International Journal of Computer Vision, 1, 73–103 (1987)
© 1987 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands

Interpreting Line Drawings of Curved Objects

JITENDRA MALIK
Computer Science Division, Department of EECS, University of California, Berkeley, CA 94720

Abstract

In this paper, we study the problem of interpreting line drawings of scenes composed of opaque regular solid objects bounded by piecewise smooth surfaces with no markings or texture on them. It is assumed that the line drawing has been formed by orthographic projection of such a scene under general viewpoint, that the line drawing is error free, and that there are no lines due to shadows or specularities. Our definition implicitly excludes laminae, wires, and the apices of cones.

A major component of the interpretation of line drawings is line *labelling*. By line labelling we mean (a) classification of each image curve as corresponding to either a depth or orientation discontinuity in the scene, and (b) further subclassification of each kind of discontinuity. For a depth discontinuity we determine whether it is a *limb*—a locus of points on the surface where the line of sight is tangent to the surface—or an occluding *edge*—a tangent plane discontinuity of the surface. For an orientation discontinuity, we determine whether it corresponds to a convex or concave edge. This paper presents the first mathematically rigorous scheme for labelling line drawings of the class of scenes described. Previous schemes for labelling line drawings of scenes containing curved objects were heuristic, incomplete, and lacked proper mathematical justification.

By analyzing the projection of the neighborhoods of different kinds of points on a piecewise smooth surface, we are able to catalog all local labelling possibilities for the different types of junctions in a line drawing. An algorithm is developed which utilizes this catalog to determine all legal labellings of the line drawing. A local minimum complexity rule—at each vertex select those labellings which correspond to the minimum number of faces meeting at the vertex—is used in order to prune highly counter-intuitive interpretations. The labelling scheme was implemented and tested on a number of line drawings. The labellings obtained are few and by and large in accordance with human interpretations.

1 Introduction

In this paper, we study the problem of interpreting line drawings of scenes composed of opaque solid objects bounded by piecewise smooth surfaces with no markings or texture on them. The vivid three-dimensional impression conveyed by line drawings in the complete absence of other cues points to the significance of this problem for visual perception. Line drawings also provide a vital source of constraint for other shape inference modules, e.g., the solution of the shape-from-shading differential equation relies on the boundary conditions provided by occluding contours.

It is trivially obvious that many different shapes could project to the same line drawing. What then do we mean by line drawing interpretation? Some researchers have attempted to determine a unique three-dimensional shape which supposedly is the one perceived by a human observer. Typically these approaches [1,3] use some criterion to choose one among the (infinite) possible spatial interpretations. For example, Brady and Yuille [3] search for the most compact shape, i.e., the one which maximizes the ratio of area to the square of the perimeter. It is easy to construct real world counterexamples where these criteria fail; e.g., Brady and Yuille's scheme interprets rectangles as slanted squares. Indeed, it is arguable whether the sparse data in a line drawing contains sufficient information to make such detailed quantitative inference in the absence of high-level model-driven processing.

If one abandons the desire to determine a unique three-dimensional shape, the logical alternative is to characterize in some useful and complete way the constraints on the infinite set of scenes which could project to the line drawing. This is the approach taken in this paper. It is our belief that the natural way to do this is to

1. partition the set of possible scenes into equivalence classes such that in each equivalence class all the scenes correspond to the same *line labelling*, and then
2. determine the metric constraints on three-dimensional shape that any of the scenes corresponding to a particular line labelling must satisfy.

By line labelling we mean (a) classification of each image curve as corresponding to either a depth or orientation discontinuity in the scene, and (b) further subclassification of each kind of discontinuity. For a depth discontinuity we determine whether it is a *limb*—a locus of points on the surface where the line of sight is tangent to the surface—or an occluding *edge*—a tangent plane discontinuity of the surface. For an orientation discontinuity we determine whether it corresponds to a convex or concave edge. Notation and precise definitions may be found in section 2.

In section 3, we list a set of position and orientation constraints that a scene corresponding to the line drawing must satisfy. As may be expected, lines with different labels exert different types of constraints on three-dimensional interpretation. It is this phenomenon which makes line labelling an important problem; it is difficult to formulate useful quantitative constraints on the scene which are independent of line labels.

The problem of labelling line drawings has received considerable attention in the computer vision community. Largely due to the work described by Huffman [11], Clowes [5], Mackworth [19], and Sugihara [35], the problem has been solved in a formal sense for scenes containing only polyhedral objects. The analyses and the resulting algorithms are mathematically rigorous. For curved objects, while several attempts were made [4,18,31,36], the analyses (and the resulting algorithms) were heuristic, incomplete, and lacked proper mathematical justification.

In this paper we present the first mathematically rigorous scheme for labelling line drawings of a very general class of curved objects.

We will deal with a simplified model of the world where the objects have no surface marks and where the lines due to illumination discontinuities like shadow edges and specularities have been removed in some preprocessing step. While currently there is no known algorithm for doing this preprocessing, some potentially useful techniques are discussed by Witkin [39] and Shafer [29]. The consequence of these restrictions is that we limit our attention to line drawings where each line corresponds either to a depth or orientation discontinuity. An example of such a drawing is figure 1. Note that none of the currently available edge detectors operating on a real image would yield such an idealized result—typically there would be missing lines, spurious lines, and missing and improperly classified junctions. This issue is discussed in section 16. For now, we will ignore the resulting difficulties and assume that a clean drawing like figure 1 can somehow be obtained.

We model curved objects as opaque solids bounded by piecewise smooth surfaces (defined in section 5). Examples of scenes containing such objects can be found in figures 29–34. There is no restriction on the number of faces that can meet at a vertex, and each face can be a connected portion of any smooth surface. The surfaces must be at least C^3, i.e., for their parametric representations, all derivatives of order $\leq$ 3 must exist. Other restrictions implicit in our definition lead to the exclusion of laminae, wires, and the apices of cones.

Consider a scene composed of the class of curved objects permitted by us. We catalog all possible junctions resulting from the orthographic projection of such a scene under general viewpoint.[1] In the course of this analysis, several intermediate results are proved which are of interest in their own light.

[1] Roughly speaking, this disallows accidental alignment.

While the labelling problem for polyhedra had been solved in a formal sense, a major limitation as pointed out by Draper [7] was the multiplicity of labellings produced (98 for a simple figure like the tetrahedron). This is in contrast to humans, who perceive very few interpretations. We propose a local minimum complexity rule—at each vertex select those labellings which correspond to the minimum number of faces meeting at the vertex—which is empirically demonstrated to be extremely successful in pruning the unwanted "weird" interpretations. (For the tetrahedron, we are left with three labellings corresponding to a tetrahedron floating in air, stuck to a table, or stuck to a wall.) While Draper's observations were made in the context of polyhedra, the same phenomenon occurs for curved objects and the same rule is employed successfully.[2]

An algorithm is developed which takes an idealized line drawing as input and makes use of the junction catalog to find all[3] legal labellings of a line drawing. The algorithm makes an additional restriction on the scene. Smooth transitions from convex to concave along an edge—as in figure 3—are not permitted.[4] A computer implementation of the algorithm was done and tested on several line drawings. The labellings obtained are few and by and large in accordance with human interpretations. Figures 29–34 display all the labellings found by our program for some scenes.

2 Preliminaries

Each point on an image curve in a drawing can have one of six possible *labels* which provide a qualitative characterization of three-dimensional physical shape at the point in the scene.

1. A "+" label represents a convex edge—an orientation discontinuity such that the two surfaces meeting along the edge in the scene enclose a filled volume corresponding to a dihedral angle less than π.
2. A "−" label represents a concave edge—an orientation discontinuity such that the two surfaces meeting along the edge in the scene enclose a filled volume corresponding to a dihedral angle greater than π.
3. A "←" or a "→" represents an occluding convex edge. When viewed from the camera, both the surface patches which meet along the edge lie on the same side, one occluding the other. As one moves in the direction of the arrow, these surfaces are to the right.
4. A "←←" or a "→→" represents a limb. Here the surface curves smoothly around to occlude itself. As one moves in the direction of the twin arrows, the surface lies to the right. The line of sight is tangential to the surface for all points on the limb. Limbs move on the surface of the object as the viewpoint changes.

We will use the term *connect edge* to mean either a convex or concave edge such that both the surfaces meeting along the edge are visible. The notation for different kinds of labels is illustrated in figure 1.

In line drawings of polyhedral scenes, the label is necessarily the same at all points on a single line segment. This permits us to use the term line label as opposed to label at a point on a line. For curved objects the label can change along a line. Because of this phenomenon, we need to distinguish between two different senses of line labelling.

A *dense labelling* is a function which maps the set of *all* points on curves in the drawing into the

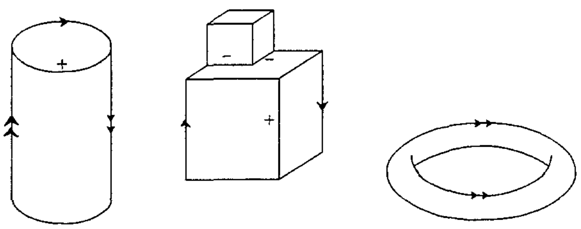

Fig. 1. Different kinds of line labels.

[2]Exactly which interpretations are "natural" and which are "weird" is to some extent a matter of personal preference. Even for the most complex examples tried, more than 50% of the interpretations found were judged by us to be "natural" as opposed to the less than 3% found for the simple tetrahedron in the absence of such a rule.

[3]Subject to the minimum number of faces rule.

[4]The mathematical analysis leading to the junction catalog is valid for these kinds of objects—our algorithm, however, makes this additional assumption.

set of labels. The dense labelling problem is to find all the dense labellings of a drawing which can correspond to a projection of some scene. Such a dense labelling is said to be *legal*. The set of legal dense labellings can be infinite (even uncountably infinite).

Instead of trying to find the label at each point on a curve, we could restrict our attention to sufficiently small neighborhoods of the junctions of the line drawing. For each line segment (between junctions) we now have to specify only two labels—one at each end. Of the 6^{2n} combinatorially possible label assignments to the n lines in a drawing, only a small subset correspond to physically possible scenes. We refer to these as *legal sparse labellings*. The determination of all legal sparse labellings of a particular line drawing is the sparse labelling problem. Note that the set of legal sparse labellings is always a finite set (usually small).

