in the VCSP.

Notation: For $I \subseteq N$ we denote the cartesian product of the domains $d_i(X_i \in I)$ (i.e. the set of possible labellings for the variables in I) by $L(I)$.

Let $I \subseteq J \subseteq N$ with $J = \{X_{j_1}, \dots, X_{j_q}\}$ and $I = \{X_{i_1}, \dots, X_{i_p}\}$. Then, given an assignment $t = (t_{j_1}, \dots, t_{j_q}) \in L(J)$, recall that $t[I]$ denotes the sub-assignment of t to the variables in I , i.e. $(t_{i_1}, \dots, t_{i_p})$.

Definition 8.17 *In a VCSP $V = \langle N, D, C, S \rangle$, the valuation (or cost) of an assignment $t \in L(N)$ is defined by*

$$Val_V(t) = \bigoplus_{I \in C} c_I(t[I]).$$

To solve a VCSP we have to find an assignment $t \in L(N)$ with a minimum valuation.

A CSP can be viewed as a VCSP over the idempotent valuation structure S_1 (which is isomorphic to $\langle \{0, \infty\}, +, \geq \rangle$). MAX-CSP, the problem of maximizing the number of satisfied constraints in an over-constrained CSP, is a VCSP over the valuation structure $\langle \mathbb{N} \cup \{\infty\}, +, \geq \rangle$ in which the constraint functions can only take on the value 0 or 1.

It has been shown [41] that if we restrict ourselves to discrete fair valuation structures, then we need only provide consistency algorithms for VCSPs over the valuation structures S_m and $\langle \mathbb{N} \cup \{\infty\}, +, \geq \rangle$ (which we can write more succinctly as S_∞). These are exactly the valuation structures covered by the WCSP framework [97, 54]. Thus, in the remainder of the chapter we assume that the valuation structure of the VCSP to be solved is S_m for some $m \in \mathbb{N} \cup \{\infty\}$.

Definition 8.18 *Two VCSPs $V_1 = \langle N, D, C_1, S \rangle$, $V_2 = \langle N, D, C_2, S \rangle$ are equivalent if $\forall t \in L(N)$, $Val_{V_1}(t) = Val_{V_2}(t)$.*

Definition 8.19 *The subproblem of a VCSP $\langle N, D, C, S \rangle$ on $J \subseteq N$ is the problem $VCSP(J) = \langle J, D_J, C_J, S \rangle$, where $D_J = \{d_i : X_i \in J\}$ and $C_J = \{I \in C : I \subseteq J\}$.*

Definition 8.20 *For a VCSP $\langle N, D, C, S \rangle$, an equivalence-preserving transformation (EPT) on $J \subseteq N$ is an operation which transforms the subproblem $VCSP(J)$ into an equivalent VCSP.*

Figure 8.7 gives three basic equivalence-preserving transformations. When $t \in L(I)$ and $X_i \in I$, $t[i]$ represents the value assigned to variable i by the labelling t . If $X_i \in I$, we say that $t \in L(I)$ is an *extension* of the assignment (X_i, a) if $t[i] = a$. We write c_i as a shorthand for $c_{\{i\}}$. **Project** projects weights from a valued constraint (on two or more variables) to a unary valued constraint: an increase in $c_i(a)$ is compensated by a corresponding decrease in $c_I(t)$ for each t such that $t[i] = a$. **Extend** performs the inverse operation, sending weights from a unary valued constraint to a higher-order valued constraint. Finally, **UnaryProject** projects weights from a unary valued constraint to the nullary valued constraint c_\emptyset . This nullary constraint is independent of the values assigned to variables and hence provides a lower bound on the cost of any solution, which is an essential ingredient of branch-and-bound search.

In the valuation structure S_m , $m \ominus m = m$. This means that when $\alpha = m$, our three basic equivalence-preserving transformations simply propagate inconsistencies (represented by the maximal cost m). Furthermore, **Extend** also sets $c_I(t)$ to m if the sum of $c_I(t)$, the nullary cost c_\emptyset and the unary costs $c_i(t[i])$ is equal to m .

8.8 Soft Arc Consistency Techniques

In this section we give the definitions of several different notions of soft arc consistency (SAC). Note that certain of these notions have only been defined in special cases, such as binary VCSPs. In the case of binary constraints, we write c_{ij} as a shorthand for $c_{\{i,j\}}$. The following two definitions are slightly modified versions of the definitions given in [50], to allow for a constraint c_\emptyset with empty scope and to bring them in line with the general definition of consistency given in [41].

Procedure **Project**(I, i, a, α)
 (* Precondition: $\alpha \leq \min\{c_I(t) : t \in L(I) \text{ an extension of } (X_i, a)\}$ *)
 $c_i(a) := c_i(a) \oplus \alpha$;
 for each $t \in L(I)$ an extension of (X_i, a) do
 $c_I(t) := c_I(t) \ominus \alpha$;

Procedure **Extend**(i, a, I, α)
 (* Precondition: $\alpha \leq c_i(a)$ *)
 for each $t \in L(I)$ an extension of (X_i, a) do
 $\beta := c_\emptyset \oplus (\bigoplus_{X_j \in I} c_j(t[j]))$;
 $c_I(t) := (c_I(t) \oplus \beta) \ominus \beta$;
 $c_I(t) := c_I(t) \oplus \alpha$;
 $c_i(a) := c_i(a) \ominus \alpha$;

Procedure **UnaryProject**(i, α)
 (* Precondition: $\alpha \leq \min\{c_i(a) : a \in d_i\}$ *)
 $c_\emptyset := c_\emptyset \oplus \alpha$;
 for each $a \in d_i$ do
 $c_i(a) := c_i(a) \ominus \alpha$;

Figure 8.7: The basic equivalence-preserving transformations required to establish different forms of soft arc consistency.

Definition 8.21 A fair VCSP $\langle N, D, C, S \rangle$ is generalized arc consistent if for all $I \in C$ such that $|I| > 1$ we have:

1. $\forall t \in L(I)$, $c_I(t) = (c_I(t) \oplus \beta) \ominus \beta$, where $\beta = c_\emptyset \oplus (\bigoplus_{X_j \in I} c_j(t[j]))$;
2. $\forall X_i \in I$, $\forall a \in d_i$, $c_i(a) = \min\{c_i(a) \oplus c_I(t) : t \in L(I) \text{ is an extension of } (X_i, a)\}$.

If the VCSP is binary, then generalized arc consistency is known as (soft) arc consistency.

Definition 8.22 A binary VCSP is directional arc consistent according to an order $<$ on the variables if for all c_{ij} such that $i < j$, $\forall a \in d_i$

$$c_i(a) = \min_{b \in d_j} ((c_i(a) \oplus c_{ij}(a, b) \oplus c_j(b) \oplus c_\emptyset) \ominus c_\emptyset).$$

Consider a fair binary VCSP with e binary constraints and maximum domain size d . Then directional arc consistency can be established in $O(ed^2)$ time [50].

Definition 8.23 [39] A binary VCSP is full directional arc consistent (FDAC) if it is both arc consistent and directional arc consistent.

Full directional arc consistency can be established in $O(ed^2)$ time if the valuation structure is strictly monotonic (e.g. S_∞) [39] and in $O(end^3)$ time if the valuation structure is S_m (for finite m) [97].

The following definitions are slight generalizations of the definitions given, respectively, in [97] and [54].

Definition 8.24 A VCSP is node consistent if $\forall i$,

$$c_\emptyset = \min\{c_\emptyset \oplus c_i(a) : a \in d_i\}.$$

Definition 8.25 A binary VCSP is existential arc consistent (EAC) if it is node consistent and if $\forall i$, $\exists a \in d_i$ such that for all constraints c_{ij} ,

$$c_i(a) = \min_{b \in d_j} ((c_i(a) \oplus c_{ij}(a, b) \oplus c_j(b) \oplus c_\emptyset) \ominus c_\emptyset).$$

Definition 8.26 [54] A binary VCSP is existential directional arc consistent if it is both existential arc consistent and full directional arc consistent

Over the valuation structure S_m existential directional arc consistency can be established in $O(ed^2 \max\{nd, m\})$ time [54].

Definition 8.27 [40] A binary instance of MAX-CSP P is 3-cyclic consistent (3CC) if for all 3-variable subproblems J of P there is no EPT of J which increases c_\emptyset without introducing new constraints.

3-cyclic consistency can be established in $O(d^3n^4)$ time on an instance of MAX-CSP [40].

An essential difference between consistency in CSPs and consistency in VCSPs is that the closure under the corresponding consistency operations is unique in CSPs, but this is not, in general, the case for VCSPs [139, 50]. For example, even for a 2-variable VCSP with domains of size 2, the arc consistency and existential arc consistency closures are not necessarily unique. Similarly, for problems with more than two variables, in general, neither the FDAC closure nor the 3-cyclic consistency closure is unique.

Figure 8.8(a)–(c) illustrates the three techniques FDAC, EAC and 3CC. In each case the VCSP on the left can be transformed into the equivalent VCSP on the right by establishing the corresponding consistency property. In Figure 8.8(a) the VCSP on the right is obtained by establishing full directional arc consistency, in Figure 8.8(b) by establishing existential arc consistency and in Figure 8.8(c) by establishing 3-cyclic consistency. In each case the lower bound c_\emptyset is increased from 0 to 1. A line joining (X_i, a) and (X_j, b) represents a weight $c_{i,j}(a, b) = 1$ and a value α written next to $a \in d_i$ represents $c_i(a) = \alpha$.

The VCSP on the left-hand side of Figure 8.8(a) is EAC and the problem on the left-hand side of Figure 8.8(b) is FDAC, which proves that these two properties are complementary. Over domains of size 2, existential arc consistency is subsumed by 3-cyclic consistency, but this is no longer the case for larger domains. In 3-variable problems, 3-cyclic consistency subsumes existential and full directional arc consistency but subsumes neither property when there are more than three variables. Therefore all three techniques are potentially complementary.

The classic use of consistency techniques in CSPs is to maintain the corresponding notion of consistency at every node of a search tree. In the case of VCSPs, maintaining a form of soft consistency at every node of a branch-and-bound search tree provides a lower bound c_\emptyset on the cost of extending the present partial assignment to a complete solution. Comparing this lower bound with the cost of the best solution found so far allows us to prune the search tree. From a practical point of view, both maintaining FDAC during search and maintaining EDAC during search have proved their utility in various applications, such as frequency assignment, warehouse allocation [54] and optimal planning [47]. They are part of public domain software devoted to solving soft constraint problems [53].

8.9 Optimal Soft Arc Consistency

An arc consistency closure of a VCSP P is any VCSP obtained from P by propagating inconsistencies and by repeated calls to **Project** and **UnaryProject** until convergence.

Definition 8.28 *An arc consistency closure of a VCSP P is optimal if it has the maximum lower bound c_\emptyset among all arc consistency closures of P .*

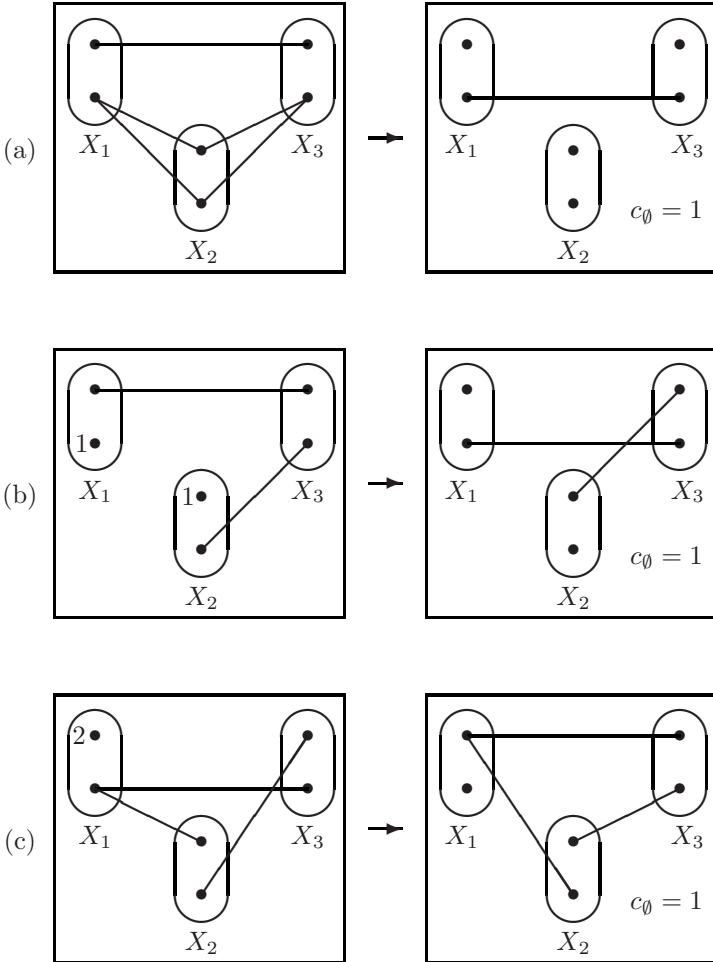


Figure 8.8: Examples of (a) full directional arc consistency, (b) existential arc consistency, (c) 3-cyclic consistency.

In [50] we proved that over a discrete valuation structure such as the non-negative integers together with infinity, the problem of finding the optimal arc consistency closure is NP-hard. However, we will show in this section that extending the valuation structure to include all rationals and extending our notion of arc consistency closure allows us to determine an optimal arc consistency closure in polynomial time by a simple reduction to linear programming. From a theoretical point of view, this result demonstrates that extending the valuation structure not only allows us to produce better lower bounds but also avoids intractability. As Werner [174] has pointed out, an equivalent result was first published by Schlesinger (in Russian [142, 143]) for the finite-cost case. Experimental trials on both random problems and benchmark instances of the frequency assignment problem indicate that, for particularly difficult-to-solve instances, finding an optimal SAC transformation may even be of direct practical use, but only as a preprocessing technique [48].

Definition 8.29 *Given a VCSP P , a SAC transformation is a set of soft arc consistency operations (Extend, Project, UnaryProject) which when applied simultaneously transforms P into a valid VCSP.*

Note that a VCSP is valid if all cost functions take values in the valuation structure. For example, a negative cost is not valid. Affane and Bennaceur [3] split integer costs by propagating a fraction w_{ij} of the binary constraint c_{ij} towards variable X_i and a fraction $1 - w_{ij}$ towards variable X_j (where $0 \leq w_{ij} \leq 1$) and suggested determining the optimal values of the weights w_{ij} . In a more recent paper, Bennaceur and Osmani [8] suggested introducing different weights $w_{i_a j_b}$ for each pair of domain values $(a, b) \in d_i \times d_j$. It turns out that assigning a different weight to each triple (i, j, a) , where $a \in d_i$, allows us to find optimal weights in polynomial time.

Theorem 8.30 *If the valuation structure of a VCSP P is $\mathbb{Q}^+ \cup \{\infty\}$ (where \mathbb{Q}^+ is the set of non-negative rationals), then it is possible to find in polynomial time a SAC transformation of P which maximizes the lower bound c_\emptyset , provided the arity of constraints in P is bounded by a constant.*

Proof: Firstly, as in [39], we can assume that all infinite costs have been propagated using a standard generalized arc consistency algorithm [116]. Note that we assume that $c_I(a)$ has been set to ∞ if $c_i(a[i]) = \infty$ for some $X_i \in I$. At this point no more infinite costs can be propagated in the VCSP by the operations **Extend**, **Project** or **UnaryProject**.

We then want to determine the set of finite SAC operations which when applied simultaneously maximizes the increase in c_\emptyset . For each $I \in C$ such that $|I| > 1$ and for each $X_i \in I$, let $e_i^I(a)$ be the sum of the weights extended from $c_i(a)$ to c_I minus the sum of the weights projected from c_I to $c_i(a)$. Let u_i be the sum of the weights projected (by **UnaryProject**) from c_i to c_\emptyset . Thus the problem is to maximize $\sum_i u_i$ such that the resulting constraint functions take

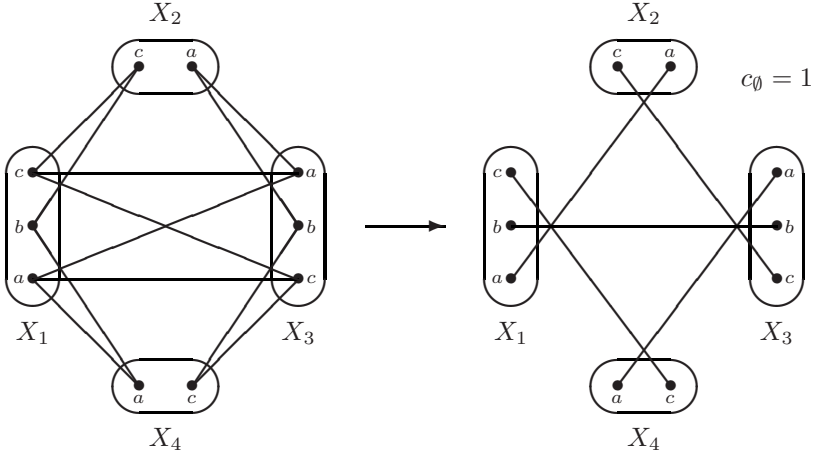


Figure 8.9: An example of a SAC transformation.

on non-negative values, i.e.

$$\forall i \in \{1, \dots, n\} \forall a \in d_i \quad c_i(a) - \sum_{(I \in \mathcal{C}) \wedge (X_i \in I)} e_i^I(a) - u_i \geq 0,$$

$$\forall I \in \mathcal{C} \text{ such that } |I| > 1, \forall a \in L(I) \quad c_I(a) + \sum_{X_i \in I} e_i^I(a[i]) \geq 0.$$

We can simply ignore the inequalities for which $c_i(a) = \infty$ or $c_I(a) = \infty$ since they are necessarily satisfied. The remaining inequalities define a standard linear programming problem with $O(ed+n)$ variables (if e is the number of constraints, n the number of variables and all domains are of size d) which can be solved in polynomial time [19]. ■

A weaker version of the above theorem, limited to 3-variable subproblems, is the basis of the algorithm to establish 3-cyclic consistency [40].