It may be noted that the definitions above are incomplete in an important sense—we have not yet specified the legal class of scenes. Examples are well known in line labelling literature (e.g., Draper [8]) of labellings which would be legal for curved objects but cannot correspond to any polyhedral scene. The class of scenes dealt with in this paper is defined in section 5.

In the case of polyhedra, a legal sparse labelling uniquely determines a legal dense labelling and vice versa. For drawings of curved objects, the set of dense labellings can be partitioned into equivalence classes where each equivalence class corresponds to a single sparse labelling.

The labelling problem addressed in this paper is the sparse labelling problem.

3 How Does a Line Labelling Constrain Solid Shape?

As mentioned earlier, lines with different labels correspond to different types of constraints. In this section we study the system of position and orientation constraints associated with a dense labelling of a line drawing. Most of these constraints are well known; our purpose is to provide a coherent list of the "fundamental" constraints and also supply additional motivation for the line labelling problem.

It is assumed that the line drawing has been formed by orthographic projection, with the eye along the z-axis at $z = +\infty$. We now consider the constraints from the different elements of a labelled line drawing.

1. *Shape constraint at a limb*. At limbs, one can determine the surface orientation uniquely. Let $\mathbf{n}$ be the unit surface normal, and $\mathbf{l}$ the unit tangent vector at a point on the limb. Obviously, $\mathbf{n.l} = 0$. As a limb corresponds to points on the surface where the line of sight vector $\hat{\mathbf{z}}$ lies in the tangent plane to the surface, we also have $\mathbf{n}.\hat{\mathbf{z}} = 0$ for points on the limb (equivalently $n_z = 0$). $\mathbf{n}$ therefore lies in the image plane and can be constructed by drawing the outward-pointing unit vector perpendicular to the projection of the limb in the image plane.

   What is stated above is the orientation constraint for the surface on which the limb lies, i.e., the surface on the right of the twin arrows. We also have a position constraint—the surface on the right of the twin arrows is nearer, implying a linear inequality between the z values on either side of the limb.

   The surface orientation constraint due to limbs is well known and has been used by Barrow and Tenenbaum [1] and by Ikeuchi and Horn [34].
2. *Shape constraint at an edge*. Let $\hat{\mathbf{e}}$ be the unit tangent vector to the edge at a point, and let $\mathbf{n}_1$ and $\mathbf{n}_2$ be the unit surface normals to the tangent planes to the two faces f_1 and f_2 at the point. Let $\hat{\mathbf{e}}$ be oriented such that, when one walks on the edge in the direction of $\hat{\mathbf{e}}$, the face f_1 is to the left. Now $\hat{\mathbf{e}}$ is perpendicular to $\mathbf{n}_1$ because $\hat{\mathbf{e}}$ lies in the tangent plane to the face f_1. Similarly $\hat{\mathbf{e}}$ is perpendicular to $\mathbf{n}_2$. Therefore $\hat{\mathbf{e}}$ is parallel to $\mathbf{n}_1 \times \mathbf{n}_2$. We do not know the vector $\hat{\mathbf{e}}$, but from a line drawing we can determine its orthographic projection into the image plane. We thus have the constraint $(\mathbf{n}_1 \times \mathbf{n}_2)_{\text{proj}} = \lambda\hat{\mathbf{e}}_{\text{proj}}$. Here the notation $\mathbf{v}_{\text{proj}}$ is used for the orthographic projection of $\mathbf{v}$ into the image plane. λ is a positive scalar if the edge is convex, and negative if the edge is concave. Note that this constraint is equally valid for occluding convex edges, where one of the surface normals corresponds to a hidden face.

This constraint when expressed using p, q—the gradient space coordinates—gives the rule used by Mackworth [19] and many other researchers in their gradient space constructions.

The position constraint at an edge is trivial —the depth z is continuous across a convex or concave edge and is discontinuous at an occluding edge.

3. *Shape constraint inside an area.* If each area in the image is to be the projection of a connected part of a smooth surface, the functions that map each image point to its position and orientation must be smooth within a single area. Also the surface normals at all the visible surface patches must have positive n_z components.

   Note that, as the surface normal in a smooth patch can be wirtten in terms of the partial derivatives of z with respect to x and y, these two functions are not independent. If we specify a C^2 function $z(x, y)$, the orientation function $\mathbf{n}(x, y)$ is automatically determined and smooth.

We feel that it is appropriate to regard the position and orientation constraints listed above as the fundamental system of constraints associated with a dense labelling of a line drawing. We will refer to this constraint set as the DL-system corresponding to a dense labelling. A candidate solution to this set of constraints is obtained by specifying

1. A piecewise smooth function $z(x, y)$ corresponding to the depth at each visible point in the scene.
2. A smooth function $\mathbf{n}_h(x, y)$ defined on all points on the lines in the drawing which are labelled $\leftarrow$ or $\rightarrow$. This function corresponds to the surface normal on the hidden face at that point on the occluding edge.

As pointed out earlier, the surface normal at each visible point is then automatically determined.

It is obvious that for a dense labelling of a line drawing to correspond to a legal scene, it is necessary that there exist a candidate solution which satisfies its DL-system. To show sufficiency, we also have to show that a suitable "completion" of the hidden parts of the scene exists which corresponds to a set of valid physical objects. We conjecture that this is always possible, but we do not have a rigorous proof.

We hope that the reader is now convinced of the primary importance of the dense labelling problem in line drawing interpretation. In this paper we present a solution to the sparse labelling problem, the motivation being to obtain a useful, finite characterization of the set of legal dense labellings. Because of the continuity of physical surfaces, the label at points along an image curve segment (between junctions) can undergo transitions only in a well-behaved fashion (see section 13).

For obvious reasons, in the rest of this paper, when we employ the ambiguous term "labelling problem," the correct term is usually "sparse labelling problem."

4 Review of Past Work on Line Labelling

The first successful attempt to solve this problem was made by Huffman [11] and Clowes [5] in 1971. They exhaustively cataloged the vertices that could arise in line drawings of trihedral objects (objects whose corners are formed by exactly three meeting faces) and then used the catalog to interpret lines as corresponding to convex, concave, or convex occluding edges. The catalog gives the possible labellings at each junction and global consistency is forced by the rule that each line in the drawing be assigned one and only one label along its length. Waltz [37] proposed an algorithm for this problem (actually for an augmented version with shadows, cracks, and separably concave edges) which reduced the search by a filtering step in which adjacent pairs of junctions are examined and incompatible candidate labellings discarded. Mackworth [19] developed the concept of *gradient space*, which enabled his program to label line drawings of arbitrary polyhedral scenes. One consequence of the attempt to deal with an arbitrary number of planes meeting at a vertex was a combinatorial explosion in the number of labellings generated which correspond to highly counter-intuitive interpretations. Draper [7] points out that there are 33 legal labellings (98 if accidental viewpoint is allowed) for the line drawing of a tetrahedron. In the context of the Origami World, Kanade [14] had to face a similar problem.

For objects bounded by curved surfaces, there have been two major efforts. The first was by Turner [36], who used a heuristic procedure called the PC (polyhedral-to-curved) transformation to obtain the labelling possibilities for junctions. Turner's approach suffered from several basic weaknesses:

- The Huffman–Clowes procedure is guaranteed to find all the legal labellings of a line drawing of a legal trihedral scene. Such a convincing claim cannot be made for Turner's procedure for the class of scenes it is supposed to handle.
- Turner's procedure is limited to objects such that each face is only one type of surface—planar, parabolic but nonplanar, elliptic or hyperbolic. A simple object like a torus which is bound by a single smooth surface which has both elliptic and hyperbolic patches can not be handled.
- A major problem is the huge number of junction labels and the consequent explosion in the number of legal interpretations (see table 1). This is to be contrasted with the small size of the Huffman–Clowes catalog.

The next major effort was by Shapira–Freeman–Chakravarty [4,31]. They considered objects such that exactly three faces meet at a vertex where each face is either a quadric surface or a plane. Their junction catalog is much smaller than that of Turner, which makes it practically usable in certain situations. However, some fundamental weaknesses remain:

1. No arguments are given to prove the validity of the junction catalog. One is left with the suspicion that it was "derived" by observation of junctions in some typical curved objects.
2. The scheme is limited to objects bounded by quadric surfaces/planes and exactly three faces meeting along a vertex.
3. For a nonoccluding edge, convex and concave edges are not distinguished.

Lee, Haralick, and Zhang [18] extend this catalog by adding line labels based on Huffman–Clowes rules. The justification for the validity of this step is not given. While this partially solves problem 3 mentioned above, the first two weaknesses remain. For a detailed discussion we would refer the reader to Malik [24], where we also point out some mistakes in these catalogs.

Table 1. Number of labellings for each of Turner's corner classes.

Corner	Labels
P_3	152
P_4	652
C_1	16
C_2	348
C_1P_1	138
C_1P_2	1905
E_1	8
E_2	205
E_1P_1	138
E_1P_2	443

5 Modelling the Scene and the Projection

The scene consists of a set of *objects* in three-dimensional space.

Definition. An *object* is a connected, bounded, and regular subset of $\mathbf{R}^3$ whose boundary is a piecewise smooth surface.

By *regular* we mean that it is the closure of its interior. This disallows objects with "dangling" faces or edges (the interior of a "dangling" face or edge is empty). Imposing this condition is a standard practice in solid modelling. The definition of *piecewise smooth surface* is the subject of the rest of this section.

The traditional approach in differential geometry to define and study surfaces is to decompose a given surface into a number of small pieces, each of which can be described parametrically as the function of two variables. A good development of the concepts and terminology of this approach may be found in Millman and Parker [26]. In what follows, we will assume that

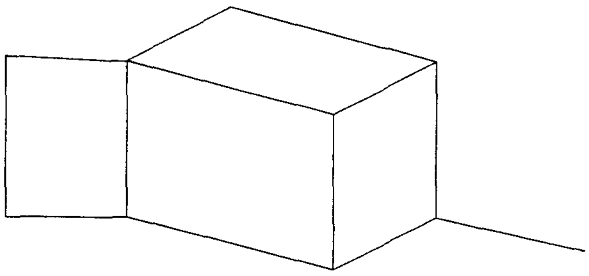
Fig. 2. Unacceptable object.

the reader is familiar with the definition of a C^k smooth surface.

We will need to enlarge the class of surfaces considered to include *piecewise smooth* surfaces. Roughly speaking, a piecewise smooth surface consists of portions of smooth surfaces joined together. The boundary of a polyhedron or a finite cylinder are simple examples. If two C^k surfaces intersect at a point P where the surfaces have distinct tangent planes, then it can be easily shown ([24], page 36) that the part of the intersection of the two surfaces near P is a smooth C^k arc. Such an arc on a piecewise smooth surface is called an *edge*. A point of intersection of three or more edges is called a *vertex*.

To formalize these notions, we first need to define a *surface element with boundary*. Recall that a *surface element without boundary* is just the image of a C^k coordinate patch.

Definition. Let $\mathbf{x}$: $U \rightarrow \mathbf{R}^3$ be a C^k coordinate patch, where U is a connected open subset of $\mathbf{R}^2$. Let D be a region such that

1. its boundary, ∂D, consists of a finite number of piecewise smooth simple closed curves.
2. $D \cup \partial D \subset U$

Then S, the image of the restriction of $\mathbf{x}$ to $D \cup \partial D$, is said to be a surface element with boundary. It can be shown that ∂S, the boundary of S, is the image of ∂D.

Note that $\mathbf{x}$ and its derivatives are defined for all points of $D \cup \partial D$ and hence a tangent plane is defined for all the points of a surface element with boundary.

From now on, we will use the term *surface element* for both kinds of surface elements—with and without boundary. We are now ready to define *piecewise smooth surfaces*. We use the notation $B_\epsilon(P; M) = B_\epsilon(P) \cap M$ to denote the ϵ-neighborhood of P relative to M. $B_\epsilon(P)$ is, as usual, the open ball of radius ϵ in $\mathbf{R}^3$ centered at P.

Definition. A C^k piecewise smooth surface in $\mathbf{R}^3$ is a subset $M \subset \mathbf{R}^3$ for which there exists a collection of surface elements $S = \{S_1, \ldots, S_n\}$ with each of the S_i, $i = 1, \ldots, n$ a subset of M. For every point $P \in M$ exactly one of the following three conditions is true:

1. There exists an $S_i \in S$ such that $B_\epsilon(P; M) = B_\epsilon(P; S_i)$ for some $\epsilon > 0$. In other words, P is the interior point of some surface element S_i. If there are two or more such surface elements, then on the overlap the corresponding patches are related by a C^k coordinate transformation as in the definition for C^k surfaces.
2. There exist two surface elements S_i, S_j in S, such that for some $\epsilon > 0$,
   $B_\epsilon(P; M) = B_\epsilon(P; S_i) \cup B_\epsilon(P; S_j)$
   Furthermore the tangent planes to S_i and S_j are distinct at P.
3. There are $m \geq 3$ surface elements $S_{i_1}, S_{i_2}, \ldots, S_{i_m}$ such that for some $\epsilon > 0$,
   $$B_\epsilon(P; M) = B_\epsilon(P; S_{i_1}) \cup B_\epsilon(P; S_{i_2}) \cup \ldots \cup B_\epsilon(P; S_{i_m})$$
   For any two of these surface elements, the tangent planes are distinct at P.

Depending on which of the above three conditions is true at P, P is said to be a face point, edge point, or a vertex, respectively.

Definition. A *face* is a connected set of face points.

Definition. An *edge* is a connected set of edge points.

The definition given above would allow degenerate cases like two cubes connected by a single vertex. In the definition of polyhedra [6], such cases are excluded by requiring that the faces surrounding each vertex form a simple circuit. We impose a similar condition by requiring that the surface elements surrounding each vertex form a simple circuit.

Example. A polyhedron is a C^∞ piecewise smooth

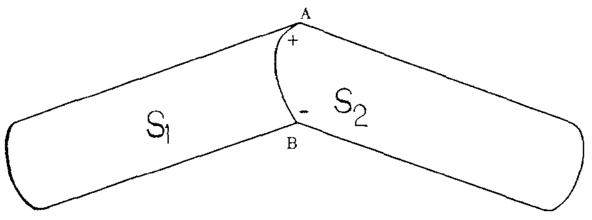

Fig. 3. Not a piecewise smooth surface.

surface. Our definitions for faces, edges, and vertices reduce to the standard definitions.

Example. A right circular cylinder is a C^∞ piecewise smooth surface with 3 faces, 2 edges and 0 vertices.

An important issue is the order of smoothness needed. We restrict our attention to C^3 piecewise smooth surfaces. This choice is purely for mathematical convenience—as a consequence of this assumption, each face is a subset of a C^3 surface and we can apply Whitney's theorem [38] to study how it projects.

We would like to point out that our definition of piecewise smooth surfaces differs somewhat from some other definitions, which do not require that the tangent planes be distinct across an edge. An example of a surface which is *not* piecewise smooth under our definition is shown in figure 3. There is a point on AB at which the tangent planes to surfaces S_1 and S_2 are the same. The requirement that the tangent planes always be distinct along an edge forces each edge to be either convex or concave. Smooth transitions from convex to concave, as along AB, are not permitted. This consequence is a crucial one for the algorithm developed in section 14.

The process of image formation is modelled as orthographic projection, which corresponds to the eye/camera being effectively at infinite distance from the scene. Given an image plane, we assume a right-handed coordinate system with x- and y-axes lying in the plane. The eye is assumed to be located on the z-axis at $z = +\infty$. A point (x,y,z) in the scene projects to the point (x,y) in the image plane, and is visible if there is no other point (x,y,z') belonging to any object in the scene with $z' > z$. The projection of the scene is the projection of the visible points in the scene.

The viewpoint is assumed to be *general*—there exists an open neighborhood of the vantage point in which the "topological" structure of the line drawing remains unaltered. This will be made more precise in the next section.

6 The Line Drawing

As pointed out earlier, we are dealing with a simplified model of the world so that the only lines we need to consider are the projections of depth and orientation discontinuities. In this context we can make our definition of line drawing precise. Each point in the image is the projection of a visible point in the scene. With each point (x,y) in the image can be associated a scalar z and a unit vector $\mathbf{n}$ corresponding respectively to the depth and the direction of the surface normal at the associated point in the scene. At most points in the image, these functions are continuous (at least C^1, in fact C^2 because of the restriction on our scenes). The locus of discontinuities of either of these functions constitutes the line drawing. The curves in the line drawing are segmented at tangent and curvature discontinuities. The points where there are tangent or curvature discontinuities are referred to as junctions. Figure 4 is a sample line drawing. Endings of image curves, e.g., j_3, are also referred to as junctions as are points where two or more image curve segments meet, e.g., j_4, j_5, j_6.

From the line drawing, an image structure graph may be constructed. It is an undirected graph. Its nodes are all the junctions in the line drawing and additionally *pseudo*-junctions like j_8, one for each isolated smooth closed curve like C_1. Each image curve segment corresponds to an arc between the vertices corresponding to the junctions on which it is incident. For example, the

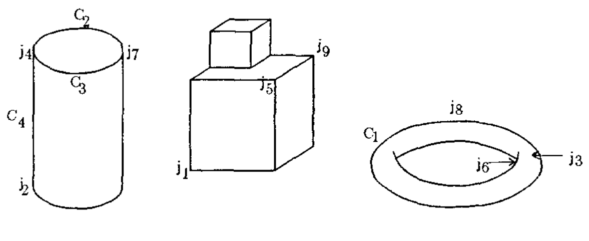

Fig. 4. Different kinds of junctions.

node corresponding to j_4 has arcs corresponding to C_2, C_3, C_4—the first two to the node corresponding to j_7 and the third to the node corresponding to j_2. The node corresponding to j_6 has three arcs; the ones corresponding to j_3, j_8 have only one arc each. As is obvious, in general the *image structure graph* (henceforth the ISG) may be disconnected, have more than one arc between two nodes, and have self-loops.

By considering the observed geometric properties in the line drawing, these junctions can be classified as follows:

Terminal: Curve ends there, e.g., j_3.
L: Tangent discontinuity across junction, e.g., j_1.
Curvature-L: Tangents continuous, curvature discontinuity, e.g., j_2.
T-junction: Two of three image curves at the junction have same slope and curvature, e.g., j_6.
Pseudo: Corresponds to isolated, closed, smooth curves, e.g., j_8.
Three-tangent: Three curves with common tangent. Two have same curvature, e.g., j_4.
Arrow: Three curves with distinct tangents. One angle $> \pi$, e.g., j_9.
Y: Three curves with distinct tangents. No angle $> \pi$, e.g., j_5.
Multi: Four or more image curves at the junction.

The ISG is augmented by storing at each node an attribute field corresponding to which one of the above classes the junction belongs.

We can now define precisely what we mean by general viewpoint. Consider the set of line drawings formed by viewing a scene from different points of a sufficiently small spherical neighborhood of the vantage point. If the augmented ISGs corresponding to these line drawings are isomorphic, the viewpoint is said to be general.

7 Projection of Curved Objects

We wish to study how curved objects (as defined in section 5) project to a line drawing. This is done by studying how neighborhoods of different kinds of points on the surface project and cataloging the resulting junctions in the line drawing. This can be broken up into three cases:

- The projection of a neighborhood of an interior point of a face.
- The projection of a neighborhood of an interior point of an edge.
- The projection of a neighborhood of a vertex.

These are tackled respectively in sections 8–10.

As we are dealing with the projection of *opaque* surfaces, we also have to worry about the phenomenon of occlusion–obstruction of the view of a surface of an object by another object (or another part of the same object). This gives rise to T-junctions. Here we know that the top of the T-junction corresponds to a nearer surface occluding another object. Note that there is no constraint on the label of the stem of the T-junction.

8 Projection of a Face Neighborhood

This corresponds to the projection of a single C^3 surface element. It is an instance of the class of mappings from two-dimensional manifolds to two-dimensional manifolds. Whitney [38] showed that generically there are only two singularities—the *fold* and the *cusp*. This result is discussed in section 8.1. In section 8.2 we study Whitney's theorem in the context of the projection mapping. Limbs (e.g., C_1 in figure 4) are associated with the fold singularity and terminals (e.g., j_3 in figure 4) are associated with cusps. At a terminal, we can determine which of the two surface patches is nearer.

8.1 Whitney's Singularity Theory

In 1955, Whitney published a landmark paper

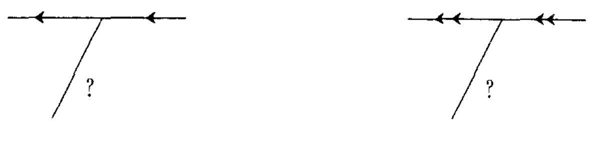

Fig. 5. Labels for a T-junction.