It is important to note that there is a difference between SAC transformations and sequences of SAC operations: the former are stronger due to the fact that several SAC operations applied simultaneously can produce a valid VCSP even when no individual SAC operation can be applied. As an example, consider the binary VCSP P over domains $d_1 = d_3 = \{a, b, c\}$, $d_2 = d_4 = \{a, c\}$, illustrated on the left-hand side of Figure 8.9. It can easily be verified that P is soft arc consistent and that there are no unary weights. It follows that no sequence of SAC operations (**Extend**, **Project** or **UnaryProject**) can be applied to P . Nonetheless, applying the SAC transformation given by $e_2^{23}(c) = e_3^{34}(a) = e_3^{31}(b) = e_1^{12}(a) = e_1^{14}(c) = u_4 = 1$ and $e_3^{23}(a) = e_3^{23}(b) = e_4^{34}(c) = e_1^{31}(a) = e_1^{31}(c) = e_2^{12}(c) = e_4^{14}(a) = -1$ produces an equivalent binary VCSP Q , shown on the right-hand side of Figure 8.9. Since in Q we have $c_\emptyset = 1$, in

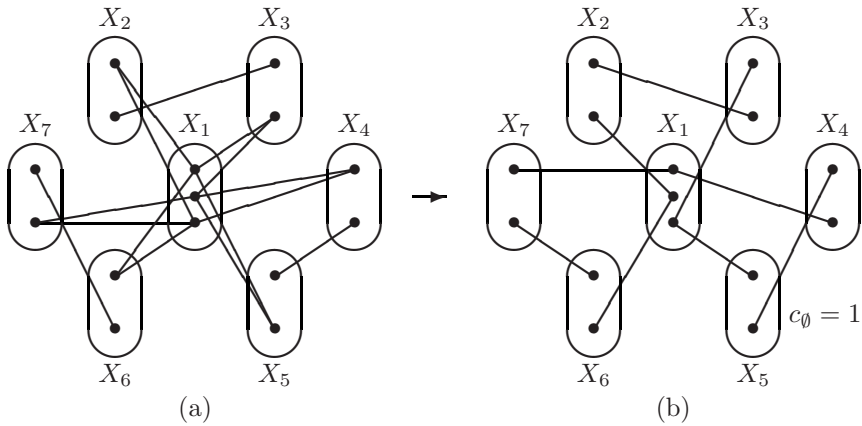


Figure 8.10: No sequence of SAC operations can be applied to the VCSP in (a), but a set of simultaneous SAC operations transforms it into the VCSP in (b).

this example a SAC transformation increases the lower bound c_0 even though no sequence of SAC operations can be applied to P .

Figure 8.10 gives another example. In this instance, there are three isomorphic subproblems on $\{X_1, X_2, X_3\}$, $\{X_1, X_4, X_5\}$ and $\{X_1, X_6, X_7\}$. For each $u \in d_1$, a SAC transformation on one of these 3-variable subproblems leads to an increase in the unary cost $c_1(u)$.

We have seen that applying a set of SAC operations simultaneously (i.e. a SAC transformation) leads to a stronger notion of consistency than applying a set of SAC operations sequentially. An obvious question is whether another even stronger form of consistency exists which transforms a VCSP into an equivalent VCSP.

Definition 8.31 A VCSP P is in-scope c_0 -irreducible if there is no equivalent VCSP Q with the same set of constraint scopes as P and such that $c_0^Q > c_0^P$ (where c_0^P, c_0^Q are the nullary constraints in P, Q).

Let VCSP(sm) denote a VCSP with a strictly monotonic aggregation operator \oplus .

Definition 8.32 [40] A VCSP(sm) is finitely bounded if for all valued constraints $\langle S, c_S \rangle, \forall x \in L(S), c_S < \top$.

The following theorem is a direct consequence of Lemma 5.2 in [40].

Theorem 8.33 Let P be a finitely bounded binary VCSP(sm). If no SAC transformation applied to P produces a VCSP Q with $c_0^Q > c_0^P$, then P is in-scope c_0 -irreducible.

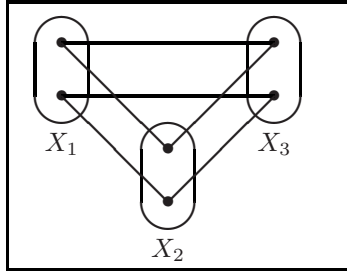


Figure 8.11: A VCSP corresponding to the 2-colour graph-colouring optimization problem on a triangle.

Thus, when all costs are finite rational numbers, the linear programming approach can be used to establish in-scope c_\emptyset -irreducibility in binary VCSPs. This is unfortunately not the case if infinite costs can occur. Consider, for example, the graph-colouring problem on a triangle with two colours, expressed as a VCSP with costs in $\{0, \infty\}$. The problem is clearly inconsistent and hence equivalent to a VCSP with a single constraint $c_\emptyset = \infty$, but no SAC transformation can be applied to this VCSP to increase c_\emptyset .

It should also be mentioned that forms of higher-order consistency have been proposed for VCSPs [41] which can find a better lower bound than any SAC transformation. This is at the cost of introducing higher-order constraints. Consider the optimization version of the graph-colouring problem on a triangle with two colours, equivalent to the VCSP in Figure 8.11, where a line represents a cost of 1. The aim is to assign a colour to each node so as to minimize the number of pairs of nodes joined by an edge and assigned the same colour. No SAC transformation applied to this VCSP increases c_\emptyset , whereas soft 3-consistency produces a lower bound of $c_\emptyset = 1$ [41]. One disadvantage of establishing soft 3-consistency is that some weights are now stored in ternary constraints.

Note that the special case of real-valued binary VCSPs over boolean domains has been extensively studied under the name of quadratic pseudo-boolean function optimization [13]. In the case of boolean domains, it is well known that finding an equivalent quadratic posiform representation (i.e. an equivalent binary VCSP) with an optimal value of c_\emptyset can be formulated as a linear programming problem [72] and can even be solved by finding a maximum flow in an appropriately defined network [13]. It is also worth noting that in this special case of boolean binary VCSPs, determining whether there exists a zero-cost solution is an instance of 2SAT and hence can be completely solved in polynomial time.

8.10 Virtual Arc Consistency

Definition 8.34 *If $P = \langle N, D, C, S \rangle$ is a VCSP, then $Bool(P)$ is the CSP $\langle N, D, \overline{C} \rangle$, where $\langle S, R_S \rangle \in \overline{C}$ iff $\exists \langle S, c_S \rangle \in C$ with $S \neq \emptyset$ such that $\forall x \in$*

$$L(S) (x \in R_S \Leftrightarrow c_S(x) \oplus c_\emptyset = c_\emptyset).$$

$\text{Bool}(P)$ is a CSP whose solutions are exactly those n -tuples x such that $\text{Val}_P(x) = c_\emptyset$. For ease of comparison with the corresponding VCSP P , in figures we will always represent $\text{Bool}(P)$ as if it were a VCSP over the boolean valuation structure $\langle \{0, 1\}, +, \geq \rangle$ with $0 < 1$ and $1 + 1 = 1$. In other words, a line between (X_i, a) and (X_j, b) represents the fact that (a, b) is not a consistent assignment to variables (X_i, X_j) and a weight of 1 next to (X_i, a) represents the fact that a is not a consistent assignment to variable X_i .

Definition 8.35 A VCSP P is virtual (generalized) arc consistent if establishing (generalized) arc consistency in the CSP $\text{Bool}(P)$ does not lead to an inconsistency.

Note that, even if P is virtual arc consistent, then the arc consistency closure of $\text{Bool}(P)$ may still be of help in indicating which elements of domains necessarily imply a non-zero cost. This information could be employed in variable or value ordering heuristics which help to find the optimal solution (or at least a good solution) sooner. Quickly finding a good (but not necessarily optimal) solution is an essential ingredient of branch and bound, since it provides a good upper bound on the value of the optimal solution. During branch and bound on a VCSP with integer weights, the valuation structure is effectively S_m , where m is the best upper bound found so far. A smaller value of m leads to more pruning of the search tree.

If P is not virtual arc consistent, then we know by Theorem 8.30 that we can theoretically find in polynomial time a SAC transformation that produces an optimal value of c_\emptyset (provided that the valuation structure can be embedded in $\mathbb{Q}^+ \cup \{\infty\}$). In this section we present a low-order polynomial-time algorithm which determines a sequence of SAC operations which necessarily increases c_\emptyset if such a sequence exists. If no such sequence exists, then the VCSP is virtual arc consistent.

When establishing SAC [50] we often have a choice as to the direction in which we project or extend weights. Note that the name virtual arc consistency comes from the fact that instead of making such choices, we effectively project or extend simultaneously virtual weights in all possible directions, by establishing arc consistency in $\text{Bool}(P)$.

Consider the following instance P of MAX-SAT: $\neg X_1; X_1 \vee \neg X_4; \neg X_3 \vee X_4; X_2; \neg X_2 \vee X_3$. This VCSP is shown in the leftmost box of Figure 8.12. A line joining (X_i, a) and (X_j, b) represents a cost $c_{ij}(a, b) = 1$. Unary costs $c_i(a) = 1$ are noted next to the domain element (i, a) . Note that P is existential directional arc consistent and 3-cyclic consistent. However, establishing arc consistency in $\text{Bool}(P)$, as shown in Figure 8.12, leads to an inconsistency. The leftmost box in Figure 8.12 also represents $\text{Bool}(P)$ where now weights are interpreted as being boolean values (i.e. either 0 or 1 with $1+1=1$).

During establishment of arc consistency in $\text{Bool}(P)$, the reason for each inconsistency (i.e. a weight which changes from 0 to 1) is recorded. Suppose

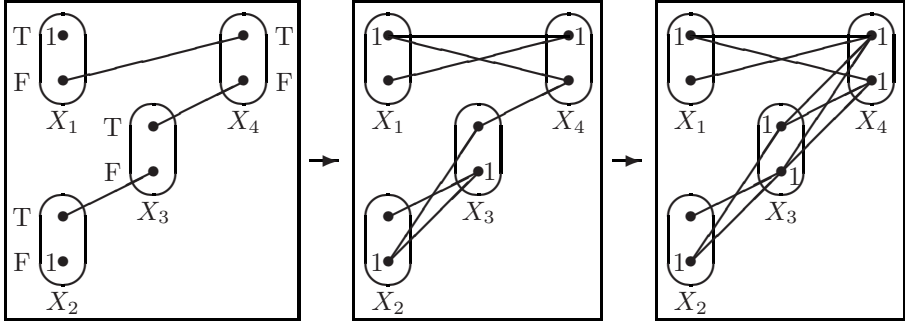


Figure 8.12: A VCSP P (leftmost box) which is full directional arc consistent, existential arc consistent and 3-cyclic consistent but not virtual arc consistent, as shown by establishing arc consistency in $\text{Bool}(P)$.

that inconsistency in $\text{Bool}(P)$ is first detected at X_4 . This means that by SAC operations in P we can transform P into an equivalent VCSP in which $\forall x \in d_4, c_4(x) \geq \lambda$. We can associate λ to each $x \in d_4$ and trace back these weights by, at each step, using the reason for inconsistency as recorded during the establishment of arc consistency in $\text{Bool}(P)$. This is illustrated in Figure 8.13(a). The algorithm halts when all weights have been traced back to a non-zero weight in the original VCSP P . All the weights of λ shown in the final box of Figure 8.13(a) correspond to non-zero weights in the original problem P . The value of λ must not exceed any of these original weights. In this case the maximal value we can assign to λ is clearly 1. The SAC operations illustrated in Figure 8.13(b) can now be applied in reverse (i.e. in reverse order and with each **Extend** replaced by **Project** and vice versa). This is illustrated in Figure 8.13(b). In the resulting VCSP we have $c_\emptyset = 1$. This VCSP is easily seen to be virtual arc consistent.

Unfortunately, establishing virtual arc consistency (VAC) may require the introduction of fractional weights, as the following example illustrates. Consider the instance P of MAX-SAT given by $\neg X_1; X_1 \vee \neg X_2; X_1 \vee X_3; X_2 \vee \neg X_3$. This problem is illustrated in the leftmost box of Figure 8.14(a). As usual, each line represents a cost of 1 and unary costs are noted next to the corresponding domain element. $\text{Bool}(P)$ is also represented by the same figure, where now the value 1 is understood to be the element of the boolean algebra $\{0, 1\}$ with $1 + 1 = 1$. Figure 8.14(a) illustrates the process of establishing arc consistency in $\text{Bool}(P)$: arc consistency is established on the pairs of variables (X_1, X_2) , (X_1, X_3) and then on the pair (X_2, X_3) , which leads to a domain wipe-out at X_3 . We can therefore already deduce a lower bound of the integer value 1 for the original problem P . However, in this example, no set of SAC operations with integer weights produces a non-zero lower bound.

In order to determine a sequence of SAC operations which lead to an increase in c_\emptyset , we have to retrace the steps made while establishing arc consistency in $\text{Bool}(P)$. Suppose that the final increase in c_\emptyset in P is λ . We place a value of λ at each element of d_3 , as illustrated by the leftmost box in Figure 8.14(b).

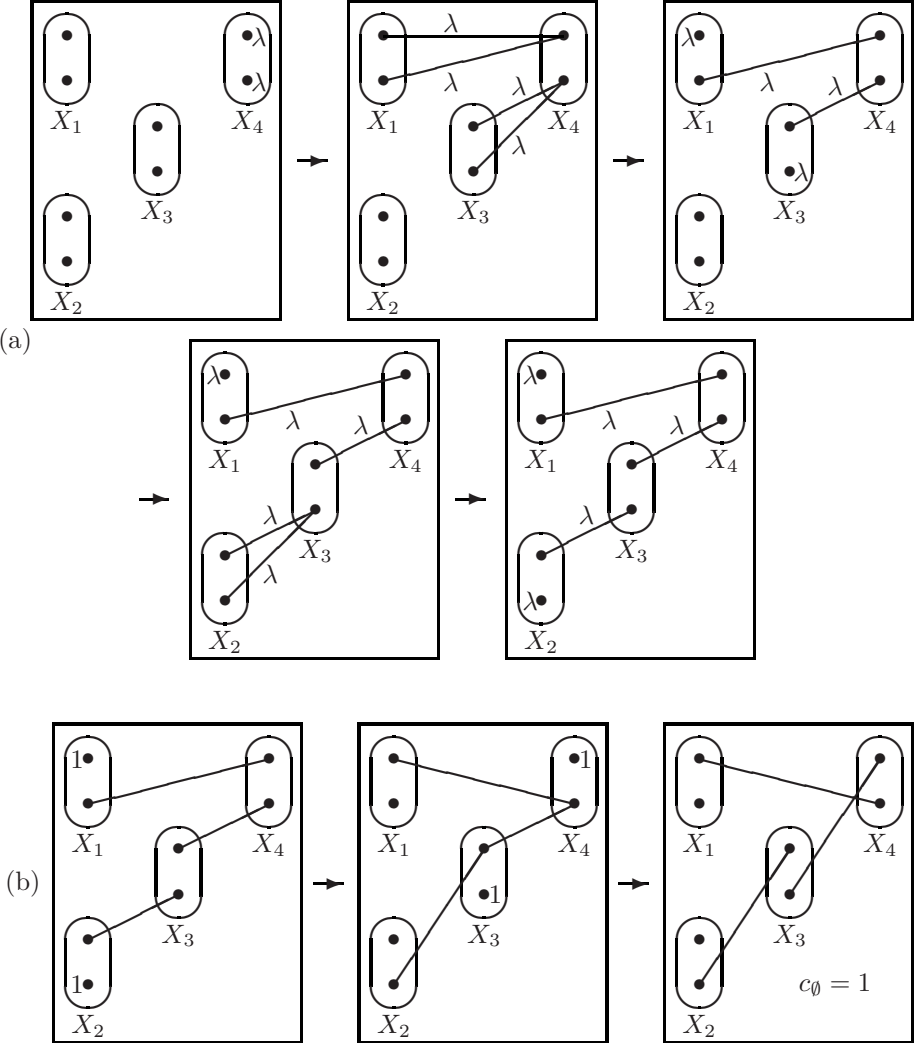


Figure 8.13: (a) Tracing back weights of λ from X_4 until we arrive at non-zero weights in the original VCSP P of Figure 8.12; (b) applying the corresponding SAC operations in reverse to P .

Retracing our steps, we know that these weights can be obtained by projection from the binary constraints c_{13} and c_{23} (as illustrated in the next box in Figure 8.14(b)). If the corresponding weight in the original problem P was non-zero, such as $c_{13}(F,F)$ and $c_{23}(F,T)$, then such weights do not need to be traced back any further. We know that the other weights can be obtained by extension from c_1 and c_2 . A weight of λ has to be traced back further via c_{12} to c_1 . The algorithm halts when all remaining weights were non-zero in the original VCSP P (as shown in the last box in Figure 8.14(b)). We have traced a combined weight of 2λ back to $c_1(T)$. Since $c_1(T)=1$ in P , the maximum value we can assign to λ is $\frac{1}{2}$. Applying these soft arc consistency operations in reverse to the original VCSP P with $\lambda = \frac{1}{2}$, as shown in Figure 8.14(c), produces an equivalent VCSP with $c_\emptyset = \frac{1}{2}$.

The following theorem shows that if establishing arc consistency in $\text{Bool}(P)$ produces an inconsistency, then it is possible to increase c_\emptyset by a sequence of SAC operations.

Theorem 8.36 *Let P be a VCSP over a rational valuation structure, with $c_\emptyset = \perp$. Then there exists a sequence of SAC operations which when applied to P leads to an increase in c_\emptyset if and only if the arc consistency closure of $\text{Bool}(P)$ is inconsistent.*

Proof: \Rightarrow : Let O_1, \dots, O_t be a sequence of SAC operations in P which produce an equivalent VCSP with $c_\emptyset > \perp$. Let O'_1, \dots, O'_t be the corresponding arc consistency operations over a boolean valuation structure with the weight being projected or extended always equal to the boolean value 1. Applying this sequence of operations to $\text{Bool}(P)$ (viewed as a VCSP over a boolean valuation structure) inevitably leads to a domain wipe-out and hence inconsistency.

\Leftarrow : Let O_1, \dots, O_t be a sequence of arc consistency operations in $\text{Bool}(P)$ which leads to an inconsistency (a domain wipe-out). We can assume without loss of generality that no two of the operations O_i, O_j are strictly identical, since the same arc consistency operation never needs to be applied twice in a CSP. Each O_i corresponds to an **Extend**, **Project** or **UnaryProject** operation when $\text{Bool}(P)$ is viewed as a VCSP over a boolean valuation structure. Let O'_i be the corresponding SAC operation in P applied with a weight δ/e^i , where δ is the minimum non- \perp weight occurring in P . We divide by e each time, since a weight may need to be divided into smaller quantities to be extended to all constraints involving the same variable (or projected to all variables in the same constraint scope). After applying O'_1, \dots, O'_t to P we necessarily have $c_\emptyset \geq \delta/e^t > \perp$. ■

It may not seem that increasing c_\emptyset by such a small amount as δ/e^t is worthwhile. However, if the original weights in P were all integers, then $c_\emptyset \geq \delta/e^t > 0$ actually implies that $\text{Val}_P(x) \geq 1$, for all x , thus allowing us to increase the lower bound used by branch and bound by 1.