[38] on the singularities of mappings[5] of open sets in E^2 into E^2. Examples of such mappings are projection (orthographic and perspective) of surfaces and the Gauss map. Whitney showed that generically there are only two kinds of singularities: *folds* lying along curves and isolated *cusp* points lying on the folds. We will explain what this means in the context of projection in the next section. In this section we will define various terms and try to give a feel for Whitney's results.

Such a mapping is defined by the two functions $u_1 = f(x_1,x_2)$ and $u_2 = g(x_1,x_2)$ where (x_1,x_2) and (u_1,u_2) are the coordinates in the two spaces. Let J be the Jacobian of the mapping. A point p is said to be a regular if $J(p) \neq 0$; otherwise it is singular. We are interested in studying the locus of the singularities.

Example 1. Consider the mapping

$$u_1 = x_1^2,\ u_2 = x_2$$

The Jacobian $J = 2x_1 = 0 \Longrightarrow x_1 = 0$. The straight line $x_1 = 0$ is the locus of points where the mapping is singular. This is an example of a fold.

Example 2. Consider the maping

$$u_1 = x_1^3 - x_1x_2,\ u_2 = x_2$$

The Jacobian $J = 3x_1^2 - x_2 = 0 \Longrightarrow x_2 = 3x_1^2$. This corresponds to two fold curves, one for positive x_1 and onc for negative x_1—both in the half-plane where $x_2 \geq 0$. The two fold curves meet at the origin at a cusp point. This singularity occurs whenever two fold curves come together and disappear.

Our choice of examples was not accidental. After suitable coordinate transformations, all folds and cusps can be described by the canonical forms in examples 1 and 2, respectively. Whitney showed that every singularity of a smooth mapping from E^2 to E^2, after an appropriate small deformation, splits into folds and cusps. As we discuss in the next section, this *generic* property in the vision context corresponds to general viewpoint.

[5]It is assumed that the mappings are at least C^3 smooth. This explains our choice of C^3 piecewise smooth surfaces to define curved objects.

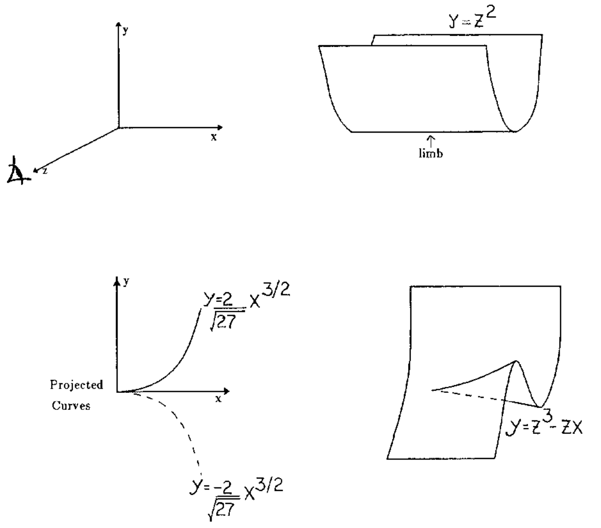

Fig. 6. Two canonical examples.

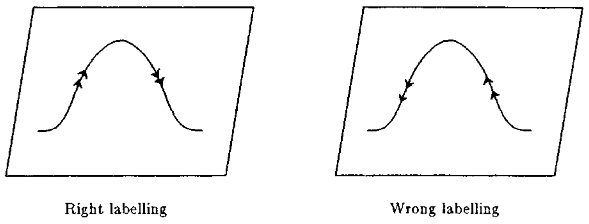

Fig. 7. Inferring the labelling from terminals.

8.2 Singularities of the Projection Mapping

Projection is a mapping from a surface into a plane. One can immediately interpret examples 1 and 2 as corresponding to the orthographic projection of two surfaces of the form $y = f(x,z)$ viewed from an infinitely distant point on the z-axis. Figure 6 shows the two surfaces.

For $y = z^2$ the projection of the fold curve is the line $y = 0$.

For $y = z^3 - zx$ the two fold curves have the equation

$$x = 3z^2$$

one for z positive, the other for z negative.

By eliminating z we get the equations of the projected curves

$$y = z^3 - z(3z^2) = -2z^3 = -2(\pm\sqrt{x/3})^3$$
$$= \pm\frac{2}{\sqrt{27}}x^{3/2}$$

This is a semi-cubic parabola with a cusp at the origin. Only the positive branch is visible. Note that the contour ends concavely. For an extended discussion of this, see Koenderink and Van Doorn [16]. This fact can be used to determine to which side the curve belongs as shown in figure 7.

What does all this imply for the labelling problem? The only curves which exist in the projection of a smooth surface patch are limbs (folds), and the only junctions are terminals (cusps). In the line drawing, each limb projection borders two regions, or sometimes two strips of the same region. The limb curve lies on the surface patch corresponding to one of these strips and is in front of the other surface patch. Which is the nearer patch can be determined by looking at the curvature of the projection of the limb (in the image plane) at a terminal junction as shown in figure 7. If the scene consists only of objects bound by single smooth surfaces (no edges), then the only junctions in the line drawing would be T-junctions and terminals.

9 Projection of an Edge Neighborhood

Locally, an edge e is the intersection of two surface patches S_1 and S_2 with different tangent planes. Consider a point P in the interior of this edge. To study how P and its neighborhood in S_1,S_2 project, we have to consider three cases:

- No limb through P on either S_1,S_2.
- Limb through P on both of S_1,S_2.
- Limb through P on one of S_1,S_2.

The first case is easy. Since there is no limb on either patch, the projection of both the patches are diffeomorphisms. The edge segment in the neighborhood of P is the boundary of both patches and hence the boundary of both their

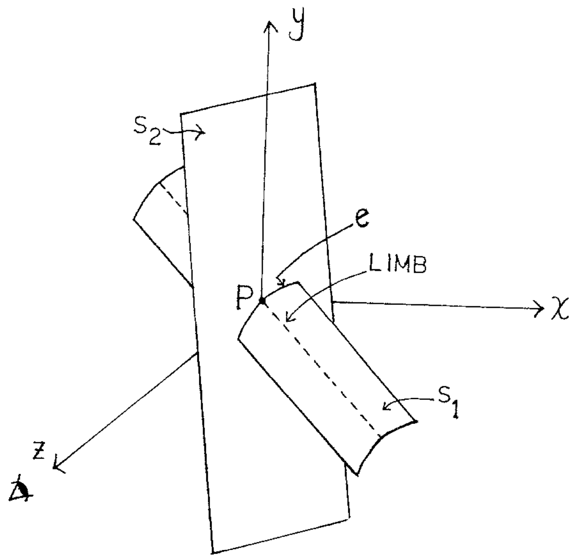

Fig. 8. The coordinate system.

projections. The edge segment has the same line label (convex, concave, or occluding convex) on both sides of P.

The second case is also easy. As there is a limb on S_1, the ray from the viewer must lie on the tangent plane to S_1 at P. Similarly it must lie on the tangent plane to S_2 at P. It therefore must lie on their intersection, which is a straight line. In other words, the vantage point is constrained to lie along a line. That would be a violation[6] of the general viewpoint assumption stated in section 6.

The third case is more interesting. Physical examples of points of this kind are the points in the scene corresponding to junctions A, B, C, and D in figure 13. Without loss of generality, we can assume that S_1 has a limb passing through P. For S_2 we will assume a general equation and then do the case analysis. Cartesian coordinates are introduced with the origin at P. The eye is along the z-axis at $z = +\infty$, so that the projection is on a plane parallel to the x–y plane (see figure 8).

For surface S_1

$$y_1 = a_2z^2 + a_3zx + a_4x^2 \qquad (1)$$

Without loss of generality, we have assumed that

[6]Strictly true only for an infinite precision line drawing. Figures 38 and the accompanying discussion in section 16 gives an example of what can happen under finite resolution.

the $x - z$ plane is tangent to S_1.

Now, the limb is given by

$$\frac{\partial y_1}{\partial z} = 2a_2 z + a_3 x = 0 \Longrightarrow z = \frac{-a_3 x}{2a_2}$$

By substituting back, we get

$$y_1 = (a_4 - \frac{a_3^2}{4a_2})x^2 \qquad (2)$$

This is the equation of the limb on S_1 in the neighborhood of the origin.

For surface S_2

$$y_2 = b_0 z + b_1 x + b_2 z^2 + b_3 zx + b_4 x^2 \qquad (3)$$

As before, any limb on this would be given by

$$\frac{\partial y_2}{\partial z} = b_0 + 2b_2 z + b_3 x = 0 \Longrightarrow$$
$$z = \frac{-b_3 x - b_0}{2b_2}$$

By substituting back, we get the limb equation

$$y_2 = -\frac{b_0^2}{4b_2} + (b_1 - \frac{b_0 b_3}{2b_2})x + (b_4 - \frac{b_3^2}{4b_2})x^2$$

Recall that we have assumed that there is no limb on this patch passing through P. This implies that $b_0 \neq 0$, since $x = 0$ at P.

From (1) and (3), the equation of the intersection curve of S_1 and S_2, i.e., the edge, is:

$$-b_0 z - b_1 x + (a_2 - b_2)z^2 + (a_3 - b_3)zx + (a_4 - b_4)x^2 = 0$$

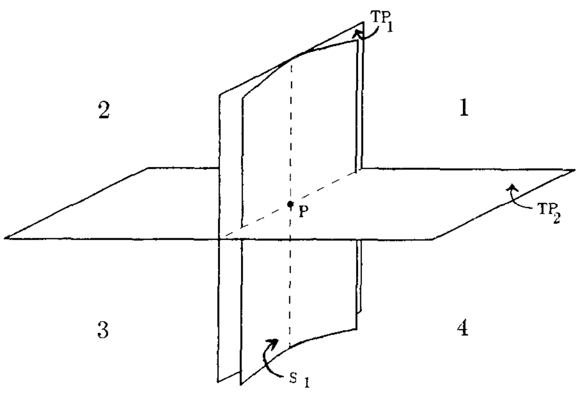

Fig. 9. The four quadrants.