Unfortunately, if the original weights in P were rationals rather than integers, then finding an inconsistency in $\text{Bool}(P)$ does not necessarily imply that we can

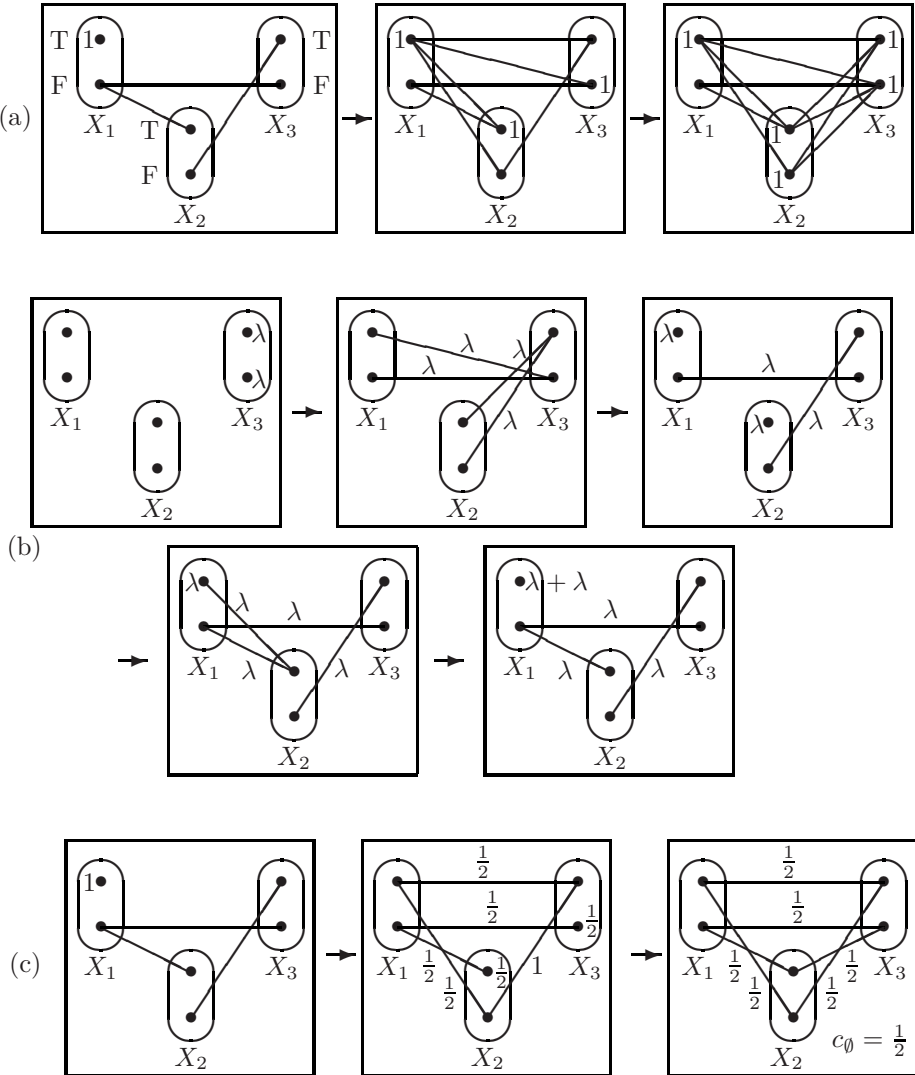


Figure 8.14: An example of a VCSP where virtual arc consistency produces a better lower bound than FDAC, EAC and 3-cyclic consistency by allowing fractional weights.

increase the lower bound on $Val_P(x)$ by 1. We have seen in the example of Figure 8.14 that applying a sequence of SAC operations found by our virtual arc consistency algorithm may lead to the introduction of fractional weights in the VCSP. We have to ensure that we avoid an infinite loop in which we make smaller and smaller increases to c_\emptyset each time. VAC can be tested in $O(ed^2)$ time using an optimal arc consistency algorithm, such as **AC-2001** (Figure 8.3) in the CSP $Bool(P)$. It can also, in theory at least, be established in polynomial time via linear programming if the valuation structure can be embedded in $\mathbb{Q}^+ \cup \{\infty\}$ by Theorem 8.30. An alternative and much less costly solution to the problem of fractional weights is presented in Section 8.11.

Existential arc consistency [54] can be seen as applying virtual arc consistency but limited to a single iteration of arc consistency in $Bool(P)$. In EAC, weights are transferred virtually from all variables to their neighbours; if a unary projection with a non-zero weight is possible, then we trace back and actually perform the necessary SAC operations. Thus EAC avoids the problem of fractional weights by applying only a weak form of VAC.

8.11 VAC Decomposition

We have seen that there is no guarantee that our VAC algorithm will terminate in a finite number of iterations, due to the possible introduction of smaller and smaller fractional weights. However, we will show in this section that this problem can be avoided while at the same time turning fractional values of c_\emptyset to our advantage. Throughout this section we assume that all weights in the original VCSP are non-fractional (i.e. integers or infinite).

Imagine a VCSP P consisting of $2k$ copies of the VCSP P_3 illustrated in Figure 8.14. The following analysis applies whether P is a $6k$ -variable VCSP consisting of $2k$ non-intersecting copies of P_3 or whether P is the same 3-variable VCSP as P_3 with all weights multiplied by $2k$. Establishing VAC only produces $c_\emptyset = k$, and, in fact, no SAC transformation produces a better lower bound. However, a lower bound of $2k$ is easily obtained by observing that P is the sum of $2k$ integer-valued VCSPs each with a non-zero lower bound. We therefore abandon the idea of finding a single equivalent VCSP. Instead we express the VCSP as the sum of integer-valued VCSPs in order to obtain a better lower bound while, at the same time, guaranteeing termination in polynomial time.

In the previous section we presented an iteration of our VAC algorithm as a three-stage procedure: (1) search for an inconsistency in $Bool(P)$, (2) trace back the value of λ from c_\emptyset until we reach non-zero weights in P , and (3) calculate the value of λ and propagate forward in order to increase c_\emptyset by λ . We adapt this algorithm as follows.

Let Q be a copy of our original VCSP P . After tracing back weights in the VCSP Q , instead of propagating weights in Q , we create another VCSP R (with

cost functions c^R) defined as follows: for all constraint scopes I , $\forall a \in L(I)$

$$c_I^R(a) = \begin{cases} \top & \text{if } c_I^Q(a) = \top \wedge k(I, a) > 0, \\ 1 & \text{if } c_I^Q(a) < \top \wedge k(I, a) > 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $k(I, a)$ is the coefficient of λ associated with (I, a) after tracing back a cost of λ in the VCSP Q . We then propagate in R . The VCSP R contains exactly the set of weights which provoked the arc inconsistency in $\text{Bool}(Q)$, and hence propagating in R necessarily produces a VCSP R' with $c_\emptyset^{R'} = \lambda > 0$. Since the weights in R were non-fractional, we can deduce a lower bound of 1 for Val_R . We can then subtract out R from Q and repeat the process until $\text{Bool}(Q)$ is arc consistent (or $c_\emptyset^{R'} = \top$). Since the VCSP Q always has non-fractional weights, this process necessarily converges in finite time.

Let R_i (for $i = 1, \dots, s$) be the i th such VCSP which is subtracted out. All weights in R_i belong to $\{0, 1, \top\}$. The algorithm halts when the remaining problem Q is virtual arc consistent. When this occurs the original VCSP can be expressed as the sum

$$\begin{aligned} P &\equiv Q \oplus R_1 \oplus \dots \oplus R_s \\ &\equiv Q \oplus R'_1 \oplus \dots \oplus R'_s. \end{aligned}$$

This basic idea can be improved in several ways. Firstly, in the case where weights may be large integers, we can considerably optimize the algorithm. When the weights $c_I^Q(a)$ corresponding to assignments (I, a) such that $k(I, a) > 0$ (i.e. those weights in Q which lead to an inconsistency in $\text{Bool}(Q)$) are all no less than some integer $w \geq 2$, then we can simultaneously subtract out w copies of R from Q rather than tracing back and propagating w times.

Secondly, if the value of λ is 1, then we can simply apply the **Traceback** and **Propagate** operations directly to Q , rather than creating a new problem R_i .

Another improvement we can make to our basic algorithm follows from Proposition 8.38 below, which tells us that the integer part of any weights in R'_i can be added back to Q .

Definition 8.37 A VCSP S is integral if $\forall x \in L(N) \text{Val}_S(x) \in \mathbb{N} \cup \{\infty\}$.

An integral VCSP may nevertheless have cost functions which take on fractional values. The VCSP shown in the rightmost box of Figure 8.14(c) is an example.

Proposition 8.38 Let S be an integral VCSP. Suppose that $0 < c_\emptyset^S < 1$ and $c_I^S(a) \geq 1$ for some particular (I, a) . Let S' be identical to S except that $c_I^{S'}(a) = c_I^S(a) - 1$. Then $\text{Val}_{S'} \geq 1$.

Proof: S' is also clearly an integral VCSP with $c_\emptyset^{S'} > 0$. Hence $\text{Val}_{S'} \geq 1$. ■

Definition 8.39 Let P be a binary VCSP with non-fractional weights. A VAC decomposition of P is a set of binary VCSPs Q, R_1, \dots, R_s such that

1. $P \equiv Q \oplus R_1 \oplus \dots \oplus R_s$;
2. Q has non-fractional weights and is virtual arc consistent;
3. Each R_i is integral, has weights in $[0, 1) \cup \top$ (over a rational extension of the valuation structure of P) with the nullary constraint satisfying $0 < c_{\emptyset}^{R_i} < 1$.

The following lemma is an immediate consequence of the fact that an integral VCSP R_i with $c_{\emptyset}^{R_i} > 0$ satisfies $Val_{R_i} \geq 1$.

Lemma 8.40 If $P = Q \oplus R_1 \oplus \dots \oplus R_s$ is a VAC decomposition of a binary VCSP P , then $c_{\emptyset}^Q \oplus s$ is a lower bound on Val_P , where c_{\emptyset}^Q is the nullary constraint in Q .

If $P \equiv Q \oplus R_1 \oplus \dots \oplus R_s$ is a VAC decomposition, we can apply any local consistency operations to each of the VCSPs R_i . For example, if after applying FDAC to R_i , some weight $c_I^{R_i}(a)$ is now greater than 1, we can add 1 back to $c_I^Q(a)$ (by Proposition 8.38). We can, of course, apply any form of local consistency.

The following theorem follows from the fact that arc consistency can be established in $O(ed^2)$ time in $\text{Bool}(P)$. When the VAC decomposition algorithm is integrated into a branch-and-bound search, it could be combined with dynamic arc consistency algorithms [9] to improve efficiency.

Theorem 8.41 A VAC decomposition of a VCSP over the valuation structure S_m can be determined in $O(ed^2m)$ time.

Li et al. [100] find a lower bound for a MAX-SAT instance P by effectively establishing a form of generalized arc consistency in $\text{Bool}(P)$. (In fact, they apply a weak form of generalized arc consistency since clauses with identical scopes are not merged into a single constraint). If a contradiction is detected, then all clauses used are eliminated and the process iterates. Our algorithm differs in that we do not subtract out whole constraints but only the weights which are necessary to obtain a lower bound of 1. As a concrete example, consider the instance of MAX-SAT given by $X_1; \neg X_1 \vee \neg X_2; X_2; \neg X_1 \vee \neg X_3; X_3; \neg X_2 \vee \neg X_3$. The first three clauses are mutually inconsistent. Eliminating these three clauses leaves the three mutually consistent clauses $\neg X_1 \vee \neg X_3; X_3; \neg X_2 \vee \neg X_3$. Our VAC decomposition algorithm, on the other hand, effectively replaces the first three clauses by $X_1 \vee X_2$ (thanks to the forward propagation step which, in this example, simply establishes SAC on the the subproblem on variables $\{X_1, X_2\}$). The remaining four clauses ($X_1 \vee X_2; \neg X_1 \vee \neg X_3; X_3; \neg X_2 \vee \neg X_3$) are mutually inconsistent, which allows us to find a total lower bound of 2 instead of 1.

8.12 Soft Neighbourhood Substitution

A potentially important reduction operation in optimization problems is the elimination from variable domains of values which are inessential for the construction of an optimal solution.

Definition 8.42 *If $t \in d_1 \times \dots \times d_n$, then let $t[i, a]$ represent the tuple t' which is identical to t except that $t'_i = a$. In a VCSP, a value $a \in d_i$ is substitutable for $b \in d_i$ if $\forall t \in d_1 \times \dots \times d_n$ such that $t_i = b$, the total cost of $t[i, a]$ is no greater than the total cost of t , i.e. $Val(t[i, a]) \leq Val(t)$. The corresponding substitution operation is the elimination of b from domain d_i .*

It is clearly NP-hard to test for substitutability in a VCSP. We therefore define a tractable local version which is the generalization of neighbourhood substitution from CSPs to VCSPs. We first require the following definition of a total order on $S \times S$, where $\langle S, \oplus, \geq \rangle$ is the valuation structure of the VCSP. To gain a broad understanding of the remainder of this section, the reader can think of (a, b) as representing $a - b$, where a and b are real numbers. However, we cannot formally identify (a, b) with $a - b$ or with $a \ominus b$ since $<$ is defined over all pairs (a, b) , even when $(a, b) = (\infty, \infty)$ or when $b > a$.

Definition 8.43 *If $\langle S, \oplus, \geq \rangle$ is a valuation structure, then the relation $<$ on $S \times S$ is defined by*

$$\begin{aligned} (a, b) < (c, d) &\Leftrightarrow (a \oplus d < b \oplus c) \\ &\vee (a \oplus d = b \oplus c \wedge b > d) \\ &\vee (b = d = \top \wedge a < c). \end{aligned}$$

It was shown in [39] that $<$ is a total order on $S \times S$ whenever \oplus is strictly monotonic (which is the case if \oplus is the standard addition operation in $\mathbb{R} \cup \{\infty\}$).

Definition 8.44 *Consider a VCSP on a valuation structure $\langle S, \oplus, \geq \rangle$ with a strictly monotonic aggregation operator \oplus . For a pair of values $a, b \in d_i$ and a constraint c_P such that $i \in P$, define a best block of $(b \rightarrow a, i)$ on P to be a tuple $t \in L(P - \{i\})$ such that $(c_P(a, t), c_P(b, t))$ is maximal according to the total order $<$ defined in Definition 8.1, where (a, t) represents the extension of t in which $X_i = a$.*

A best block of the neighbourhood substitution $(b \rightarrow a, i)$ on P is a labelling t of the variables in $P - \{i\}$ such that

$$(c_P(a, t), c_P(b, t)) \geq (c_P(a, s), c_P(b, s))$$

for all labellings s of $P - \{i\}$. It is a labelling of the variables in $P - \{i\}$ which least supports the substitution of b by a at X_i .

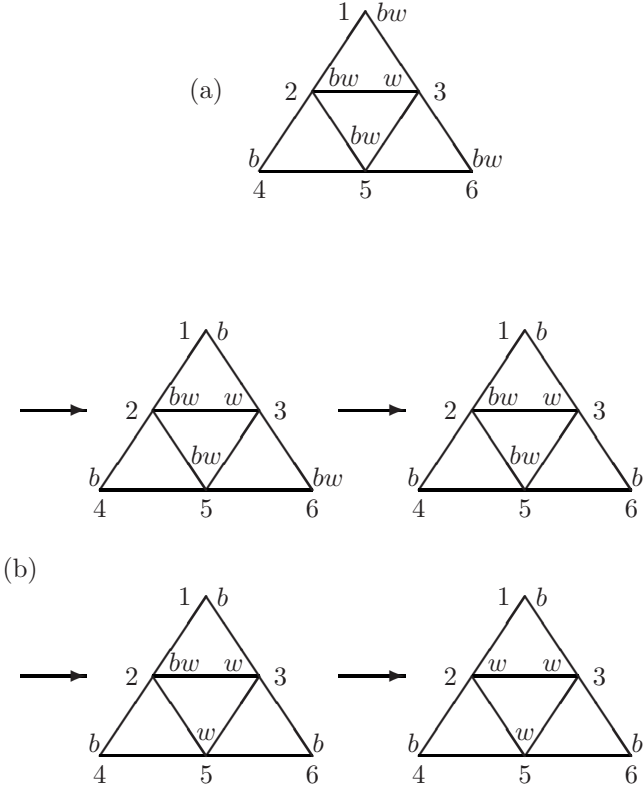


Figure 8.15: (a) A graph-colouring problem considered as a VCSP; (b) a sequence of neighbourhood substitutions.

Definition 8.45 In a VCSP with a strictly monotonic aggregation operator \oplus , given labels $a, b \in d_i$, a is neighbourhood substitutable for b at X_i if

$$\bigoplus_{P \in C_i} c_P(b, t_P) \geq \bigoplus_{P \in C_i} c_P(a, t_P),$$

where each t_P is the best block of $(b \rightarrow a, i)$ on P , and C_i is the set of constraint scopes P such that $i \in P$.

It is important to note that compensation can occur between the different valued constraints in C_i ; we do not require that a be substitutable for b in each individual constraint c_P . However, considering the VCSP as a whole, it is easy to prove that if $(b \rightarrow a, i)$ is a valid neighbourhood substitution, then it is a valid substitution according to Definition 8.42 [39].

When a is neighbourhood substitutable for b at X_i , the label b can be eliminated from d_i without any risk of increasing the cost of the optimal solution to

the VCSP. Such eliminations can propagate in the same way that eliminations by neighbourhood substitution can propagate in CSPs. Figure 8.15(a) shows a simple VCSP instance in which the aim is to colour the nodes of the graph with just two colours while minimizing the number of pairs of adjacent nodes assigned the same colour. Nodes 3 and 4 have only one possible label, whereas all other nodes can be labelled either *black* or *white* (shortened to *b* and *w* in the figure). Figure 8.15(b) shows that result of a sequence of neighbourhood substitutions. The label *white* can be eliminated from domains d_1 and d_6 because one of the two adjacent nodes has the unique label *white*. The label *black* can then be eliminated from domains d_2 and d_5 since two of the four adjacent nodes have the unique label *black*. This is a particularly favourable example, since an optimal solution is found without search.