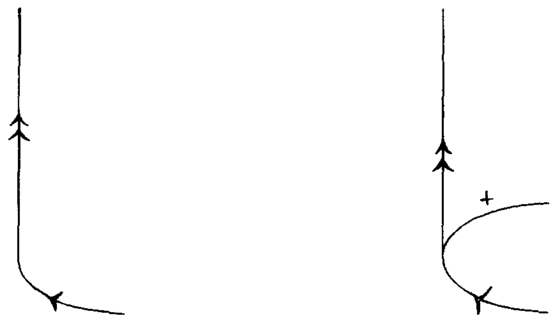

Fig. 10. Viewing quadrant 1.

from which

$$\frac{dz}{dx} = \frac{b_1 - 2(a_4 - b_4)x - (a_3 - b_3)z}{-b_0 + 2(a_2 - b_2)z + (a_3 - b_3)x}$$

What we really want is the slope of the tangent to the projected edge, i.e.,

$$\frac{dy}{dx} = \frac{\partial y}{\partial z}\frac{dz}{dx} + \frac{\partial y}{\partial x}$$

Substituting, we get

$$\frac{dy}{dx} = (2a_2 z + a_3 x)\frac{b_1 - 2(a_4 - b_4)x - (a_3 - b_3)z}{-b_0 + 2(a_2 - b_2)z + (a_3 - b_3)x} + (a_3 z + 2a_4 x) \qquad (4)$$

As $b_0 \neq 0$, $\frac{dy}{dx} = 0$ at P, or stating this in words, the slope of the tangent to the intersection curve is 0 at P. From (2), the equation of the limb, it is easy to see that the slope of its projection is also 0 at P. This means that the projection of the intersection curve is tangent to the projection of the limb curve.

We are now ready to list the junctions which arise from the projection of a sufficiently small neighborhood of P. Let TP_1 be the tangent plane to S_1 at P. The viewpoint is constrained to lie in this plane. Let TP_2 be the tangent plane to S_2 at P. S_1 and S_2 divide the three-dimensional space in the neighborhood of P into four quadrants as shown in figure 9. By putting in solid material in various quadrants and viewing from various directions in TP_1 we can generate all possibilities. The reader will note the similarity of this procedure to the procedure used by Huffman [11] for the trihedral world. To begin the case analysis:

1. Solid material in only one quadrant. This gives two subcases:
   a) Solid material in quadrant 1. Depending on whether the viewpoint is in the upper or lower half-plane, we get the two junctions in figure 10. Recall that we had shown earlier that the projection of the edge is tangent to the projection of the limb. Examples of these junctions are A and B in figure 13.
   b) Solid material in quandrant 2. This gives us the "junction" in figure 11. Note that here the limb curve itself is occluded. To help visualize the physical situation, examine junction C in figure 13. The dotted line there corresponds to the hidden limb. Unlike the other junctions, this cannot be identified directly because it does not correspond to any tangent or curvature discontinuity in the line drawing. We have to allow for the possibility of this junction being present by introducing "phantom" nodes on all curved lines in the drawing which could correspond to convex edges. (This "junction" cannot occur on projections of limbs or concave edges.)

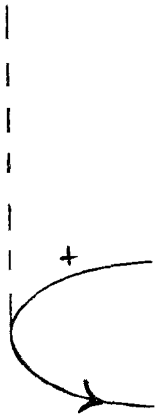

Fig. 11. Another "junction".

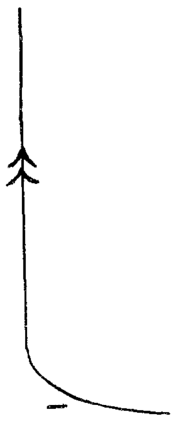

Fig. 12. One more labelling for a curvature-L.

2. Solid material in two quadrants. There are two subcases. One is when adjacent quadrants are occupied. In that case there is only one surface at *P*. The other case is when opposite quadrants are occupied, for example (1,3) or (2,4). In this case, *e* violates our definition of an edge.
3. Solid material in three quadrants. In the first subcase, let 1 be the empty quadrant. In this case *P* is hidden. If 2 is the empty quadrant, we get the junction in figure 12. We get an example of this junction when a cylinder is joined to a plane—junction D in figure 13.

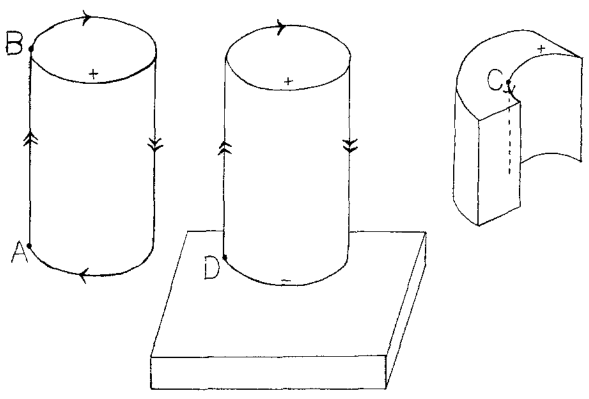

Fig. 13. Objects corresponding to different junctions.

4. Solid material in all four quadrants. Here no surfaces are defined.

10 Projection of a Vertex Neighborhood

We begin our analysis of the projection of a vertex with the observation that, under general viewpoint, no limb can pass through a vertex. As the vantage point moves, the limb curve moves on the surface and it interests the vertex only for a particular viewpoint.

Our next observation is elementary if one is familiar with differential geometry. However, for completeness, we give a simple proof.

Theorem. If two surface elements S_1 and S_2 intersect along a curve C at the point P, then TP_1 (the tangent plane to S_1 at P) and TP_2 (the tangent plane to S_2 at P) intersect along a straight line l such that at P, the projection of l on the image plane is *tangent* to the projection of C.

Proof. Consider the Taylor Series Expansions of the graphs of the two surfaces with P as the origin. The third- and higher-order terms can be ignored without loss of generality.

$$y_1 = a_0z + a_1x + a_2z^2 + a_3zx + a_4x^2$$
$$y_2 = b_0z + b_1x + b_2z^2 + b_3zx + b_4x^2$$

First let us find the equation of the edge curve along which these surfaces intersect. This is given by

$$(a_0 - b_0)z + (a_1 - b_1)x + (a_2 - b_2)z^2 + (a_3 - b_3)zx + (a_4 - b_4)x^2 = 0$$

from which

$$\frac{dz}{dx} = -\frac{(a_1 - b_1) + 2(a_4 - b_4)x + (a_3 - b_3)z}{(a_0 - b_0) + 2(a_2 - b_2)z + (a_3 - b_3)x}$$

What we really want is the slope of the tangent to the projected curve, i.e.,

$$\frac{dy}{dx} = \frac{\partial y dz}{\partial z dx} + \frac{\partial y}{\partial x}$$

Substituting, we get

$$\frac{dy}{dx} = -(a_0 + 2a_2z + a_3x) \frac{(a_1 - b_1) + 2(a_4 - b_4)x + (a_3 - b_3)z}{(a_0 - b_0) + 2(a_2 - b_2)z + (a_3 - b_3)x}$$
$$+ (a_1 + a_3z + 2a_4x)$$

At the origin, this simplifies to

$$\frac{dy}{dx} = \frac{a_0b_1 - a_1b_0}{a_0 - b_0} \tag{5}$$

Now consider the equations of the two tangent planes

$$y_1 = a_0z + a_1x$$
$$y_2 = b_0z + b_1x$$

The equation of the intersecting line is given by

$$(a_0 - b_0)z + (a_1 - b_1)x = 0$$

Using this to eliminate z, we get

$$y = \frac{a_0b_1 - a_1b_0}{a_0 - b_0}x \tag{6}$$

which has the same slope as (5).

This result has an immediate consequence. The projection of a vertex *locally* looks like the projection of an equivalent polyhedral vertex formed by replacing each of the surface elements by their tangent planes. This results in a great simplification in the analysis, as all the results on polyhedral junction labelling become relevant. Find the equivalent straight line junction by replacing each image curve at the junction by its tangent and look up (or derive) its labelling possibilities from a polyhedral junction catalog. For example, if it is known *a priori* that exactly three surface elements meet at a vertex, then the labelling possibilities are exactly those of the Huffman–Clowes set (see figure 14). As we want to deal with a more general class, further analysis will be necessary as in the next subsection.

10.1 Labelling Polyhedral Junctions

In section 4, we listed the most significant pieces of work on this problem—starting with Huffman–Clowes labelling for the trihedral world and Mackworth's gradient space approach for dealing with arbitrary number of surfaces meeting at a vertex. We also noted Draper's empirical observations on the combinatorial explosion in the number of alternative interpretations when no restrictions are made on the number of surfaces meeting at a vertex.

It is clear that we need some way of pruning

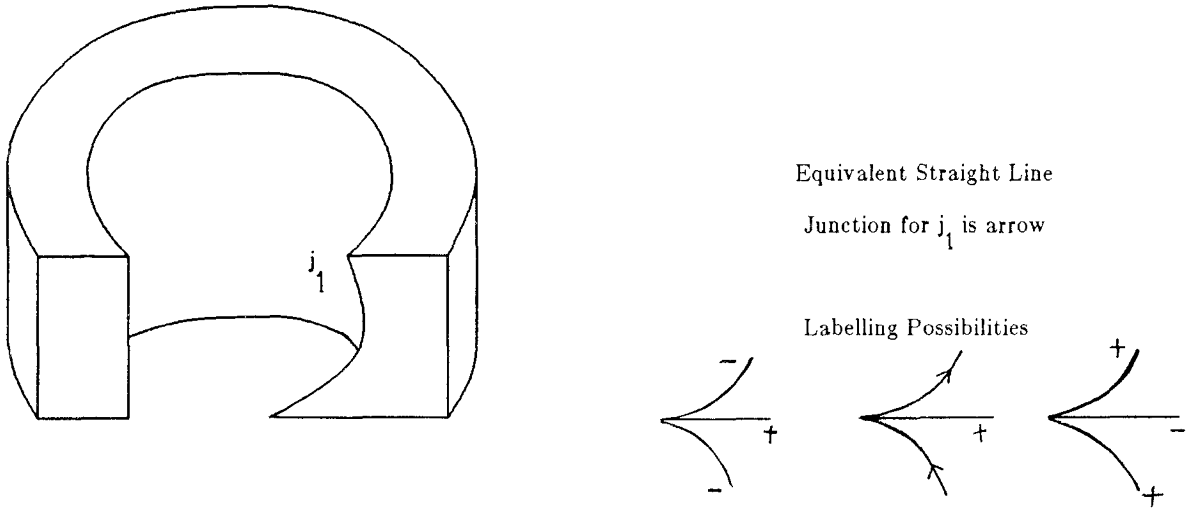

Fig. 14. Equivalent polyhedral junction.

these weird interpretations. Kanade's heuristics unfortunately are applicable only to the classes of objects which have parallel edges and faces with axes of symmetry. We need a criterion which is more generally applicable. Several attempts [1,3] have been made to find simplicity criteria/ minimization schemes to find the psychologically preferred interpretations of line drawings. Most of these approaches are limited to isolated image curves. Instead of attempting to find grand global simplicity criteria, we limited ourselves to local simplicity. Based on the observation that all the highly counter-intuitive interpretations involve a number of hidden faces, we develop our criterion—*for each junction find the vertex interpretations with the minimum number of faces meeting at the vertex*. The interpretation should be stable under general viewpoint. Now for polyhedra, at a vertex there are exactly two faces sharing an edge, and exactly two edges bounding a face. It follows that the number of edges at a vertex is equal to the number of faces incident at the vertex. Therefore we have an equivalent version of the rule—find interpretations involving the minimum number of edges.