Since a CSP is a just a VCSP over the valuation structure $\{\perp, \top\}$, we can also apply Definition 8.45 to a CSP. In this case, a label can be eliminated by soft neighbourhood substitutability (Definition 8.45) if and only if it can be eliminated by either crisp neighbourhood substitution (Definition 8.7) or crisp generalized arc consistency (Definition 8.6).

Let c denote the number of constraints in the VCSP, d the maximum domain size, and k (a constant) the maximum arity of the constraints. When costs belong to $\mathbb{R} \cup \{\infty\}$, a convergent sequence of eliminations by neighbourhood substitutability can be found and applied in $O(d^{k+2}c)$ time and $O(d^2c)$ space [39]. In VCSPs in which all costs are finite, such as MAX-CSP, arc consistency produces a lower bound but does not immediately reduce the size of the search space. Although neighbourhood substitution is more costly in time to apply, it can actually reduce the size of the search space, by eliminating domain values, before embarking on an exhaustive search.

8.13 Discussion

Arc consistency is today ubiquitous in constraint processing. It is interesting to note that it first came to light as the result of pioneering work on line drawing interpretation [173]. Moving away from the rather idealistic problems first studied, towards the interpretation of realistic line drawings such as hand-drawn sketches, has necessarily led to the introduction of optimization criteria. In this book we have nevertheless retained one essential property of constraint satisfaction problems, namely the finiteness of domains. We consider the discovery of the topology of objects as well as the identification of properties such as convex/concave edges, cubic corners, orthogonal edges, planar faces, etc. as the most important part of picture interpretation. We have seen that all the resulting finite-domain variables can be combined in a single VCSP.

Traditional arc consistency can be generalized in several different ways to the valued constraint framework due to the fact that the arc consistency closure is not unique. Four distinct ideas have been shown to be useful in finding a better lower bound via SAC operations:

- Sending weights in the same direction (FDAC, EDAC)

- Allowing projections and extensions of rational (rather than integer) costs (OSAC)
- Allowing simultaneous consistency operations (VAC)
- Decomposing the VCSP into the sum of problems (VAC decomposition)

OSAC is too costly in computational resources to be applied during search but could be applied once before branch-and-bound search [48]. The best arc consistency operation to apply during search varies from application to application. In general, applying a stronger form of arc consistency requires more time and hence may turn out to be counterproductive if only a small amount of extra pruning of the search tree occurs. For example, FDAC, although weaker than EDAC, was found to be slightly more efficient on optimal planning problems [47]. On frequency assignment problems, on the other hand, EDAC was found to be much more efficient than FDAC [54].

If a VCSP is virtual arc consistent, then this means that no sequence of SAC operations could increase the lower bound c_\emptyset . In particular, FDAC and EDAC cannot increase c_\emptyset for any variable order. Thus VAC is certainly theoretically stronger than FDAC and EDAC. Unfortunately, establishing VAC may lead to the introduction of smaller and smaller fractional weights. We therefore suggest finding a VAC decomposition since this is guaranteed to terminate and can even provide a better lower bound than establishing VAC. A disadvantage of VAC decomposition is that we no longer produce a single problem equivalent to the original problem. Another practical solution is to establish VAC on \mathcal{P}_ϵ , where \mathcal{P}_ϵ is identical to the original VCSP \mathcal{P} except that all costs less than or equal to ϵ are rounded down to zero. [49].

Chapter 9

Tractability of Drawing Interpretation

9.1 Tractable Constraint Classes

In this section we introduce the basic theoretical tools that have been developed for the analysis of the tractability of constraint satisfaction and valued constraint satisfaction problems. In the next section we apply some of these results to problems related to the interpretation of line drawings. A problem is considered to be tractable if there is a polynomial-time algorithm to solve it, and intractable if not. Assuming $\text{NP} \neq \text{P}$, any problem that is NP-hard is intractable.

9.1.1 Zero/One/All Constraints

We can consider the class of zero/one/all (ZOA) constraints to be a generalization of 2SAT to the constraint satisfaction problem (CSP). In constraint satisfaction, domains are of arbitrary finite size, whereas in SAT they are boolean.

If R_{ij} is a binary relation on variables $\langle X_i, X_j \rangle$, then we say that $u \in d_i$ is *compatible with* $v \in d_j$ iff $\langle u, v \rangle \in R_{ij}$. Recall that $\pi_i R_{ij}$ represents the projection of the binary relation R_{ij} onto variable X_i .

Definition 9.1 *A binary constraint $\langle \langle X_i, X_j \rangle, R_{ij} \rangle$ is 0/1/all (ZOA) if each $u \in d_i$ is compatible with 0,1 or all the elements of $\pi_j R_{ij}$ and each $v \in d_j$ is compatible with 0,1 or all the elements of $\pi_i R_{ij}$.*

The ZOA tractable class of constraints was first discovered simultaneously by Kirousis [91] and Cooper et al. [46]. There is an alternative characterization of ZOA constraints based on the notion of component-wise closure operation, called polymorphism, defined below.

Definition 9.2 [86] *Given a relation $R \subseteq D^r$, the function $f : D^t \rightarrow D$ is a*

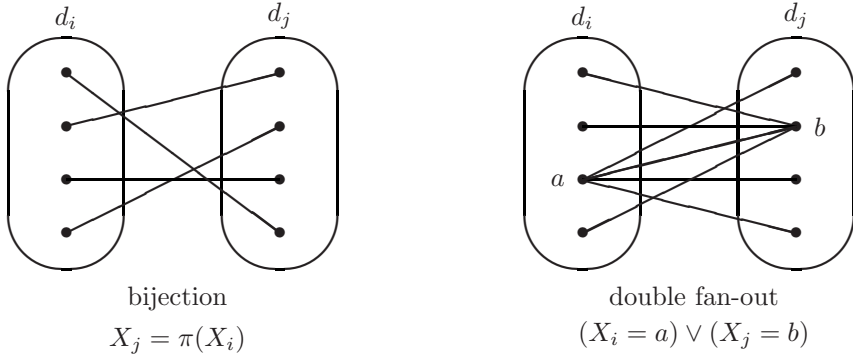


Figure 9.1: The two types of binary ZOA constraints: each oval represents a domain, each bullet a domain value and each line a compatible pair of values (i.e. an element of the constraint relation).

polymorphism of R if $\forall \langle x_{11}, \dots, x_{1r} \rangle, \dots, \langle x_{t1}, \dots, x_{tr} \rangle \in R,$

$$\langle f(x_{11}, \dots, x_{t1}), \dots, f(x_{1r}, \dots, x_{tr}) \rangle \in R.$$

Theorem 9.3 [86] *A binary relation R_{ij} is ZOA iff it has the polymorphism Mjty, where*

$$\text{Mjty}(x, y, z) = \begin{cases} y & \text{if } y = z, \\ x & \text{otherwise.} \end{cases}$$

If the multiset $\{x, y, z\}$ contains a majority element, i.e. the same value which occurs at least twice, then the function $\text{Mjty}(x, y, z)$ returns this element. Any function which satisfies this property is known as a majority function.

Theorem 9.4 [85] *If the r -ary relation R is closed under a majority function, then R is decomposable into its binary projections, i.e.*

$$R = \bowtie_{1 \leq i < j \leq n} R_{ij}.$$

To characterize ZOA relations we therefore only need to consider unary and binary relations. A unary relation is simply a subset A of the domain. It is easily verified that all unary constraints (i.e. $X_i \in A$) are ZOA. It was shown in [46] that binary ZOA constraints (which are not decomposable into unary constraints) have one of only two possible forms (illustrated in Figure 9.1):

- $X_j = \pi(X_i),$ where π is a bijection;
- $(X_i = a) \vee (X_j = b)$ (a double fan-out constraint).

Theorem 9.5 *Let $\text{CSP}(\Gamma_{\text{ZOA}})$ be the set of instances \mathcal{P} of the CSP such that all of the constraint relations of \mathcal{P} are ZOA. Then $\text{CSP}(\Gamma_{\text{ZOA}})$ can be solved in polynomial time.*

Proof: We give a polynomial-time algorithm to solve all $\mathcal{P} \in \text{CSP}(\Gamma_{\text{ZOA}})$. First of all we eliminate all bijection constraints by merging variables X_i, X_j if they are linked by such a constraint. The remaining constraints are either double fan-outs or unary constraints. This CSP \mathcal{P} can then be solved by reduction to 2SAT, for which there is a polynomial-time algorithm [114]. In the instance $I_{\mathcal{P}}$ of 2SAT, there is a boolean variable $V_{i,c}$ for each proposition of the form $X_i \leq c$ ($c \in d_i$). Without loss of generality, suppose that $\forall i d_i = \{1, \dots, d\}$. In $I_{\mathcal{P}}$, there are three types of clauses:

1. Clauses to code the fact that $X_i \in d_i = \{1, \dots, d\}$:
 $X_i \leq d$ and $(\forall c \in d_i) X_i \leq c \Rightarrow X_i \leq c + 1$;
2. Clauses to code each double fan-out $(X_i = a) \vee (X_j = b)$:
 $X_i \leq a - 1 \Rightarrow X_j \leq b, X_i \leq a - 1 \Rightarrow \neg(X_j \leq b - 1),$
 $\neg(X_i \leq a) \Rightarrow X_j \leq b, \neg(X_i \leq a) \Rightarrow \neg(X_j \leq b - 1)$;
3. Clauses to code each unary constraint $X_i \in A$:
 $(\forall c \in d_i - A) X_i \leq c \Rightarrow X_i \leq c - 1.$

■

9.1.2 Max-Closed Constraints

In this section, we suppose that each domain has a total ordering. We introduce a tractable class of constraints which can be seen as a generalization of Horn clauses to domains of arbitrary finite size.

Definition 9.6 [84] *A relation R is max-closed if it has the polymorphism \max , i.e.*

$$(x_1, \dots, x_r), (y_1, \dots, y_r) \in R \implies (\max\{x_1, y_1\}, \dots, \max\{x_r, y_r\}) \in R.$$

The following lemma follows by a simple inductive proof from the above definition.

Lemma 9.7 *Let R be a max-closed relation. If $x^1, x^2, \dots, x^t \in R$, where $x^k = (x_1^k, \dots, x_r^k)$, then*

$$(\max\{x_1^k : k = 1, \dots, t\}, \dots, \max\{x_r^k : k = 1, \dots, t\}) \in R.$$

Example 9.8 Examples of max-closed constraints:

- Unary:* $X_i \in A$ (all unary constraints are max-closed);
- Arithmetical:* $X_i = X_j + c, X_i \geq X_j + c, X_i + X_j \geq X_k + c, \dots$ [84];
- Logical:* Horn clauses (with the order **false** > **true**) [84].

Examples of constraints which are not max-closed:

- Arithmetical:* $X_i \neq X_j, X_i + X_j \leq X_k + c, X_i + X_j = c, \dots$;
- Logical:* $X_i \vee X_j$ (with the order **false** > **true**).

□

Theorem 9.9 *Let $CSP(\Gamma_{max})$ be the set of instances \mathcal{P} of the CSP such that all of the constraint relations of \mathcal{P} are max-closed relations of order no greater than k , for some constant k . Then $CSP(\Gamma_{max})$ can be solved in polynomial time.*

Proof: We can establish generalized arc consistency in polynomial time since the order of the constraints is bounded by a constant. Indeed, the algorithm AC2001 [10] is easily adapted so that it establishes generalized arc consistency in $O(cd^k)$ time, where c is the number of constraints and d the maximum domain size. Establishing generalized arc consistency cannot destroy the max-closed property of constraints in \mathcal{P} since it only updates domains and all unary constraints are max-closed. Let d_i be the domain of variable X_i ($i = 1, \dots, n$) after establishing generalized arc consistency. Clearly, if $d_i = \emptyset$ for some i , then \mathcal{P} has no solution. Otherwise, let $a_i = \max(d_i)$. Generalized arc consistency implies that, for each constraint $\langle S, R_S \rangle$,

$$\forall X_i \in S \quad \exists x^i = (x_1^i, \dots, x_s^i) \in R_S \quad \text{such that} \quad x^i[X_i] = a_i.$$

By Lemma 9.7, $(a_1, \dots, a_n)[S] \in R_S$. Therefore, (a_1, \dots, a_n) is a solution to \mathcal{P} . ■

9.1.3 Characterization of Tractable Boolean Constraints

We use the notation $CSP(\Gamma)$ to represent the set of instances \mathcal{P} of the constraint satisfaction problem such that all constraint relations in \mathcal{P} belong to Γ . P is the set of decision problems which can be solved in polynomial time.

Definition 9.10 *A constraint class Γ is tractable if $CSP(\Gamma) \in P$.*

If c is a domain element, we can consider c a constant function. A set of constraints which is closed under a constant polymorphism c (known as the set of c -closed constraints) can trivially be solved by assigning the value c to each variable. Horn clauses are disjunctions of literals in which at most one of these literals is a non-negated variable ($X_1 \vee \neg X_2 \vee \neg X_3$ and X_1 are just two examples). Anti-Horn clauses are disjunctions involving at most one negated variable. A constraint relation R is *affine* if R is the set of solutions to a set of linear equations (for example, $R = \{ \langle x_1, x_2, x_3, x_4 \rangle \in \{0, 1\}^4 : x_1 + x_2 + x_3 = 0 \pmod 2 \wedge x_3 = x_4 \}$).

Theorem 9.11 (Schaefer [138]) *The only tractable classes of constraints over boolean domains are subsets of one of the following*

- *The set of 0-closed constraints,*
- *The set of 1-closed constraints,*
- *The set of 2SAT clauses,*

- The set of Horn clauses,
- The set of anti-Horn clauses,
- The set of affine constraints.

Although no equivalent theorem has as yet been proved for non-boolean domains, considerable progress has nevertheless been made. For example, it is known that the presence of a polymorphism is a necessary condition for a class of constraints to be tractable over domains of arbitrary finite size [83]. Furthermore, Bulatov has made a conjecture concerning an exact characterization of tractable classes over such domains and has proved the validity of the conjecture for size-3 domains [16]. An excellent survey of this field can be found in [25].

9.1.4 Characterization of Tractable Boolean Valued Constraints

VSAT is a generalization of SAT which allows the user to express preferences between solutions. It is simply the valued constraint satisfaction problem (see Definition 8.16) over boolean domains and over the valuation structure $\overline{\mathbb{R}}_+ = \{z \in \mathbb{R} : z \geq 0\} \cup \{\infty\}$ (i.e. costs are either non-negative real numbers or infinite). The version of VSAT given below is a decision problem. In the optimization version of VSAT, the aim is to find a solution of minimal total cost.

VSAT

Instance: n boolean variables X_1, \dots, X_n , a set of valued constraints $\langle c_{S_i}, S_i \rangle$ ($i \in \{1, \dots, m\}$), where for each i , $S_i \subseteq \{X_1, \dots, X_n\}$ and $c_{S_i} : \{0, 1\}^{|S_i|} \rightarrow \overline{\mathbb{R}}_+$ is a cost function, together with an integer k .

Question: Does there exist an assignment t to the variables X_1, \dots, X_n such that

$$\sum_{i=1}^m c_{S_i}(t[S_i]) \leq k$$

We can observe that SAT is the special case of VSAT in which the cost functions only take on values in $\{0, \infty\}$. In what follows, we identify a propositional formula F (such as $X_1 \vee \neg X_2$) with the cost function which returns 0 if F is satisfied and ∞ otherwise.

Example 9.12 We can encode the search for a minimum cut in a weighted directed graph G as a VSAT instance \mathcal{P} with a variable for each node of G and a valued constraint $\langle \chi^{w_{ij}}, \chi^{w_{ij}} \rangle$ for each directed edge $\langle i, j \rangle$ of weight w_{ij} in G , where

$$\chi^w(x, y) = \begin{cases} w & \text{if } (x, y) = (0, 1), \\ 0 & \text{otherwise.} \end{cases}$$

If we impose unary constraints on the source and target nodes to ensure that they take the values $\{0\}$ and $\{1\}$, respectively, then a minimum cut in G corresponds to the set of directed edges $\langle i, j \rangle$ whose corresponding variables are assigned $(0, 1)$ in a solution to \mathcal{P} . \square

Definition 9.13 ([24]) *A list of functions, $\langle f_1, \dots, f_m \rangle$, where each f_i is a function from D^m to D , is a multimorphism of a cost function $\phi : D^r \rightarrow \overline{\mathbb{R}}_+$ if, for all $\langle a_{11}, \dots, a_{1r} \rangle, \dots, \langle a_{m1}, \dots, a_{mr} \rangle \in D^r$, we have*

$$\sum_{i=1}^m \phi(f_i(a_{i1}, \dots, a_{im}), \dots, f_i(a_{ir}, \dots, a_{mr})) \leq \sum_{i=1}^m \phi(a_{i1}, \dots, a_{ir}). \quad (9.1)$$

Note that if $\langle f_1, \dots, f_m \rangle$ is a multimorphism of a cost function ϕ , then the average cost of a set of m assignments is lowered by applying the functions f_1, \dots, f_m co-ordinatewise. A trivial example of a multimorphism is $\langle c \rangle$, where c is a constant; if all cost functions in an instance of VSAT have the multimorphism $\langle c \rangle$, then an optimal solution can be obtained by assigning c to each variable.

As another trivial example of a multimorphism, define two functions $\text{Mjty}, \text{Mnty} : \{0, 1\}^3 \rightarrow \{0, 1\}$ as follows: $\forall x, y \in \{0, 1\}$, $\text{Mjty}(x, x, y) = \text{Mjty}(x, y, x) = \text{Mjty}(y, x, x) = x$ and $\text{Mnty}(x, x, y) = \text{Mnty}(x, y, x) = \text{Mnty}(y, x, x) = y$. If exactly two out of their three arguments are equal, then Mnty returns the minority element and Mjty returns the majority element. In [24] it was shown that, over boolean domains, the only cost functions having the multimorphism $\langle \text{Mjty}, \text{Mjty}, \text{Mnty} \rangle$ are the propositional formulae $X_i = X_j$ and $X_i \neq X_j$ together with all unary cost functions (i.e. functions with a single argument). There is a simple algorithm to solve an instance of VSAT with only these types of valued constraints: firstly eliminate all propositional formulae by merging variables (which involves permuting domain elements in the case of $X_i \neq X_j$ constraints); if no contradiction is discovered during this stage, then for each variable X_i , simply assign to X_i the value which has the least cost according to the unary cost function on X_i .

A non-trivial class of tractable valued constraints is the class of submodular cost functions.

Definition 9.14 *A cost function is submodular if it has the multimorphism $\langle \min, \max \rangle$.*

Submodular function minimization (SFM) [67, 159] is a tractable discrete optimization problem which has applications in such diverse areas as statistical physics [6] and the design of electrical networks [121]. Well-known examples of submodular functions are the cut function of a graph [52] (Example 9.12) or of a hypergraph [68], and the rank function of a matroid.

The ellipsoid algorithm provides a polynomial-time algorithm for SFM in theory, but is not efficient in practice [71]. Recently, several more efficient

polynomial-time algorithms have been published to solve SFM [81, 148, 79, 80, 123]. The fact that these algorithms can be applied to minimize a submodular function defined on a distributive lattice [79] (also known as a ring family [148]) has been used to show that they can be applied to submodular functions which may take on both finite and infinite values over totally-ordered finite domains of arbitrary size [24]. The complexity of the fastest known algorithm for SFM is $O(n^5\gamma + n^6)$ where n is the number of variables and γ is the time to calculate the objective function [123]. Nevertheless, for certain special cases of SFM, more efficient algorithms exist. For example, MINIMUM WEIGHTED CUT (see Example 9.12) is a special case of SFM that can be solved in cubic time [69]. The minimization of the sum of binary submodular cost functions over domains of arbitrary finite size [23] can be seen as a generalization of MINIMUM WEIGHTED CUT.

A VCSP is *permuted-submodular* if its cost functions can be made submodular by applying possibly different permutations to each variable domain. Schlesinger [141] showed that given a permuted-submodular VCSP with finite costs, domain permutations which render it submodular can be found in polynomial time by reduction to 2SAT. It has also recently been shown that submodular or permuted-submodular VCSP can also be solved directly by establishing OSAC [44] or VAC [49].

The importance of submodularity is emphasized by the following theorem which characterizes all tractable classes of VSAT constraints.

Theorem 9.15 [24]: *The only tractable classes of valued constraints over boolean domains are subsets of one of the following:*

- *The set of cost functions with the $\langle 0 \rangle$ multimorphism;*
- *The set of cost functions with the $\langle 1 \rangle$ multimorphism;*
- *The set of cost functions with the $\langle Mjty, Mjty, Mnty \rangle$ multimorphism;*
- *The set of 2SAT clauses;*
- *The set of affine constraints;*
- *The set of cost functions with the $\langle \min, \min \rangle$ multimorphism, which are the set of Horn clauses together with monotone cost functions;*
- *The set of cost functions with the $\langle \max, \max \rangle$ multimorphism, which are the set of anti-Horn clauses together with antitone cost functions;*
- *The set of cost functions with the $\langle \min, \max \rangle$ multimorphism, i.e. the set of submodular functions.*

An interesting recent theoretical development is the merging of the last three classes in the list in Theorem 9.15, in the case of non-boolean domains [22].

Definition 9.16 A tournament operation is a binary operation $f : D^2 \rightarrow D$ with the following properties:

- f is conservative: $f(x, y) \in \{x, y\}$, for all $x, y \in D$.
- f is commutative: $f(x, y) = f(y, x)$, for all $x, y \in D$.

A tournament pair is a pair $\langle f, g \rangle$, where f and g are both tournament operations.

Both max and min are tournament operations, and $\langle \min, \min \rangle$, $\langle \max, \max \rangle$, $\langle \min, \max \rangle$ are all tournament pairs. It was shown in [22] that over domains of any finite size, a set of cost functions having any tournament pair $\langle f, g \rangle$ as a multimorphism is tractable.

9.2 Complexity of Line Drawing Interpretation

A fundamental result on the complexity of line drawing interpretation was proved by Kirousis and Papadimitriou [92]: determining whether a line drawing of a polyhedral scene has a legal labelling according to the Huffman–Clowes catalogue (of projections of trihedral vertices) is NP-complete. They also proved that determining whether a line drawing is realizable as the projection of a polyhedral scene involving only trihedral vertices is NP-complete. Realizability is a stronger condition than labelability since some legally labelled drawings are not realizable (such as the Penrose triangle illustrated in Figure 2.4 or Escher’s ‘Belvedere’ illustrated in Figure 2.7).

Surprisingly, when all object surfaces are curved and there are no linear features such as straight lines, collinearity or planarity, then these two problems become solvable in polynomial time. As we show below, this follows from the fact that the labelling constraints are max-closed.

Since all lines are curved, label transitions can occur between occluding and convex labels on any line in a drawing [34]. This means that the difference between the labels $+$, \rightarrow and \leftarrow cannot be propagated along a line. We introduce a new label δ to represent any of the set of labels $\{+, \rightarrow, \leftarrow\}$. The same convex edge, when viewed from different viewpoints, can project into a line with any one of these three different labels. The label δ gives the essential information about the structure of the edge (it is convex) without specifying the position of the viewpoint in relation to the two surfaces meeting at the edge.

In Figure 9.2 we give the simplified catalogue of labelled junctions for objects composed of C^3 surfaces meeting non-tangentially at trihedral vertices, in which $+$, \rightarrow and \leftarrow labels have been replaced by δ . For compactness of presentation, we do not show the reflected versions of terminal, curvature-L and 3-tangent junctions even though they are an essential part of the catalogue. An important point is that C junctions (also known as phantom junctions or virtual junctions) no longer appear in the catalogue, since no label transitions can occur within the reduced label set $\{\leftarrow, \Rightarrow, \delta, -\}$. Thus the labelling problem consists in assigning a unique label to each line.

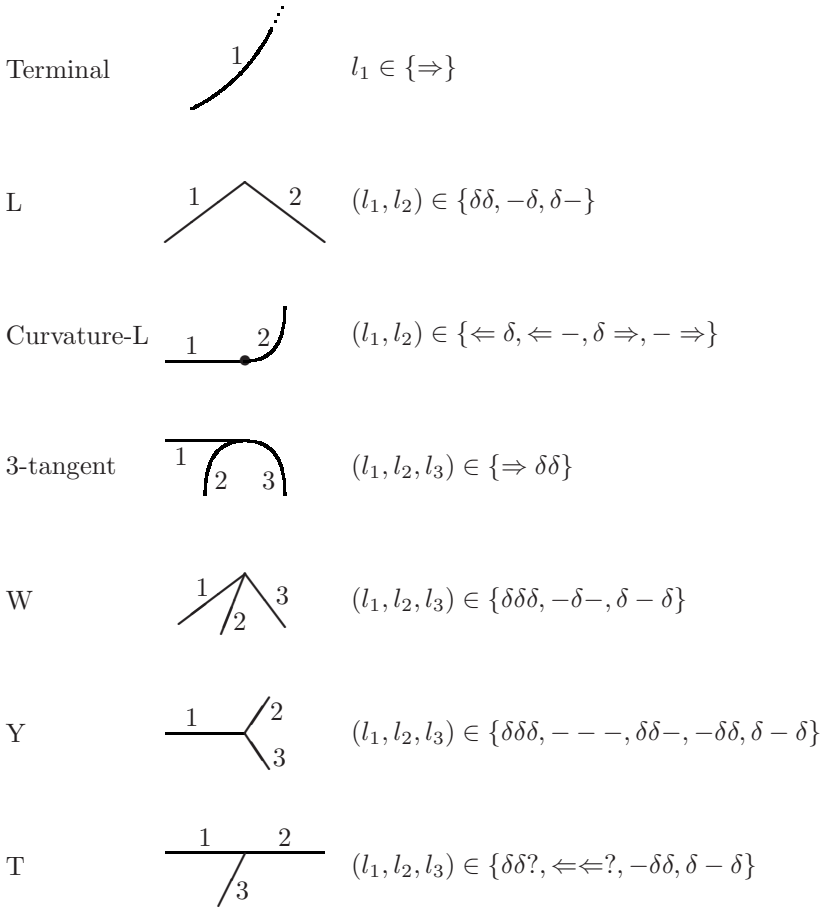


Figure 9.2: Catalogue of junction labellings for objects with C^3 surfaces meeting non-tangentially at trihedral vertices. For $i \in \{1, 2, 3\}$, l_i is the label of line i . A question mark represents any label.

Theorem 9.17 [36]. *Given a drawing of curved objects, composed of n lines, we can produce a legal global labelling, according to the curved-object trihedral catalogue (Figure 9.2), if it exists, or determine that no such labelling exists, in $O(n)$ time.*

Proof: We prove the theorem by showing that it is possible to express the line drawing labelling problem as a CSP with max-closed constraints.

To distinguish between the two labels \leftarrow and \Rightarrow , we must assign a direction to each line in the drawing. The direction of each line can be arbitrary, so we choose them to be consistent for all lines in the same cycle or chain of curvature-L junctions, for example, clockwise for all lines in the same cycle. The result is that all curvature-L junctions J have one line entering and one line leaving J . Similarly, at each T junction J , we choose the directions of the two lines forming the bar of the T to be identical, so that one line enters and one line leaves J .

Consider a line drawing labelling problem with constraints derived from the catalogue of Figure 9.2. Due to the arbitrary choice of the directions assigned to the majority of lines in the drawing, the constraints which can occur in the labelling problem are those shown in Figure 9.2 with the direction of any number of lines reversed. Reversing the direction of a line means interchanging the labels \Rightarrow and \leftarrow . The exceptions are curvature-L junctions and the bars of T junctions which, by our choice of directions, always have one line entering and one line leaving. The possible constraints are therefore:

$\{\Rightarrow\}$ or $\{\leftarrow\}$	for terminal junctions
$\{\delta\delta, -\delta, \delta-\}$	for L junctions
$\{\leftarrow \delta, \leftarrow -, \delta \Rightarrow, - \Rightarrow\}$ or $\{\Rightarrow \delta, \Rightarrow -, \delta \leftarrow, - \leftarrow\}$	for curvature-L junctions
$\{\Rightarrow \delta\delta\}$ or $\{\leftarrow \delta\delta\}$	for 3-tangent junctions
$\{\delta\delta\delta, -\delta-, \delta - \delta\}$	for W junctions
$\{\delta\delta\delta, - - -, \delta\delta-, -\delta\delta, \delta - \delta\}$	for Y junctions
$\{\delta\delta\delta, \delta\delta-, \delta\delta \Rightarrow, \delta\delta \leftarrow, \leftarrow \leftarrow \delta, \leftarrow \leftarrow -, \leftarrow \leftarrow \Rightarrow, \leftarrow \leftarrow \leftarrow, -\delta\delta, \delta - \delta\}$ or $\{\delta\delta\delta, \delta\delta-, \delta\delta \Rightarrow, \delta\delta \leftarrow, \Rightarrow \Rightarrow \delta, \Rightarrow \Rightarrow -, \Rightarrow \Rightarrow \Rightarrow, \Rightarrow \Rightarrow \leftarrow, -\delta\delta, \delta - \delta\}$	for T junctions

We define the function $\max : L \times L \rightarrow L$, where L is the reduced label set of line labels, according to the ordering:

$$‘\Rightarrow’ < ‘-’ < ‘\delta’ < ‘\leftarrow’.$$

It is easy to verify that all the constraint relations given above are max-closed under this ordering. For example, applying the function \max pointwise to the two legal labellings $\delta\delta-$ and $-\delta\delta$ for a Y junction produces $\delta\delta\delta$, which is indeed a legal labelling for a Y junction.

As was shown in the proof of Theorem 9.9, a CSP with max-closed constraints can be solved in $O(cd^k)$ time by establishing generalized arc consistency, where c is the number of constraints, d the maximum domain size and k the maximum number of variables in a constraint. A drawing composed of n lines has $O(n)$ junctions, thus $c = O(n)$. The domain size is a constant $d = 4$, as is

the maximum number of variables in a constraint $k = 3$. Thus the CSP can be solved in linear time. ■

The outer-boundary constraint says that the outer boundary of a drawing is an occluding contour, consisting of \rightarrow and \Rightarrow labels. This is equivalent to imposing a unary constraint on all lines on the outer boundary of the drawing. Since all unary constraints are also max-closed, Theorem 9.17 remains valid when we apply the outer-boundary constraint.

Theorem 9.17 is robust to changes in the assumptions we make about object shape. For example, generalizing the catalogue of Figure 9.2 to include projections of apices of cones, to include projections of non-trihedral vertices or discontinuities of surface curvature produces another catalogue whose constraints are again max-closed [36]. The interpretation of line drawings of objects with possibly tangential edges and surfaces has also been shown to be solvable in linear time [34]. A similar result holds for pottery world objects [56].

A catalogue of junction labellings only provides a necessary condition which must be satisfied by the global labelling of a drawing: each junction must have a labelling found in the catalogue. In the case of polyhedral scenes, labelability is not a sufficient condition for realizability. However, in the case of objects with only curved C^3 surfaces, we have the freedom to choose arbitrary C^3 surfaces. It was shown in [36] that, in this case, any legally labelled drawing is realizable.

If lines in a drawing represent differences in brightness or colour between adjacent regions of an image, then no line should be present between two adjacent regions of identical brightness and colour. This phenomenon is known as *contrast failure* and is very common when line drawings are derived from real images. Contrast failure is inevitable between adjacent regions which are projections of parallel surfaces of identical surface characteristics subject to the same illumination. When contrast failure can occur between parallel surfaces, labelling a line drawing of objects with curved or planar surfaces is also solvable in linear time [38]. This follows from a reduction to 2SAT for which there is a linear-time algorithm [114].

Another problem which can be solved in linear time by reduction to 2SAT is determining the existence of a legal labelling of a drawing of a polyhedral scene in which objects have trihedral vertices, provided we are given (or can determine) the vanishing points of all lines meeting at Y and W junctions in the drawing [127]. This result generalizes to the case of curved objects when we are given the vanishing points of all line ends [36]. Unfortunately, if we can only determine the vanishing points of some, but not all, lines, then the problem becomes NP-complete, as we prove below.

Theorem 9.18 [36] *Labeling a line drawing of objects with C^3 surfaces when some of the vanishing points to line ends are known is NP-complete.*

Proof: The problem is clearly in NP since the validity of a labelling can be checked in polynomial time. To complete the proof it is sufficient to produce a polynomial transformation from a known NP-complete problem.

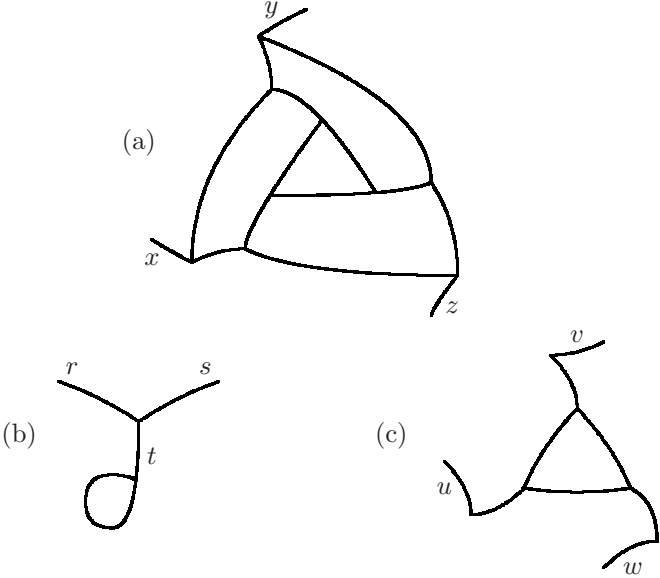


Figure 9.3: Constructions for (a) $x = y = z$, (b) $r = \neg s$, (c) $u \vee v \vee w$.

Lichtenstein [102] proved the NP-completeness of PLANAR 3SAT, a version of 3SAT in which the following bipartite graph G is planar: G has a vertex for each variable v , a vertex for each disjunction D and an edge for each pair (v, D) such that v or $\neg v$ is one of the three literals in D . In order to exhibit a polynomial reduction from PLANAR 3SAT to the line drawing labelling problem, we need to specify the coding of variables, show how to generate many copies of the same variable, give a negation construction and give a construction for $u \vee v \vee w$.

We code **true** as ‘ δ ’ and **false** as ‘ $-$ ’. Given the vanishing points of the three lines that meet at a Y junction J , we can classify J as a Y(+) or a Y(-) junction [127]. The set of legal labellings of a Y(-) junction is $\{- - -, \delta\delta -, \delta - \delta, -\delta\delta\}$ (which is no longer max-closed due to the fact that it does not contain $\delta\delta\delta$). In Figure 9.3, all Y junctions are Y(-) junctions, but all W junctions are actual W junctions (and not W(+) or W(-) junctions). Figure 9.3(a) shows a construction to generate two copies y, z of the variable x . There are only two legal labellings of this line drawing: in one $x = y = z = \delta$ and in the other $x = y = z = -$. By chaining together $N - 1$ copies of this construction we can generate N copies of the same variable x . Figure 9.3(b) is a negation construction ($s = \neg r$). Since t must take on the value ‘ δ ’, the only two legal assignments are $(r = -, s = \delta)$ and $(r = \delta, s = -)$. Figure 9.3(c) is the construction for the disjunction of three literals u, v, w . It can easily be verified that all assignments of values to u, v, w are possible except $u = v = w = -$. The construction thus imposes the condition $u \vee v \vee w$. ■

Most computational problems corresponding to the interpretation of line drawings containing linear features (such as straight lines, parallel lines, planar faces, collinearity) have turned out to be intractable. The following is a list of problems that have been shown to be NP-complete:

- The labelling of line drawings of polyhedral scenes [92];
- The realizability of a line drawing as the projection of a polyhedral scene [92];
- The labelling of line drawings of objects with only curved faces but for which we are given the vanishing points of some tangents to line-ends at junctions [36];
- The labelling of line drawings under orthographic projection which contain some parallel lines [36];
- Determining the existence of a legal labelling without label transitions in the labelling of a line drawing of curved objects [36]; minimizing the number of label transitions is NP-hard [36];
- The realizability of a line drawing of curved objects which may contain collinear line ends or points [36];
- The labelling of a line drawing with shadows of a scene involving curved objects (whether or not contrast failure can occur between parallel surfaces) [38];
- The labelling of a line drawing of an origami scene (even if we are given the vanishing points of all lines) [125]; an origami scene may contain both non-manifold objects, such as a sheet of paper, and solid objects with trihedral vertices.

A tractable class of (valued) constraints must have some kind of structure which a polynomial-time algorithm can exploit. From a theoretical point of view, it is known that a tractable class of crisp constraints must have a polymorphism [83] and that a tractable class of valued constraints must have a generalized form of multimorphism [21]. From a practical point of view, a disparate set of constraint types, such as collinearity and junction labellings, are unlikely to have a common structure (such as a polymorphism) which could be exploited by a polynomial-time algorithm. Thus, emulating human vision by integrating constraints arising from many diverse sources (junctions, vanishing points, planarity, etc.) almost inevitably leads to a theoretically intractable computational problem.