An example will make this clearer. Consider an arrow junction as shown in figure 14. Each of the three lines which meet at the junction is the projection of an edge which is incident on the corresponding vertex. From our simplicity rule, we try to find vertex interpretations which require only three edges—or equivalently only three faces meeting at the vertex. This means that, for the arrow, our local labelling possibilities are the same as that for the Huffman–Clowes scheme for the trihedral world. By the same reasoning, for the Y-junction, we get the same three labelling possibilities as in the Huffman–Clowes scheme. For L-junctions, we need three faces as well—there are no legal interpretations with two faces. Here again, the labelling possibilities are the same as that of the Huffman–Clowes scheme.

For higher-order junctions with four and more lines meeting at the junction, we need n faces meeting at the vertex if there are n lines meeting at the junction. We need a way to generate the labelling possibilities for such higher-order junctions as may occur in a line drawing, e.g., in overhead views of square pyramids. Our algorithm to do this is based on a gradient space construction reminiscent of those used by Mackworth [19] and Huffman [12]. To help in enforcing the minimum number of faces rule, we need the following theorem which is proved in Malik [24].

Theorem. If there are n lines meeting at a junction, $n \geq 3$, all the labelling possibilities for the junction which correspond to the minimum number of faces at the vertex, correspond to either zero or one hidden face at the junction.

Now we are ready to generate the labelling possibilities. To do this, first consider the simpler case—all the labels are connect. If a labelling is legal, it should be possible to construct a reciproc-

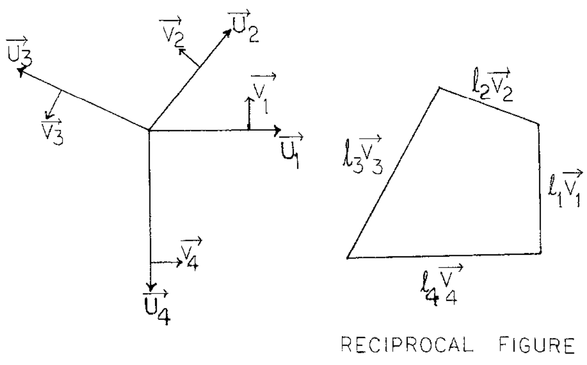

Fig. 15. Generating junction labelling possibilities.

al figure in gradient space.

Instead of treating this as a geometric construction with ruler and pencil, we can use vector notation. Let $\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_n$ be the (outward) pointing vectors corresponding to the lines which meet at the junction. Let $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_n$ be unit vectors perpendicular to these lines corresponding to a counterclockwise rotation by 90 degrees. Consider the vector equation

$$l_1\mathbf{v}_1 + l_2\mathbf{v}_2 + \ldots + l_n\mathbf{v}_n = 0 \tag{7}$$

where $l_1,l_2,\ldots,l_n$ are n scalars. If the reciprocal figure is constructible, it corresponds to a solution of this equation. Figure 15 illustrates this. Here equation (7) has a solution with l_1,l_2,l_3,l_4 all > 0; and the all convex labelling for the junction is legal. Note that our "n faces for n lines" assumption is implicitly buried in this equation. If there were hidden lines corresponding to two hidden faces meeting, they would have given rise to additional terms on the left-hand side of the equation.

This equation being a two-dimensional vector equation is actually two linear equations in the n unknowns $l_1,l_2,\ldots,l_n$. A convex (concave) labelling for a line implies that the corresponding variable $l_i > 0$ (< 0). Given a proposed labelling for the lines meeting at the junction, we have a system of $n + 2$ linear constraints (2 equality constraints and n inequality constraints). If the system has a feasible solution, the labelling is legal; else it is not. A naive approach to finding all possible legal labellings would be to construct 2^n linear programming problems corresponding to the 2^n possible labellings, and then determining for each problem the existence of a feasible solution. Of course, one can do much better than that—a fast algorithm is described in Malik [24].

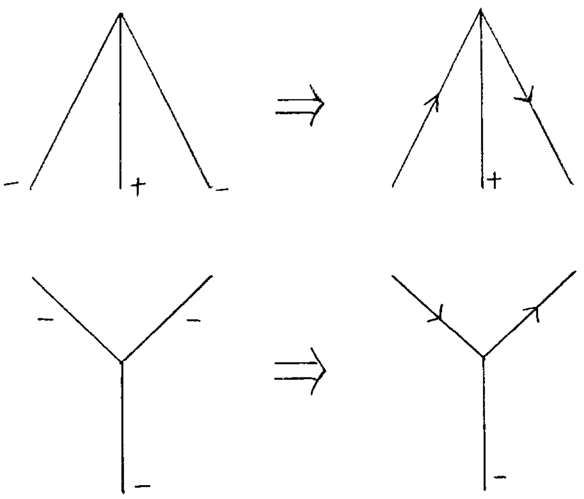

Fig. 16. Generating junctions with occlude labels.

By changing the signs of all the variables in the linear system described above, we see that labellings come in pairs. This is another way of explaining the Necker ambiguity corresponding to the convex/concave reversal.

To find the legal labellings corresponding to one hidden face is easy. Take a legal all (convex/concave) edge labelling. Each pair of adjacent lines at the junction define a sector. Consider a sector defined by two lines A and B which have been labelled concave. Consider the face defined by the corresponding edges. If this face were hidden, i.e., both these lines corresponded to convex occluding edges instead of concave edges, the reciprocal figure would remain the same. One can therefore label A and B as occluding convex edges—with the direction of the arrow such that the sector defined by A and B is to the left. Figure 16 shows this procedure.

This hierarchical determination of labelling possibilities—first between convex and everything else in the solution of equation (7) and then subsequent refinement—is also a good strategy for doing consistency checking. This is discussed in section 12.2.

11 The Junction Catalog

The results of the analysis in sections 7–10 can be

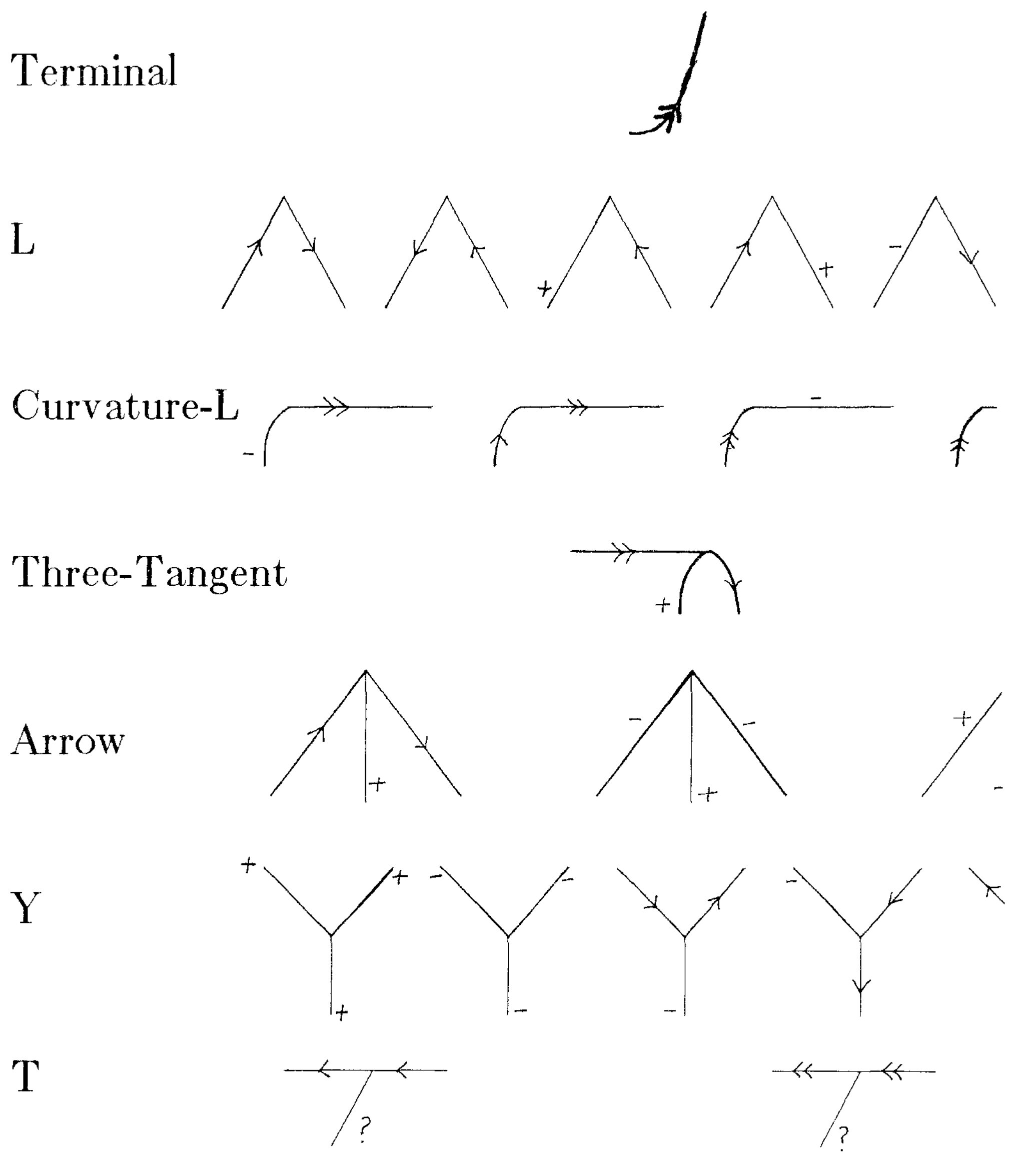

Fig. 17. The Junction Catalog for Piecewise Smooth Surfaces.

summarized in the catalog in figure 17.