It is worth remembering that knowing that a problem \mathcal{P} is NP-hard only informs us about the worst-case performance of algorithms to solve \mathcal{P} . The completely contrived nature of the constructions in Figure 9.3 illustrate that the set of drawings that are actually encountered in practice constitute a small subset S of all possible drawings; a polynomial-time algorithm may exist for a subset of drawings including most (or even all) of S .

Chapter 10

3D Reconstruction of Ambiguous Pictures

10.1 Reconstruction of Frontal Geometry

It is well known that most drawings are ambiguous, if only in the size of the object depicted. However, more serious forms of ambiguity often arise, even in drawings of simple polyhedral objects. For example, Figure 10.1 shows some line drawings in which there is considerable ambiguity in the relative depths of object vertices.

In order to determine the most likely 3D reconstruction of a drawing of a polyhedral scene, several workers have used a local search technique (such as hill climbing [111, 7], Brent’s method [103, 122], simulated annealing [30], tabu search [1] or a genetic algorithm [160]) to optimize an objective function. The arguments of the objective function are usually the Z -coordinates of object vertices but may also include the parameters (p, q, r) of planes of object faces $z = px + qy + r$. Each of the linear constraints given in Chapter 6 should be applied, either as a strict constraint in order to reduce the number of free variables over which to search or in the form of a least-squares-error term to be minimized.

The choice of objective function is obviously critical. Marill’s MSDA

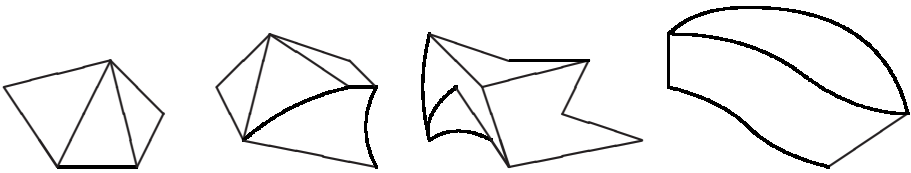


Figure 10.1: Examples of ambiguous pictures.

criterion prefers the interpretation with the minimum standard deviation of angles (SDA) [111]: given a 3D reconstruction, for each pair of edges which meet at a vertex, we can calculate the 3D angle between these two edges. Minimizing SDA, the standard deviation of the resulting set of angles $\{\theta_1, \dots, \theta_k\}$, is a heuristic which has been used by many workers since it naturally tends to “inflate” the scene, thus avoiding the trivial interpretation of the drawing as a completely flat scene. Linear constraints alone, based on coplanarity, parallel lines or collinearity, do not inflate since they are all satisfied by a completely flat scene.

Other criteria which tend to inflate objects include a preference for right angles (whether between pairs of edges, between pairs of faces or between edges and faces [103]) or for horizontal or vertical edges or faces [162]. To emphasize the fact that right angles are common features but that we have no preference between other angles, we can add a penalty function $f(\theta)$ which is constant except in the vicinity of $\theta = \frac{\pi}{2}$ [162]. Note that discrete labelling schemes which identify cubic corners (Chapter 4) or orthogonal edges (Chapter 5) naturally inflate scenes without the use of a compliance function favouring orthogonality.

When constructing a compliance function expressing a preference for horizontalness or verticality, it is important to make the distinction between viewpoint coordinates (X, Y, Z) and world coordinates (u, v, w) . We usually draw objects as if they were placed on a horizontal plane and viewed from above at an angle $\alpha \approx \frac{\pi}{6}$. This implies the following transformation between viewpoint and world coordinates, assuming right-hand coordinate systems:

$$\begin{aligned} u &= X, \\ v &= Y \cos \alpha - Z \sin \alpha, \\ w &= Z \cos \alpha + Y \sin \alpha. \end{aligned}$$

Under orthographic projection, vertical lines project into vertical lines in the picture plane but suffer from foreshortening.

Varley et al. [166, 169] and Company et al. [26] dismiss the importance of SDA in favour of compliance functions based on planarity, parallel lines, cubic corners, symmetry, etc. However, the objects depicted in Figure 10.1 demonstrate that certain objects contain none or few orthogonality features and hence require some inflation-inducing compliance function such as SDA for inflation to occur.

Unlike methods which estimate the depths of vertices while minimizing some complex objective function, Liu et al. [106] estimate the free parameters of the planar faces while minimizing a very simple objective function (SDA). The result is a much smaller search space, the number of free parameters typically being an order of magnitude less than the number of vertices. Their experiments demonstrate that the 3D reconstruction of a planar-faced object from a wireframe projection is both more reliable and faster, even with a very simple objective function. The superstrictness of the equations (where a small error in the drawing would imply that there is no solution) is dealt with in an ad hoc way by adding extra free parameters based on the singular value decomposition.

The choice of the most appropriate objective function for the 3D reconstruction of objects with some planar and some curved surfaces is at present an open problem which will no doubt be the subject of research in the near future.

10.2 Hidden-Part Reconstruction

In this section, we consider line drawings in which hidden lines are not shown. An important limitation of such drawings is that they only represent the frontal geometry of the corresponding 3D scene. This begs the question as to whether it is possible to automatically reconstruct hidden object parts. Human beings are certainly capable of this. The drawings in Figures 3.23, 5.2 and 5.9 are quite complex, but we can completely reconstruct, with little or no ambiguity, the hidden parts of all objects depicted in these drawings. In a line drawing we have very incomplete information about curved surfaces since they are only visible in the drawing at edges (surface-normal discontinuity edges or extremal edges). The reconstruction of visible curved surfaces could therefore be considered as a special case of hidden-part reconstruction.

Consider a drawing of a 3D scene and its corresponding numerically labelled wireframe projection W . Hidden scene-edges fall into two categories:

- *Occluded* edges which lie behind another object (or part of the same object) and which have a numerical label m, n in W with $m > 0$ an even number;
- *Invisible* edges which face away from the viewpoint and which have a numerical label m, n in W where $m > 0$ is an odd number.

Lines with numerical depth label $m = 0$ in W may still be *missing* due, for example, to contrast failure if the drawing has been derived from an intensity image [38]. The system of Ding and Young [57] reconstructs both hidden and missing edges based on heuristic rules inspired by Gestalt principles including symmetry, good shape, simplicity and closure. Their system is designed for drawings of polyhedral objects with trihedral vertices and performs well on objects with 3D symmetry. It is incomplete, in the sense that it can fail to propose a 3D completion of a partially reconstructed object if none of its rules can be triggered.

The system of Cao et al. [18] recovers hidden edges of polyhedral objects with trihedral vertices and no through holes. By assuming that each hidden vertex is connected to at least one visible vertex, they can bound the number of hidden vertices. A complete search is then possible over all possible hidden structures (the graph of hidden vertices and edges). They select the most plausible hidden structure by topological symmetry: hidden faces are mapped injectively to visible faces having the same number of edges; they then prefer the hidden structure with the minimum number of unmapped hidden faces, ties being broken by a preference for a smaller number of hidden vertices. This topological reconstruction is then followed by a hill-climbing algorithm (Leclerc and Fischler's continuation method [99]) to determine the 3D coordinates of

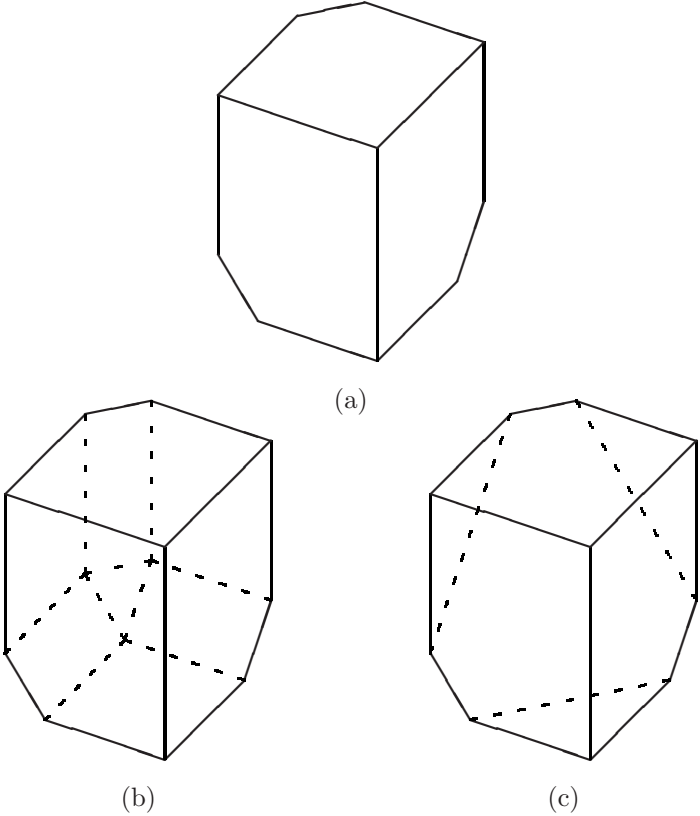


Figure 10.2: The line drawing in (a) has many different possible hidden-part reconstructions, two of which are shown in (b) and (c).

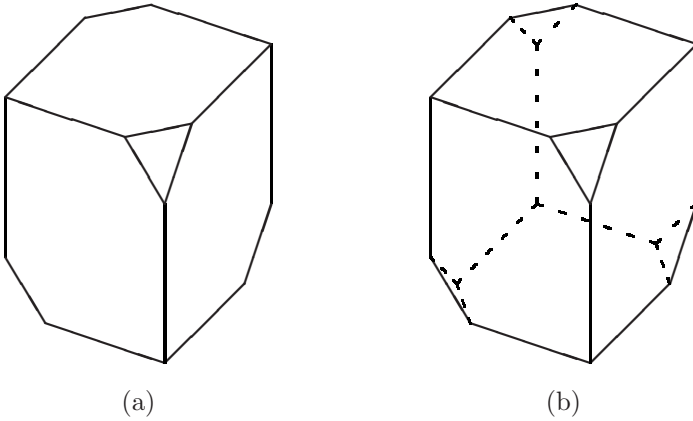


Figure 10.3: The line drawing in (a) has a hidden-part reconstruction, shown in (b), which has six planes of symmetry.

hidden vertices and the depth of visible vertices. The objective function is a weighted sum of $\Sigma P_i^2/A_i$, where P_i, A_i are the perimeter and the area of face i , Marill’s SDA (standard deviation of angles between adjacent edges) [111], and the sum of the squared distances of vertices from the planes of the faces in which they should lie. The first two criteria encourage inflation and a certain form of numerical symmetry, while the third criterion encourages planarity.

To illustrate the difficulty of hidden-part reconstruction, even in the simple case of a single polyhedral object, consider the drawing in Figure 10.2(a). Most people imagine the hidden part of the object to be as illustrated by the broken lines in Figure 10.2(b). This is despite the fact that the hidden part illustrated in Figure 10.2(c) is much simpler, in that it involves three fewer vertices, six fewer edges, three fewer faces, and furthermore all its vertices are trihedral. The explanation perhaps lies in the fact that the 3D shape in Figure 10.2(b) is closer to a cube than the shape in Figure 10.2(c). Now consider what happens if we shave off the nearest corner of the object, as shown in Figure 10.3(a). Most people now imagine the hidden part of the object to be as illustrated in Figure 10.3(b). This interpretation has six planes of symmetry, whereas completing the object as shown in Figure 10.2(b) or (c) would produce an object with only three planes of symmetry. These examples demonstrate two things: that Gestalt principles (such as symmetry, good shape and simplicity) are important but that assigning the appropriate importance to the different principles when they are in conflict with each other is a challenging open problem.

Occlusion can segment an object into two or more visible parts. Tse [161] argues that human volume completion is achieved through a combined approach using inter-relationships between reconstructed contours/surfaces, the “mergeability” of unbounded reconstructed volumes, world knowledge and the

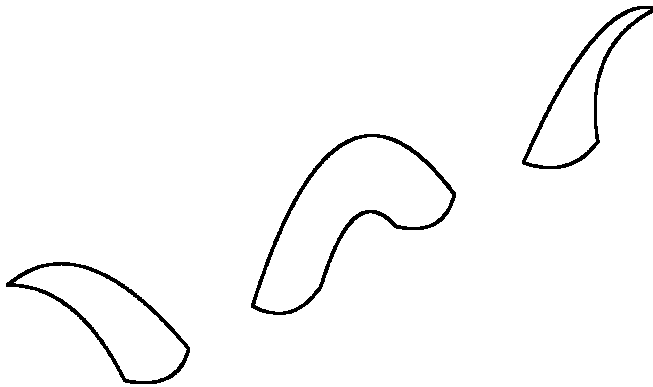


Figure 10.4: The ‘sea monster’ (adapted from [161]).

extrapolation of patterns into hidden parts of the scene. Breckon and Fisher [14] discuss the implications of applying these principles in diverse computer vision applications. A classic example of the importance of world knowledge is Tse’s sea monster [161], illustrated in Figure 10.4. We see this as a single object even though there is no visible occluding object and no obvious continuity between the three visible parts of the object.

The extreme case of use of world knowledge is when we have 3D models of all objects which may occur in the scene. Many diverse techniques have been used to recognize possibly partially-occluded objects, but all of them detect local features which are invariant under the set of rigid transformations that the object may undergo. Examples of such features are vertices, straight edge segments, zeros of curvature and discontinuities of curvature of 3D edges. (See Chapter 9 of [31] for an extensive review of techniques for partially-occluded object recognition.) The computational complexity of the interpretation of cluttered scenes containing only known objects has been studied in detail [35]: the constraint that objects must not intersect in 3D space is sufficient to render the problem of finding a single legal interpretation NP-hard (except in the case of orthographic projection of scenes in which objects do not form mutually occluding cycles).

Bibliography

- [1] Aarts, E. and Lenstra, J., *Local Search in Combinatorial Optimization*, Wiley, New York (1997).
- [2] Ablameyko, S. and Pridmore, T., *Machine Interpretation of Line Drawing Images: Technical Drawings, Maps and Diagrams*, Springer (2000).
- [3] Affane, M.-S. and Bennaceur, H., A weighted arc consistency technique for MAX-CSP, *Proc. ECAI'98*, Brighton (1998) pp. 209–213.
- [4] Agarwal, S.C. and Waggenspack, W.N. Jr., Decomposition method for extracting face topologies from wireframe models, *Comput.-Aided Des.*, 24(3) (1992) pp. 123–140.
- [5] Alevizos, P.D., A linear algorithm for labeling planar projections of polyhedra, *Proc. IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS'91*, Osaka (1991) pp. 595–601.
- [6] Anglès d'Auriac, J-Ch., Igloi, F., Preismann, M. and Sebö, A., Optimal cooperation and submodularity for computing Potts' partition functions with a large number of statistics, *J. Physics A Math. Gen.* 35 (2002), pp. 6973–6983.
- [7] Baird, L. and Wang, P., 3D object perception using gradient descent, *J. Math. Imag. and Vision* 5(2) (1995) pp. 111–117.
- [8] Bennaceur, H. and Osmani, A., Computing lower bounds for MAX-CSP problems, *Proc. 16th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-2003)*, P.W. Chung, C.J. Hinde and M. Ali (eds.), *LNAI 2718*, Springer (2003) pp. 614–624.
- [9] Bessière, C., Arc-consistency in dynamic constraint satisfaction problems, *Proc. AAAI'91*, Anaheim, CA, USA (1991) pp. 221–226.
- [10] Bessière, C. and Régin, J.-C., Refining the basic constraint propagation algorithm, *Proc. IJCAI'01*, Seattle (2001) pp. 309–315.

- [11] Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T. and Verfaillie, G., Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison, *Constraints* 4, (1999) pp. 199–240.
- [12] Bleux, J.-M. and Boudierlique, F. *Dessin industriel*, Nathan, Paris (1996).
- [13] Boros, E. and Hammer, R.L., Pseudo-Boolean optimization, *Discrete Appl. Math.* 123 (2002) pp. 155–225.
- [14] Breckon, T.P. and Fisher, R.B., Amodal volume completion: 3D visual completion, *Comput. Vision Image Understand.* 99 (2005) pp. 499–526.
- [15] Brown, E. and Wang, P., 3D object recovery from 2D images: a new approach, *SPIE Proc. Robot. Comput. Vision* 2904 (1996) pp. 138–145.
- [16] Bulatov, A.A., A dichotomy theorem for constraint satisfaction problems on a three-element set, *J. ACM* 53(1) (2006) pp. 66–120.
- [17] Burns, K.J., Mental models of line drawings, *Perception* 30 (2002) pp. 1249–1261.
- [18] Cao, L., Liu, J. and Tang, X., What the back of the object looks like: 3D reconstruction from line drawings without hidden lines, *IEEE Trans. Pattern Anal. Machine Intell.* 30(3) (2007) pp. 507–517.
- [19] Chvátal, V. *Linear Programming*, Freeman, New York, 1983.
- [20] Clowes, M.B., On seeing things, *Artif. Intell.* 2 (1) (1971) pp. 79–116.
- [21] Cohen, D., Cooper, M.C. and Jeavons, P., An algebraic characterisation of complexity for valued constraints, *Proc. CP'06, LNCS* 4204, Springer (2006) pp. 107–121.
- [22] Cohen, D., Cooper, M.C., Jeavons, P., Generalising submodularity and Horn clauses: tractable optimization problems defined by tournament pair multimorphisms, *Theor. Comput. Sci.* to appear (2008).
- [23] Cohen, D., Cooper, M.C., Jeavons, P. and Krokhin, A., A maximal tractable class of soft constraints, *J. Artif. Intell. Res.* 22 (2004) pp. 1–22.
- [24] Cohen, D., Cooper, M.C., Jeavons, P. and Krokhin, A. The complexity of soft constraint satisfaction, *Artif. Intell.* 170 (11) (2006) pp. 983–1016.
- [25] Cohen, D. and Jeavons, P., The Complexity of Constraint Languages in *Handbook of Constraint Programming*, Rossi, F., van Beek, P. and Walsh, T. (eds.) Elsevier Science, New York (2006) pp. 245–280.
- [26] P. Company, M. Contero, J. Conesa and A. Piquer, An optimisation-based reconstruction engine for 3D modelling by sketching, *Comput. Graph.* 28 (2004) pp. 955–979.

- [27] Company, P., Piquer, A., Contero, M. and Naya, F., A survey of geometrical reconstruction as a core technology to sketch-based modelling, *Comput. Graph.* 29 (2005) pp. 892–904.
- [28] Condoor, S. *Mechanical Design Modeling Using Pro/ENGINEERTM*, McGraw-Hill, New York (2002).
- [29] Conesa Pastor, J., Company Calleja, P., Gomis Marti, J.M., Initial modelling strategies for geometrical reconstruction - optimisation-based approaches, *Proc. 11th Int. Conf. on Design Tools and Methods in Industrial Engineering* (1999) pp. 161–171.
- [30] Connoly, D., General purpose simulated annealing, *J. Oper. Res. Soc.* 43 (5) (1992) pp. 495–505.
- [31] Cooper, M.C. *Visual Occlusion and the Interpretation of Ambiguous Pictures*, Ellis Horwood, Chichester, U.K. (1992).
- [32] Cooper, M.C., Interpretation of line drawings of complex objects, *Image Vision Comput.* 11 (2) (1993) pp. 82–90.
- [33] Cooper, M.C., Fundamental properties of neighbourhood substitution in constraint satisfaction problems, *Artif. Intell.* 90 (1997) pp. 1–24.
- [34] Cooper, M.C., Interpreting line drawings of curved objects with tangential edges and surfaces, *Image Vision Comput.* 15 (1997) pp. 263–276.
- [35] Cooper, M.C., The tractability of segmentation and scene analysis, *Int. J. Comput. Vision* 30(1) (1998) pp. 27–42.
- [36] Cooper, M.C., Linear-time algorithms for testing the realisability of line drawings of curved objects, *Artif. Intell.* 108 (1999) pp. 31–67.
- [37] Cooper, M.C., Linear constraints for the interpretation of line drawings of curved objects, *Artif. Intell.* 119 (2000) pp. 235–258.
- [38] Cooper, M.C., The interpretation of line drawings with contrast failure and shadows, *Int. J. Comput. Vision* 43 (2) (2001) pp. 75–97.
- [39] Cooper, M.C., Reduction operations in fuzzy and valued constraint satisfaction, *Fuzzy Sets Syst.* 134 (2003) pp. 311–342.
- [40] Cooper, M.C., Cyclic consistency: a local reduction operation for binary valued constraints, *Artif. Intell.* 155(1–2) (2004) pp. 69–92.
- [41] Cooper, M.C., High-order consistency in valued constraint satisfaction, *Constraints* 10 (2005) pp. 283–305.
- [42] Cooper, M.C., Wireframe projections: physical realisability of curved objects and unambiguous reconstruction of simple polyhedra, *Int. J. Comput. Vision* 64 (1) (2005) pp. 69–88.

- [43] Cooper, M.C., Constraints between distant lines in the labelling of line drawings of polyhedral scenes, *Int. J. of Computer Vision* 73(2) (2007) pp. 195–212.
- [44] Cooper, M.C., Minimization of locally-defined submodular functions by optimal soft arc consistency, *Constraints*, to appear (2008).
- [45] Cooper, M.C., A rich discrete labeling scheme for line drawings of curved objects, *IEEE Trans. Pattern Anal. Machine Intell.*, to appear (2008).
- [46] Cooper, M.C., Cohen, D.A. and Jeavons, P.G., Characterising tractable constraints, *Artif. Intell.* 65 (1994) pp. 347–361.
- [47] Cooper, M.C., Cussat-Blanc, S., de Roquemaurel, M. and Régnier, P., Soft arc consistency applied to optimal planning, *Proc. CP'06*, Nantes, LNCS 4204, Springer (2006) pp. 680–684.
- [48] Cooper, M.C., de Givry, S. and Schiex, T., Optimal soft arc consistency, *Proc. IJCAI'07*, Hyderabad (2007) pp. 68–73.
- [49] Cooper, M.C., de Givry, S., Sanchez, M., Schiex, T. and Zytnicki, M., Virtual arc consistency for valued CSP, to appear in *Proc. AAAI-08*, Chicago (2008).
- [50] Cooper, M.C. and Schiex, T., Arc consistency for soft constraints, *Artif. Intell.* 154(1-2) (2004) pp. 199–227.
- [51] Costa Sousa, M. and Prusinkiewicz, P., A few good lines: suggestive drawing of 3D models, *Comput. Graph. Forum* 22(3) (2003) pp. 381–390.
- [52] Cunningham, W.H., Minimum cuts, modular functions, and matroid polyhedra, *Networks* 15(2) (1985) pp. 205–215.
- [53] de Givry S., Heras, F., Larrosa, J., Rollon, E. and Schiex, T., The SoftCSP and Max-SAT benchmarks and algorithm website, <http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/softcsp>.
- [54] de Givry S., Heras, F., Zytnicki, M. and Larrosa, J., Existential arc consistency: getting closer to full arc consistency in weighted CSPs, *Proc. IJCAI-05*, Edinburgh, Scotland (2005) pp. 84–89.
- [55] Dechter, R. *Constraint Processing*, Morgan Kaufmann, San Mateo, CA (2003).
- [56] Dendris, N.D., Kalafatis, I.A. and Kirousis, L.M., An efficient parallel algorithm for geometrically characterising drawings of a class of 3-D objects, *J. Math. Imag. Vision* 4 (1994) pp. 375–387.

- [57] Ding, Y. and Young, T.Y., Complete shape from imperfect contour: a rule-based approach, *Comput. Vision Image Understand.* 70(2) (1998) pp. 197–211.
- [58] Dowling, W. and Gallier, J., Linear-time algorithms for testing the satisfiability of propositional Horn formulae, *J. Logic Programm.* 1(3) (1984) pp. 267–284.
- [59] Draper, S.W., The use of gradient and dual space in line-drawing interpretation, *Artif. Intell.* 17 (1981) pp. 461–508.
- [60] Elber, G., Line illustrations \in computer graphics, *Visual Comput.* 11 (1995) pp. 290–296.
- [61] Ernst, B., *Adventures with Impossible Figures*, Tarquin, Stradbroke, Norfolk (1986).
- [62] Fargier, H. and Lang, J., Uncertainty in constraint satisfaction problems: a probabilistic approach, in *Proc. ECSQARU*, Springer, LNCS 747 (1993) pp. 97–104.
- [63] Fargier, H., Lang, J. and Schiex, T., Selecting preferred solutions in Fuzzy Constraint Satisfaction Problems, *Proc. 1st European Congress on Fuzzy and Intelligent Technologies*, Aachen (1993) pp. 1128–1134.
- [64] Faugeras, O., *Three-Dimensional Computer Vision*, MIT Press, Cambridge, MA (1993).
- [65] Feder, T. and Vardi, M.Y., The computational structure of monotone monadic SNP and constraint satisfaction: a study through datalog and group theory, *SIAM J. Comput.* 28(1), (1998) pp. 57–104.
- [66] Freuder, E.C., Eliminating interchangeable values in constraint satisfaction problems in *Proc. AAAI-91*, Anaheim, CA (1991) pp. 227–233.
- [67] Fujishige, S., *Submodular Functions and Optimisation*, 2nd edn., Annals of Discrete Mathematics 58, Elsevier, Amsterdam, 2005.
- [68] Fujishige, S. and Patkar, S.B., Realization of set functions as cut functions of graphs and hypergraphs, *Discrete Math.* 226 (2001) pp. 199–210.
- [69] Goldberg, A. and Tarjan, R.E., A new approach to the maximum flow problem, *J. ACM* 35 (1988) pp. 921–940.
- [70] Grimstead, I.J. and Martin, R.R., Incremental line labelling for sketch input of solid models, *Comput. Graph. Forum* 15(2) (1996) pp. 155–166.
- [71] Grötschel, M., Lovász, L. and Schrijver, A., The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica* 1 (1981) pp. 169–198; corrigendum: *Combinatorica* 4 (1984) pp. 291–295.

- [72] Hammer, P.L., Hansen, P. and Simeone, B., Roof duality, complementation and persistency in quadratic 0-1 optimization, *Math. Programm.* 28 (1984) pp. 121–155.
- [73] Heyden, A., On the consistency of line-drawings, obtained by projections of piecewise planar objects, *J. Math. Imag. Vision* 6 (1996) pp. 393–412.
- [74] Horn, B.K.P., *Robot Vision*, MIT Press, Cambridge, MA (1986).
- [75] Huffman, D.A., Impossible objects as nonsense sentences in *Machine Intelligence* 6, Meltzer, B. and Michie, D. (eds.) Edinburgh University Press (1971) pp. 295–323.
- [76] Huffman, D.A., A duality concept for the analysis of polyhedral scenes, in *Machine Intelligence* 8, Elcock E.W. and Michie, D. (eds.) Ellis Horwood, Chichester (1977) pp. 475–492.
- [77] Huffman, D.A., Realizable configurations of lines in pictures of polyhedra, in *Machine Intelligence* 8, Elcock, E.W. and Michie, D. (eds.) Ellis Horwood, Chichester (1977) pp. 493–509.
- [78] Imai, H., On combinatorial structures of line drawings of polyhedra, *Discrete Appl. Math.* 10 (1985) pp. 79–92.
- [79] Iwata, S., A fully combinatorial algorithm for submodular function minimization, *J. Combinator. Theory Ser. B* 84(2) (2002) pp. 203–212.
- [80] Iwata, S., A faster scaling algorithm for minimizing submodular functions, *SIAM J. Comput.* 32(4) (2003) pp. 833–840.
- [81] Iwata, S., Fleischer, L. and Fujishige, S., A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions, *J. ACM* 48(4), (2001), pp. 761–777.
- [82] Jain, P.K., Extraction of compound volumetric features from a three-dimensional wire frame model, *Proc. Institute of Mechanical Engineers Part B J. Eng. Manufact.* 213(6) (1999) pp. 597–613.
- [83] Jeavons, P.G., On the algebraic structure of combinatorial problems, *Theor. Comput. Sci.* 200 (1998) pp. 185–204.
- [84] Jeavons, P. and Cooper, M.C., Tractable constraints on ordered domains, *Artif. Intell.* 79 (1995) pp. 327–339.
- [85] Jeavons, P., Cohen, D. and Cooper M.C., Constraints, consistency and closure, *Artif. Intell.* 101 (1998) pp. 251–265.
- [86] Jeavons, P.G., Cohen D.A. and Gyssens, M., Closure properties of constraints, *J. ACM* 44 (1997) pp. 527–548.

- [87] Kanade, T., Recovery of the three-dimensional shape of an object from a single view, *Artif. Intell.* 17 (1981) pp. 409–460.
- [88] Kanade, T., From a real chair to a negative chair, *Artif. Intell.* 59 (1993) pp. 95–101.
- [89] Kanatani, K., The constraints on images of rectangular polyhedra, *IEEE Trans. Pattern Anal. Machine Intell.* 8(4) (1986) pp. 456–463.
- [90] Kirousis, L.M., Effectively labeling planar projections of polyhedra, *IEEE Trans. Pattern Anal. Machine Intell.* 12(2) (1990) pp. 123–130.
- [91] Kirousis, L.M., Fast parallel constraint satisfaction, *Artif. Intell.* 64 (1993) pp. 147–160.
- [92] Kirousis, L.M. and Papadimitriou, C.H., The complexity of recognizing polyhedral scenes, *J. Comput. Syst. Sci.* 37 (1) (1988) pp. 14–38.
- [93] Koenderink, J.J., *Solid Shape*, MIT Press, Cambridge, MA (1990).
- [94] Koenderink, J.J. and van Doorn, A.J., The shape of smooth objects and the way contours end, *Perception* 11 (1982) pp. 129–137.
- [95] Kuo, M.H., A Systematic Approach Towards Reconstructing 3D Curved Models from Multiple 2D Views in *Graphics Recognition: Algorithms and systems; 2nd International Workshop, GREC'97*, Tombre, K. and Chhabra, A.K. (eds.), LNCS 1389, Springer (1998) pp. 265–279.
- [96] Larrosa, J., Node and arc consistency in weighted CSP, *Proc. AAAI'02*, Edmonton, Alberta (2002), pp. 48–53.
- [97] Larrosa, J. and Schiex, T., In the quest of the best form of local consistency for Weighted CSP, *Proc. IJCAI*, Acapulco (2003), pp. 239–244.
- [98] Larrosa, J. and Schiex, T., Solving weighted CSP by maintaining arc consistency, *Artif. Intell.* 159 (2004) pp. 1–26.
- [99] Leclerc, Y.G. and Fischler, M.A., An optimization-based approach to the interpretation of single line drawings as 3D wire frames, *Int. J. Comput. Vision* 9(2) (1992) pp. 113–136.
- [100] Li, C.M., Many, F. and Planes J., Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers, *Principles and Practice of Constraint Programming – CP 2005*, Sitges, Spain (2005) LNCS 3709, pp. 403–414.
- [101] Li, H., n D polyhedral scene reconstruction from single 2D line drawing by local propagation, *ADG 2004*, Gainesville, FL, Hong, H. and Wang, D. (eds.), LNCS 3763, Springer (2006) pp. 169–197.

- [102] Lichtenstein, D., Planar formulae and their uses, *SIAM J. Comput.* 11 (1982) pp. 329–343.
- [103] Lipson, H. and Shpitalni, M., Optimisation-based reconstruction of a 3d object from a single freehand line drawing, *Comput.-Aided Des.* 28 (8) (1996) pp. 651–663.
- [104] Lipson, H. and Shpitalni, M., Conceptual design and analysis by sketching, *Artif. Intell. Eng. Des., Anal. Manuf.* 14 (2000) pp. 391–401.
- [105] Liu, J., Lee, Y.T., Cham, W.-K., Identifying faces in a 2D line drawing representing a manifold object, *IEEE Trans. Pattern Anal. Machine Intell.* 24 (12) (2002) pp. 1579–1593.
- [106] Liu, J., Cao, L., Li, Z. and Tang, X., Plane-based optimization for 3D object reconstruction from single line drawings, *IEEE Trans. Pattern Anal. Machine Intell.* 30(2) (2008) pp. 315–327.
- [107] Mackworth, A.K., Interpreting pictures of polyhedral scenes, *Artif. Intell.* 4 (1973) pp. 121–137.
- [108] Mackworth, A.K., Consistency in networks of relations, *Artif. Intell.* 8 pp. 99–118.
- [109] Malik, J., Interpreting line drawings of curved objects, *Int. J. Comput. Vision* 1 (1987) pp. 73–103.
- [110] Malik, J. and Maydan, D., Recovering three-dimensional shape from a single image of curved objects, *IEEE Trans. Pattern Anal. Machine Intell.* 11(6) (1989) pp. 555–566.
- [111] Marill, T., Emulating the human interpretation of line drawings as three-dimensional objects, *Int. J. Comput. Vision* 6 (2) (1991) pp. 147–161.
- [112] Markowsky, G. and Wesley, M.A., Fleshing out wire frames, *IBM J. Res. Develop.*, 24(5) (1980) pp. 582–597.
- [113] Marsolek, C.J. and Burgund, E.D., Initial storage of unfamiliar objects: Examining memory stores with signal detection and analysis, *Acta Psychologica* 119(1) (2005) pp. 81–106.
- [114] Melhorn, K., *Graph Algorithms and NP-Completeness*, EACTS, Springer, Berlin (1974).
- [115] Meseguer, P., Rossi, F. and Schiex, T., Soft Constraints, in *Handbook of Constraint Programming*, Rossi, F., van Beek, P. and Walsh, T. (eds.) Elsevier, Amsterdam (2006) pp. 281–328.
- [116] Mohr, R. and Masini, G., Good old discrete relaxation, *Proc. 8th European Conference on Artificial Intelligence – ECAI’88*, Munich, Kodratoff, Y. (ed.) Pitman, London (1988) pp. 651–656.

- [117] Mitiche, A. and Habelrih, G., Interpretation of straight line correspondences using angular relations, *Pattern Recog.* 22(3) (1989) pp. 299–308.
- [118] Mortenson, M. *Geometric Modelling*, 2nd edn, Wiley, New York (1997).
- [119] Nalwa, V.S., Line drawing interpretation: a mathematical framework, *Int. J. Comput. Vision* 2(2) (1988) pp. 103–124.
- [120] Nalwa, V.S., Line-drawing interpretation: bilateral symmetry, *IEEE Trans. Pattern Anal. Machine Intell.* 11(10) (1989) pp. 1117–1120.
- [121] Narayanan, H., *Submodular Functions and Electrical Networks*, North-Holland, Amsterdam (1997).
- [122] Oh, B. and Kim, C., Self-correctional 3D shape reconstruction from a single freehand line drawing, *ICCSA 2003*, Montreal, Kumar, V. et al. (eds.) *LNCS* 2669, Springer (2003) pp. 528–538.
- [123] Orlin, J.B. A faster strongly polynomial time algorithm for submodular function minimization, *IPCO 2007*, *LNCS* 4513, pp. 240–251.
- [124] Ortiz, S. Jr., 3D search starts to take shape, *Computer*, August (2004) pp. 24–26.
- [125] Parodi, P., The complexity of understanding line drawings of origami scenes, *Int. J. Comput. Vision* 18(2) (1996) pp. 139–170.
- [126] Parodi, P., Lancewicki, R., Vijh, A. and Tsotsos, J.K., Empirically-derived estimates of the complexity of labelling line drawings of polyhedral scenes, *Artif. Intell.* 105 (1-2) (1998) pp. 47–75.
- [127] Parodi, P. and Torre, V., On the complexity of labelling perspective projections of polyhedral scenes, *Artif. Intell.* 70 (1994) pp. 239–276.
- [128] Penrose, L.S. and Penrose, R., Impossible objects: a special type of visual illusion, *Br. J. Psychol.* 49 (1958) pp. 31–33.
- [129] Penrose, R., On the cohomology of impossible figures, in *The Visual Mind: Art and Mathematics*, Emmer, M. (ed.) Leonardo Books, MIT Press, Cambridge, MA (1993) pp. 27–29.
- [130] Perkins, D.N., Visual discrimination between rectangular and nonrectangular parallelepipeds, *Percept. Psychophys.* 12 (5) (1972) pp. 396–400.
- [131] Petitjean, S., The enumerative geometry of projective algebraic surfaces and the complexity of aspect graphs, *Int. J. Comput. Vision* 19 (3) (1996) pp. 261–287.
- [132] Piquer Vicent, A., Company Calleja, P. and Martin, R.R., Skewed mirror symmetry in the 3D reconstruction of polyhedral models, *J. WSCG* 11 (3) (2003) pp. 504–511.