12 Labelling and Constraint Satisfaction Problems

Ignoring for the moment the invisible "junction" in figure 11, conceptually the labelling algorithm is straightforward. The local possibilities at each junction with ≤ 3 lines have been enumerated in section 11 and, for multi junctions, the labellings can be computed by the procedure described in section 10.1. Consistency is forced by requiring the label at each end of the line to be the same.[7] This makes the problem just a particular instance of the class of constraint satisfaction problems. Mackworth [20] presented a unified framework for these problems and defined a class of algo-

[7]Except that the "invisible" junctions have to be allowed for in some way.

rithms useful for solving them. Mackworth and Freuder [21] analyzed the time complexity of these algorithms. Adequate background for understanding our paper may be found in section 12.1. In section 12.2, we develop the notion of collapsed constraint satisfaction problems which can lead to great speedup for certain kinds of problems—line labelling being one of them. A strategy for dealing with "invisible" junctions is formulated in section 13. These ideas are used to develop the algorithm which is described in section 14. The algorithm was implemented and the results of some test runs are presented in section 15.

12.1 Constraint Satisfaction Problems

Following Mackworth [20], a *constraint satisfaction problem* (CSP) may be defined as follows: Given a set of n variables $(v_1, \ldots v_n)$ with associated domains $(D_1, \ldots D_n)$ and a set of constraining relations each involving a subset of the variables, find all possible n–tuples such that each n–tuple is an instantiation of the n variables satisfying the relations. We will limit ourselves to CSPs where the domains are discrete, finite sets and the relations (predicates) are unary and binary. We will use P_i to denote the unary predicate to be satisfied by v_i and P_{ij} to denote the binary predicate to be satisfied by v_i, v_j. It will be required that $P_{ij}(v_i, v_j) \equiv P_{ji}(v_j, v_i)$. To put line labelling in this framework, we let the variables correspond to the junctions in the line drawing. The initial variable domains are simply the possible interpretations for the junctions (for an L-junction this set will have six elements) and the binary predicates simply require each curve to have the same label at both ends. Waltz's filtering algorithm—one of the earliest attempts to reduce unnecessary search in solving CSPs—was developed in this context.

A straightforward approach to solve a CSP is backtracking. The variables are sequentially instantiated from ordered representations of their domains. As soon as all the variables of any predicate are instantiated, its truth value is tested. If it is true the process of instantiation and testing continues, but if it is false the process fails back to the last variable instantiated that has untried values in its domain and reinstantiates it to its next value. Backtracking can be grossly inefficient and a solution tends to be exponential both in the worst and average case.

Several CSPs are known to be NP-complete—in particular Huffman–Clowes labelling for trihedral scenes [15]. It is unlikely therefore that a polynomial time algorithm exists for solving general CSPs. Accordingly, the class of network consistency algorithms—Waltz filtering being a canonical example—was invented. These algorithms do not solve a CSP completely, but they eliminate, once and for all, local inconsistencies that cannot participate in any global solutions. These inconsistencies would otherwise have been repeatedly discovered in any backtrack solution. A clear treatment of these algorithms may be found in Mackworth [20] and in Mackworth and Freuder [21].

In the next section, we develop the notion of collapsed constraint satisfaction problems which can lead to great speedup for the line labelling problem.

12.2 Collapsed Constraint Satisfaction Problems

Consider the problem of finding a Huffman–Clowes labelling for the line drawing in figure 18. The edges are to be labelled from the set $\{+, -, \rightarrow, \leftarrow,\}$. Let us use the label nonconvex2 denoted by "!" for any label in the set $\{-, \rightarrow, \leftarrow,\}$. This gives us the collapsed label set shown in figure 19.

Consider the CSP associated with the new labelling problem. The size of each D_1 is now smaller, as shown in table 2. Consequently, we would expect to take less time for solving this *collapsed* CSP. One can actually make a stronger statement. A connected subgraph containing only arrow and Y-nodes can be labelled in exactly two ways, without *any* backtracking. Consider the subgraph ABCDEFGH is figure 18. As soon as any of the nodes in this graph are labelled (two ways), the rest of the nodes are labelled uniquely by constraint propagation. For example, if A is labelled as an arrow with the middle line a "+", B must be labelled as a Y-junction with all lines labelled "+". That implies in turn that D must be labelled as an arrow with the middle line "!" and so on until all the nodes ABCDEFGH are la-

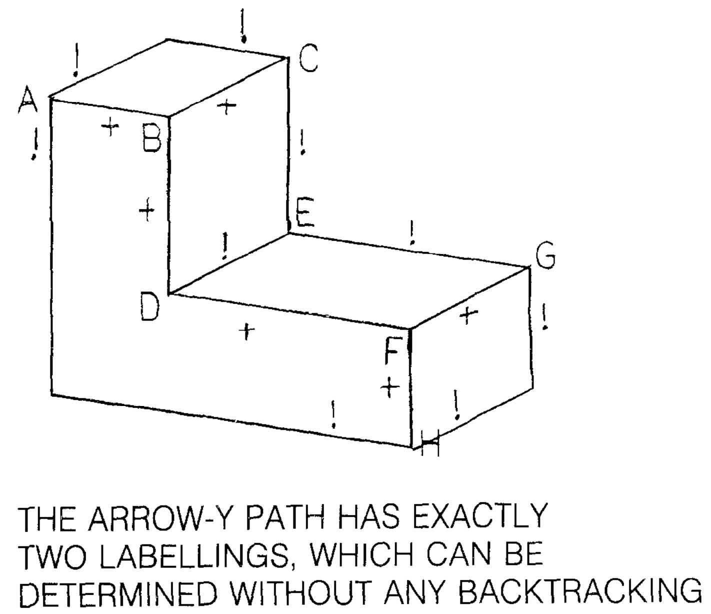
A
B
C
D
E
F
G
H
THE ARROW-Y PATH HAS EXACTLY TWO LABELLINGS, WHICH CAN BE DETERMINED WITHOUT ANY BACKTRACKING

Fig. 18. A test figure.

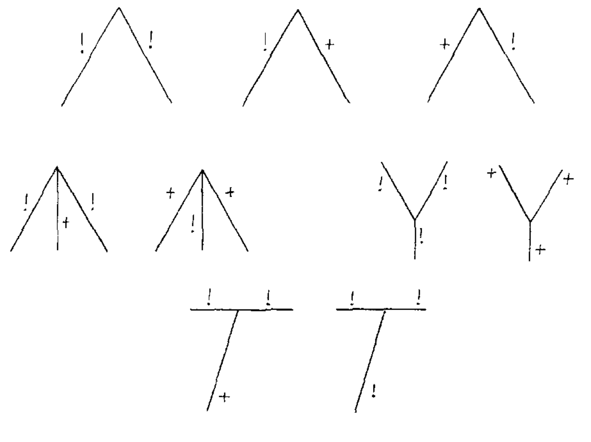

Fig. 19. A collapsed label set.

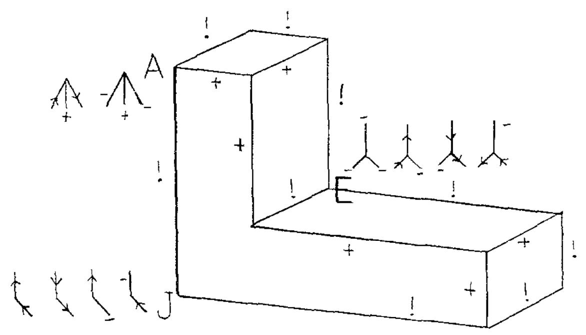

Fig. 20. Refining a coarse labelling.

belled. To see that this always works, we note that for, both arrow and Y-nodes, as soon as one of the arcs leading to it is labelled, the node is forced a unique label. The two labellings for an arrow–Y subgraph are related by a Necker flip. After labelling the arrow–Y subgraph, we try to label the nodes which are connected to it by an arc. The attempt to find a consistent labelling for these may lead to a unique determination of the labelling for the arrow–Y subgraph. In figure 18, this happens to be the case and we obtain a unique labelling for the figure without any backtrack search. We can now go back to our original problem—finding a labelling from the Huffman–Clowes set. Figure 20 shows the (now fewer) options for some nodes. At A there are two possibilities (originally three), at E there are now four options (as opposed to five), and at J there has been a reduction from six labelling possibilities to four. In fact in this figure the options at each node are strictly fewer, guaranteeing that the CSP is easier to solve.

We will now study this approach in a general

Table 2. Reduction in D_i.

Junction	D_i for H-C set	D_i for collapsed label set
L	6	3
Arrow	3	2
Y	5	2

setting. Let A be a constraint satisfaction problem (CSP) with the set of variables $(v_1, \ldots, v_n)$ with associated domains $(D_1, \ldots, D_n)$, unary predicates P_i and binary predicates P_{ij}. Construct a new constraint satisfaction problem A' with a set of variables $(v_1', \ldots, v_n')$ with associated domains $(D_1', \ldots D_n')$ such that for all i, D_i' is a partition[8] of D_i. The unary predicates P_i' and binary predicates P_{ij}' satisfy the following:

$$P_i'(v_i') \equiv \exists x.x \in v_i' \wedge P_i(x)$$
$$P_{ij}'(v_i', v_j') \equiv \exists xy.x \in v_i' \wedge y \in v_j' \wedge P_{ij}(x,y)$$

To verify that the collapsed version of Huffman–Clowes labelling described earlier is a collapsed CSP according to our definition, consider figure 21. The collapsed labels for the arrow are easily seen to correspond to a partition of the original Huffman–Clowes labels. This process can be repeated for the other kinds of nodes. To be more formal, the collapsing process defines equivalence classes of line labels and induced equivalence classes of junction labels. These equivalence classes partition each D_i.

It is easy to see that any solution of the original CSP A is mapped to a solution of the collapsed CSP A'. Note that more than one solution of A may be mapped to the same solution of A' and that it is possible that a solution of the collapsed CSP may not correspond to *any* solution of the original CSP.

At this point we would like to trace the history of the idea of using collapsed CSPs. Mackworth, in his PhD thesis, used a collapsed label set, although it was different from the one in figure 19.

[8]Recall that the partition π of a set X is a set of subsets of X such that
1. For each $x \in X$, x is an element of exactly one $S \in \pi$.
2. $\phi \notin \pi$.