- [133] Régin, J., A filtering algorithm for constraints of difference in CSPs, *Proc. 12th Natl. Conf. on Artificial Intelligence (AAAI-94)*, AAAI Press, Seattle (1994) pp. 362–367.
- [134] Rodgers, N., *Incredible Optical Illusions*, Simon and Schuster, New York (1998).
- [135] Ros, L. and Thomas, F., Overcoming superstrictness in line drawing interpretation, *IEEE Trans. Pattern Anal. Machine Intell.* 24 (4) (2002) pp. 456–466.
- [136] Rosenfeld, A., Hummel, R.A. and Zucker, S.W., Scene labeling by relaxation operations, *IEEE Trans. Syst., Man Cybern.* 6(6) (1976) pp. 420–433.
- [137] Russel, S. and Norvig, P., *Artificial Intelligence: A Modern Approach*, 2nd edn., Prentice Hall, New Jersey (2003).
- [138] Schaefer, T.J., The complexity of satisfiability problems, *Proc. of the 10th ACM Symposium on the Theory of Computing, STOC'78*, San Diego (1978) pp. 216–226.
- [139] Schiex T., Arc consistency for soft constraints, in *Principles and Practice of Constraint Programming – CP 2000*, Singapore (2000) LNCS 1894, Springer, pp. 411–424.
- [140] Schiex, T., Fargier, H. and Verfaillie, G., Valued constraint satisfaction problems: hard and easy problems, *Proc. 14th IJCAI*, Montreal (1995) pp. 631–637.
- [141] Schlesinger, D., Exact solution of permuted submodular MinSum problems, *Proc. 6th Int. Conf. Energy Minimization Methods in Computer Vision and Pattern Recognition*, Ezhou, Hubei, China (2007) pp. 28–38.
- [142] Schlesinger, M., Syntactic analysis of two-dimensional visual signals in noisy conditions, *Kibernetika* 4 (1976) pp. 113–130 (in Russian).
- [143] Schlesinger, M., *Mathematical Tools of Image Processing*, Naukova Dumka, Kiev (1989) (in Russian).
- [144] Seckel, A., *The Great Book of Optical Illusions*, Firefly Books, Toronto (2002).
- [145] Shapira, R. and Freeman, H., Computer description of bodies bounded by quadric surfaces from a set of imperfect projections, *IEEE Trans. Comput.* 27(9) (1978) pp. 841–854.
- [146] Shapira, R. and Freeman, H., The cyclic order property of vertices as an aid in scene analysis, *Commun. ACM* 22(6) (1979) pp. 368–375.

- [147] Shpitalni, M. and Lipson, H., Identification of faces in a 2D line drawing projection of a wireframe object, *IEEE Trans. Pattern Anal. Machine Intell.* 18 (10) (1996) pp. 1000–1012.
- [148] Schrijver, A., A combinatorial algorithm minimizing submodular functions in strongly polynomial time, *J. Combinator. Theory Ser. B* 80 (2000) pp. 346–355.
- [149] Sommer, W., *Jetzt lerne ich CAD*, Markt+Technik (2002).
- [150] Straforini, M., Coelha, C. and Campani, M., Extraction of vanishing points from images of indoor and outdoor scenes, *Image Vision Comput.* 11(2) (1993) pp. 91–99.
- [151] Sugihara, K., Picture language for skeletal polyhedra *Comput. Graph. Image Process.* 8 (1978) pp. 382–405.
- [152] Sugihara, K., Mathematical structures of line drawings of polyhedrons. Towards man-machine communication by means of line drawings, *IEEE Trans. Pattern Anal. Machine Intell.* 4 (1982) pp. 458–469.
- [153] Sugihara, K., Classification of impossible objects, *Perception* 11 (1982) pp. 65–74.
- [154] Sugihara, K., A necessary and sufficient condition for a picture to represent a polyhedral scene, *IEEE Trans. Pattern Anal. Machine Intell.* 6 (5) (1984) pp. 578–586.
- [155] Sugihara, K., *Machine Interpretation of Line Drawings*, MIT Press, Cambridge, MA (1986) (freely available on Kokichi Sugihara’s website: <http://www.simplex.t.u-tokyo.ac.jp/sugihara>).
- [156] Sugihara, K., Three-dimensional realization of anomalous pictures - an application of picture interpretation theory to toy design, *Pattern Recog.* 30(7) (1997) pp. 1061–1067.
- [157] Syeda-Mahmood, T., Indexing of technical line drawing databases *IEEE Trans. Pattern Anal. Machine Intell.* 21(8) (1999) pp. 737–751.
- [158] Tai, A., Kittler, J., Petrou, M. and Winderatt, T., Vanishing point detection, *Image Vision Comput.* 11(4) (1993) pp. 240–245.
- [159] Topkis, D., *Supermodularity and Complementarity*, Princeton University Press (1998).
- [160] Tsang, E., *Foundations of Constraint Satisfaction*, Academic Press, New York (1993).
- [161] Tse, P.U., Volume completion, *Cognit. Psychol.* 39 (1999) pp. 37–68.

- [162] Turner, A., Chapman, D. and Penn, A., Sketching space, *Comput. Graph.* 24 (2000) pp. 869–879.
- [163] van Hoes, W.-J. and Katriel, I., Global constraints, Chapter 6 of *Handbook of Constraint Programming*, Rossi, F., van Beek, P. and Walsh, T. (eds.) Elsevier Science, New York (2006).
- [164] Varley, P.A.C. and Martin, R.R., A system for constructing boundary representation solid models from a two-dimensional sketch, *Proc. Geometric Modelling and Processing 2000*, Martin, R. and Wang, W. (eds.) IEEE Computer Society Press (2000) pp. 13–32.
- [165] Varley, P.A.C. and Martin, R.R., The junction catalogue for labelling line drawings of polyhedra with tetrahedral vertices, *Int. J. Shape Modell.* 7 (1) (2001) pp. 23–44.
- [166] Varley, P.A.C. and Martin, R.R., Estimating Depth from Line Drawings, *Proc. 7th ACM Symp. on Solid Modelling and Applications* (2002) pp. 180–191.
- [167] Varley, P.A.C. and Martin, R.R., Deterministic and probabilistic approaches to labelling line drawings of engineering objects, *Int. J. Shape Modell.*, 9 (1) (2003) pp. 79–99.
- [168] Varley, P.A.C., Suzuki, H. and Martin, R.R., Interpreting line drawings of objects with K -vertices, *Proc. Geometric Modelling and Processing 04*, Beijing, Hu, S.-M. and Pottmann, H. (eds.) IEEE Computer Society (2004) pp. 249–258.
- [169] Varley, P.A.C., Martin, R.R. and Suzuki, H., Frontal geometry from sketches of engineering objects: is line labelling necessary, *Comput.-Aided Des.* 37 (2005) pp. 1285–1307.
- [170] Vosniakos, G., Conversion of wireframe to ACIS solid models for $2\frac{1}{2}$ -D engineering components, *Int. J. Adv. Manufactur. Technol.* 14(3) (1997) pp. 199–209.
- [171] Vosniakos, G., An intelligent software system for the automatic generation of NC programs from wireframe models of $2\frac{1}{2}$ D mechanical parts, *Comput. Integr. Manufactur. Syst.* 11(1/2) (1998) pp. 53–65.
- [172] Walker, P., Dixon, S. and Smith, D., Associating object names with descriptions of shape that distinguish possible from impossible objects, *Visual Cognit.* 7(5) (2000) pp. 597–627.
- [173] Waltz, D., Understanding line drawings of scenes with shadows, in *The Psychology of Computer Vision*, Winston, P.H. (ed.) McGraw-Hill, New York (1975) pp. 19–91.

- [174] Werner, T., A Linear Programming Approach to Max-sum Problem: A Review, *IEEE Trans. Pattern Anal. Machine Intell.* 29(7) (2007) pp. 1165–1179.
- [175] Wesley, M.A. and Markowsky, G., Fleshing out projections *IBM J. Res. Develop.* 25(6) (1981) pp. 934–954.
- [176] Whiteley, W., From a line drawing to a polyhedron, *J. Math. Psychol.* 31 (1987) pp. 441–448.
- [177] Williams, L.R., Topological reconstruction of a smooth manifold-solid from its occluding contour, *Int. J. Comput. Vision* 23(1) (1997) pp. 93–108.

Index

- 0/1/all constraints, 217
- 2Reg constraint, 38
- 2SAT, 217, 227
- 2-tangent junction, 172
- 3D reconstruction, 231
- 3-consistency, 203
- 3-cyclic consistency, 197
- 3-tangent junction, 80, 111, 130
- 4-tangent junction, 171

- AC2001, 188
- addition-with-ceiling operator ($+_m$), 193
- affine constraints, 220
- aggregation operator (\oplus), 192
- AllDiff constraint, 188
- ambiguous figure, 89, 231
- ambiguous wireframe projection, 141, 153
- angle parity, 151
- anti-Horn clause, 220
- apex of a cone, 179, 226
- arc consistency, 187
- arithmetical constraints, 219

- ‘Belvedere’ (by M.C. Escher), 8, 102, 224
- best block, 212
- Bool(P), 203
- branch and bound, 191, 193
- B-rep model, 143

- ‘c’ label, 84
- C junction, 80, 101, 172, 224
- C^3 surfaces and edges, 79, 127, 179
- CAD/CAM, 125
- catalogue of junction labelings, 23, 46, 59-60, 134-136

- Col1Reg constraint, 40
- Col2Reg constraint, 40
- collinear lines/points, 229
- collinearity constraint 105, 117
- common-surface constraints, 146
- complexity, 155
- computer-enhanced perception, 55
- computer graphics, 72, 125
- concave-edge constraints 109
- concave line label, 21, 128
- consistency, 183
- constraint satisfaction problem, 183
- continuous optimization, 76
- contrast failure, 227
- convex-edge constraints, 109
- convex line label, 21, 128
- coplanarity constraints 112, 117, 127, 147-148
- cost, 192, 194
- crack line 52, 107, 110
- crisp constraints, 192
- CSP, 183
- cubic corner, 12, 55, 94, 117-118, 232
- curvature-L junction 80, 111, 130, 171
- curved objects 79
- cut-set rule, 41, 155
- cyclic consistency, 198
- cyclic path, 43
- cyclic-path constraint, 41

- depth information (displaying of), 72
- depth label (relative), 57, 180
- depth parity, 151
- depth reversal, 141

- directional arc consistency (DAC), 197
- discontinuity of curvature, 80
- discontinuity of surface curvature, 227
- discrete valuation structure, 193
- difference operator (\ominus), 193
- dual space 41, 97, 155

- EAC, 197
- EDAC, 197
- equiangular vertex, 73
- equivalence-preserving transformation (EPT), 195
- Extend**, 196
- equivalent VCSPs, 195
- existential arc consistency (EAC), 197
- existential directional arc consistency (EDAC), 197
- extended trihedral vertex, 21, 60
- extension of an assignment, 195
- extremal edge, 79, 128, 171
- extremal line, 128

- face, 142
- face boundary 142
- face circuit, 143, 176
- face-circuit fragment, 176
- fair valuation structure, 193
- faithful encoding, 51
- FDAC, 197
- finitely bounded VCSP, 202
- focal length, 102
- fractional weights, 194
- frequency assignment, 198, 200
- frontal geometry, 231
- frontier of a region, 137
- full directional arc consistency (FDAC), 197
- fuzzy CSP, 193

- general relative position, 127
- general viewpoint assumption (GVA), 6, **26**, 85, 88
- generalized arc consistency, 188, 197

- generic label (∇), 26, 110
- genetic algorithms, 231
- Gestalt principles, 57, 233, 235
- gradient direction, 98, 117, 173
- gradient-direction constraints, 88, 174
- gradient-direction propagation constraint, 176
- gradient-direction/semantic-label constraints, 90, 175
- gradient space, 97, 173

- hidden lines, 233
- hidden-part reconstruction, 233
- hidden-surface coplanarity constraints, 114
- hidden surfaces, 71
- higher-order consistency, 203
- hill climbing, 231
- holes, 130, 132, 137, 141-142, 152-153, 169
- horizontalness, 232
- Horn clause, 185, 219
- HORNSAT, 185
- Huffman–Clowes catalogue, 21-23
- human vision, 56

- image enhancement, 73
- impossible closed curve class, 7
- impossible fork, 8
- impossible wireframe projection, 137-138, 161
- inflation, 75, 232
- in-scope irreducibility, 202
- integral VCSP, 210
- intersection constraint, 105
- invisible line, 233

- K junction, 37, 167
- kiss, 171

- L junction, 23, 37, 172
- $L_1(+)$, $L_1(-)$, $L_2(+)$, $L_2(-)$ junctions, 45
- L-chain constraint, 41
- label transition (see also C junction), 101, 148, 173, 224

- legal labeling of a wireframe projection, 138
- leximin, 193
- line label, 128
- line thickness, 72
- linear constraints, 102
- linear features, 229
- local search, 231
- locally planar face, 87
- locally planar surface, 84, 173, 176

- majority polymorphism, 218
- manifold object, 79
- man-made objects, 83
- matter-space ambiguity, 141, 143
- max-closed constraints, 219, 226
- MAX-CSP, 193, 195
- maximal matching, 188
- maximally generically reconstructible substructure, 120
- minimum (weighted) cut, 221, 223
- missing line, 233
- Möbius strip, 140, 145
- model-based vision, 236
- MSDA, 76, 94, 179, 232
- MSDL, 76, 94
- Multi junction, 37, 166
- multimorphism, 222, 229

- necessary and sufficient conditions for realizability, 125, 138-141
- Necker cube, 141
- neighbourhood substitution, 189
- node consistency, 197
- non-manifold scenes, 52
- NP-completeness, 121, 155, 229
- null intersection, 160
- nullary constraint (c_\emptyset), 195
- numerical depth labels, 126-127
- numerical optimization, 76
- numerical symmetry, 235

- oblique rectangular corner, 73
- occluded line, 233
- occluding line label, 21, 128
- optimal soft arc consistency, 198
- origami world, 52, 229
- orthogonal edge, 82, 87, 89-91, 117, 174, 232
- orthographic projection, 97, 117, 152
- OSAC, 198
- outer-boundary constraint, 22, 137, 227

- 'p' label, 84, 112, 114, 175-176
- Par2-2, Par3-2, Par3-3 constraints, 45
- ParCon constraint, 26
- parallel curve segments, 95
- parallel lines, 229
- parallel lines constraint, 26, 117, 127
- parallel junctions, 45
- parallel path, 27-28
- parity constraint, 132
- ParOcc constraint, 26
- partially-visible object recognition, 236
- path in a wireframe projection, 159
- Peak junction, 37, 166
- penalty, 192
- Penrose triangle, 7, 102, 122, 149-150, 224
- Perkins's rules, 59, 74-75, 94, 175
- perspective projection, 66, 98, 102
- phantom edge, 79
- phantom junction, 224
- Phi junction, 23, 37, 167
- planar 3SAT, 228
- planarity labels, see 'p' label
- planarity constraints, 86
- planning, 198
- polyhedral junction constraint, 33
- polymorphism, 218, 229
- pottery world, 227
- Project**, 196

- quadratic posiform, 203
- quadratic pseudo-boolean function optimization, 203

- ramp line, 110

- rational valuation structure, 193
- realizability, 125, 138-141
- reconstruction, 231
- reduction operations, 183
- reflection line, 107, 110
- region constraint, 137
- regular solid, 127
- relation, 184
- rich labels, 79, 173
- right angles, 232

- SAC transformation, 120
- SAT, 185, 188, 221
- scope, 184
- sea monster, 236
- search, 183
- semantic labels, 8, 127
- semi-fold point, 80
- semi-ring CSP, 192
- shadow lines, 107, 110, 229
- sidedness reasoning, 42
- signature of a junction, 163
- similar curve segments, 95
- simple junction, 32
- simple polyhedral wireframe projection, 150
- simple vertex, 150
- simplification of combinatorial problems, 183
- simulated annealing, 231
- size/distance ambiguity, 120
- sketch, 56, 179, 192
- smooth edge, 171
- snowflake junction, 23, 132
- soft arc consistency, 192
- soft arc consistency transformation, 200
- soft consistency, 183
- soft constraint, 25, 50
- soft constraint satisfaction, see VCSP
- soft neighbourhood substitution, 212
- straight edge formation assumption, 79, 86, 147
- strictly monotone parallel path, 28
- strictly monotonic operator, 193
- strictly positive cyclic path, 43
- strictly positive intersection, 160
- strictly positive path in a wireframe projection, 159
- submodular function, 222
- submodular function minimization, 222
- subproblem of a VCSP, 195
- substitutable, 189
- substitution operations, 183
- superstrictness, 43, 71, 120, 162, 168, 232
- surface mark, 110
- surface normal discontinuity edge, 128
- symmetric vertex with dihedral right angle, 73
- symmetric vertex with right angle, 73
- symmetry, 179
- syzygy, 121

- T junction, 23, 37, 80, 172
- T= junction, 107, 116
- T> junction, 107, 114
- T-junction constraint, 106
- T-junction label, 107
- tabu search, 231
- tangential edges and surfaces, 114, 116, 169, 226
- tangential intersection curve (TIC), 169
- terminal junction, 80, 171
- tetrahedral vertex, 37, 163, 227
- TIC, 169
- topologically equivalent labelings, 169
- tournament operation, 224
- tournament-pair multimorphism, 224
- tractability, 217
- tractable constraint class, 220
- trihedral vertex, 21, 134-136

- unambiguous picture, 150
- UnaryProject**, 196
- unit propagation, 185, 188

- VAC, 203

- VAC decomposition, 209
- valuation, 194
- valuation structure, 192
- valued constraint, 25
- valued constraint satisfaction problem, 173, 175, 191, **194**
- vanishing point, 103, 105, 132, 146, 151, 180, 227, 229
- VCSP, 173, 175, 191, **194**
- VCSP(sm), 202
- vectorial equation solving, 71
- verticality, 232
- viewpoint-dependent edge, 127
- viewpoint-dependent vertex, 171
- virtual arc consistency, 203
- virtual cubic corner, 76
- virtual edge, 79
- virtual junction, 224
- volume completion, 235
- VSAT, 221
- W junction, 23, 37, 130
- W(+), W(-) junctions, 45, 130
- W₀, W₀₀ junctions, 171
- W/Y junction pair, 147
- warehouse allocation, 198
- weight, 192
- wireframe model, 126, 152
- wireframe-path constraint, 156, 159, 168
- wireframe projection, 125-126
- X junction, 37, 130
- X(=), X(≠) junctions, 132
- Y junction, 23, 37, 130
- Y(+), Y(-) junctions, 45, 130
- Y₀ junction, 171
- ZOA (zero/one/all) constraints, 217