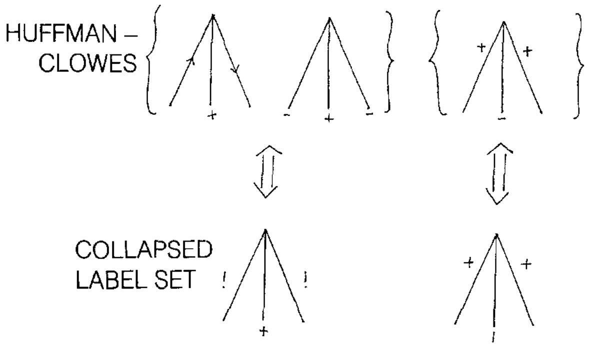

Fig. 21. Collapsing arrow labels.

He used the collapsed label "connect" for the set $\{+,-\}$ and "occlude" for the set $\{\leftarrow,\rightarrow\}$. We feel that our way of collapsing the label set is probably more useful in cutting down search (particularly noticeable for arrow–Y subgraphs).

Recently Mackworth, Mulder, and Havens [22] defined a hierarchical arc consistency algorithm HAC intended to exploit hierarchically structured domains (like in line labelling). Our work [23,24] was done independently. HAC is intended primarily for the first stage of solving a CSP—preprocessing by network consistency algorithms. Mackworth et al. analyze the complexity of their algorithm showing improved best and worst case performance when the domains obey certain restrictions. Our arguments, on the other hand, were merely heuristic.

Obviously, the idea of solving collapsed CSPs can be applied recursively. The four edge labels $\{+,-,\rightarrow,\leftarrow,\}$ could be regarded as a single label in an attempt to just label each line as either an edge or a limb. The collapsed label set for this is really simple. Except for the curvature-L and T-junctions, every junction has a unique label. At the next stage, one distinguishes between edges to be labelled + and !.

13 Dealing with Invisible Junctions

If there were no "junctions" like the one in figure

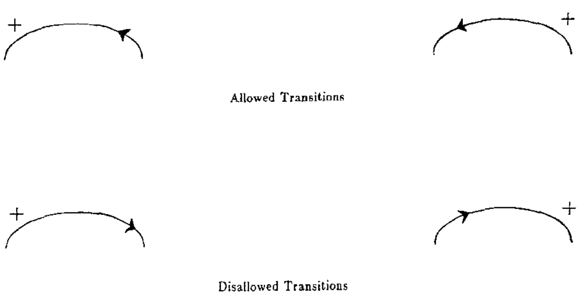

Fig. 22. Transition possibilities at an invisible junction.

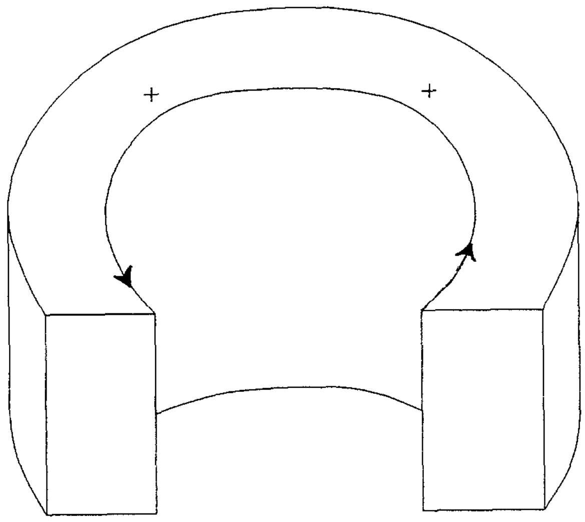

Fig. 23. An object with even number of transitions.

11, all lines would have the same label at each end. For the projection of a convex edge, the label could change from + to → or vice versa. On the occluding side of the junction, whether the label is → or ← is determined by the sign of the curvature of the line, as shown in figure 22. To understand the justification for this, the reader is referred to the derivation in section 9.

First we consider the case of image curves which have the same sign of curvature throughout. If there occur 0, 2, or any even number of such transitions, then the labels on both ends of the line are the same. Otherwise, they change as indicated in figure 22. Figure 23 shows an object with a curve for which the label changes twice. As we are trying to solve the sparse labelling problem, we are interested only in the labels at the junctions. One can model all the cases by introducing a phantom node with the labelling possibilities shown in figure 24. The junctions with concave and limb labels are included to handle correctly the case when a phantom node is introduced on a limb or concave edge.

Next consider the case of a curve like AD in figure 25 where the sign of the curvature changes at some interior points B and C. In this case we segment at B and C, and introduce one phantom node in each of the curve segments. The possible labels for each phantom node are as described earlier. At the knot points B and C, the line label is preserved. Figure 26 is an example of an object which can be labelled correctly by this procedure.

14 The Labelling Algorithm

If one is only interested in input–output behavior, a simple and yet functionally correct algorithm can be implemented by (a) introducing phantom nodes on each arc as described in section 13, followed by (b) backtrack search for all consistent labellings. If one is also interested in reducing the running time of the algorithm, more care is needed. In this section we will describe our algo-

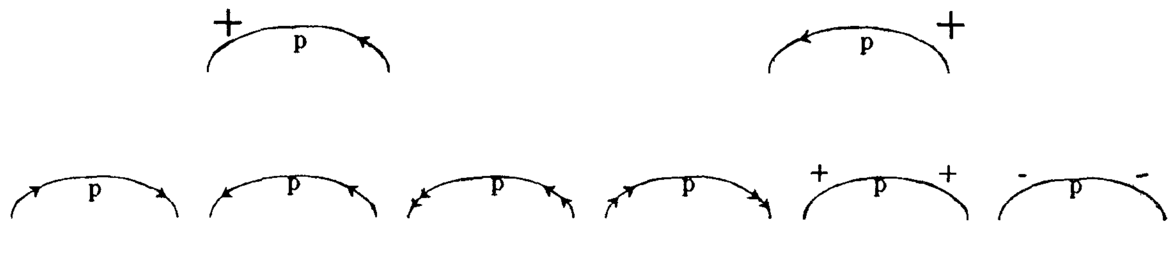

Fig. 24. Labelling possibilities for a phantom node.

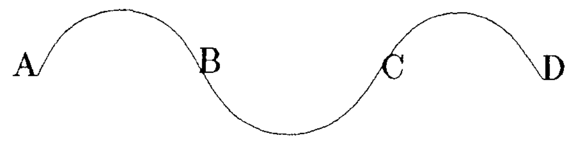

Fig. 25. Changing signs of curvature.

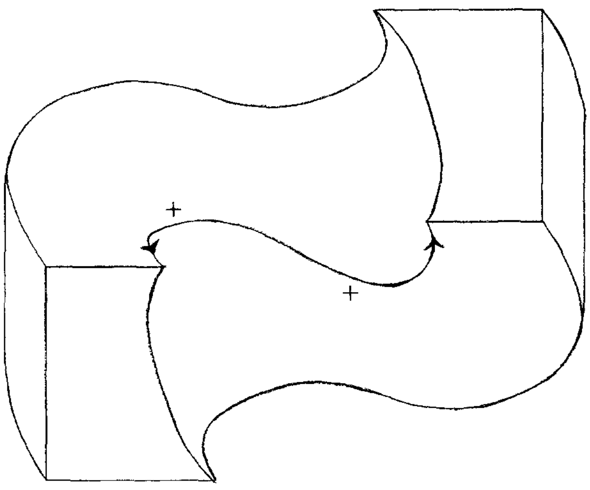

Fig. 26. An object where more than one phantom node is needed on a curve.

rithm which tries to reduce backtrack search as far as possible by exploiting the notion of collapsed label sets. We will illustrate the algorithm by tracing through it labelling the scene in figure 27.

1. Split the ISG at T-junctions and find connected components. Each component can then be labelled independently. In figure 27, there is only one connected component. In scenes with multiple objects, there would be several components with distinct objects belonging to distinct components.
2. Label the drawing with the label set {limb, edge}. For each of these labelling, perform steps 3–6 below. This produces the labelling at the top of figure 27.
3. Introduce phantom nodes on arcs corresponding to curved edges according to the procedure suggested in the previous section.[9] This is

[9]One additional trick: It can be shown that if the outer curve of a three-tangent junction has the same sign of curvature throughout, there can not be an invisible junction on it. As a consequence, no phantom node is introduced on AB.

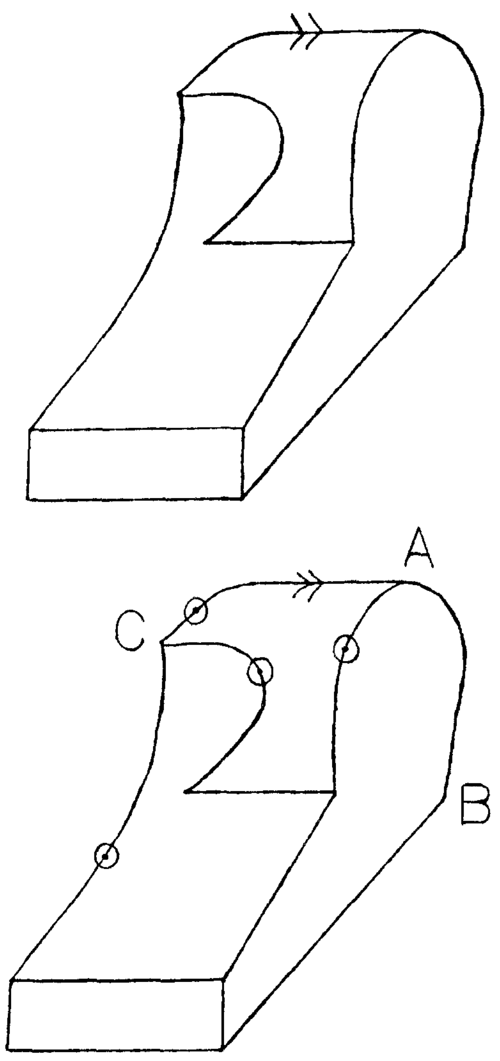

Fig. 27. Tracing through the algorithm.

shown in the bottom half of figure 27, where the symbol ⊙ is used to denote phantom nodes.

4. Label the edges with the label set {convex2, nonconvex2}. For each of these labellings perform steps 5–6. We get the two labellings in the top half of figure 28. They correspond to the two labellings for the arrow at C. The other arrow–Y path is labelled uniquely.
5. For each junction, find labelling possibilities consistent with the previously assigned coarse

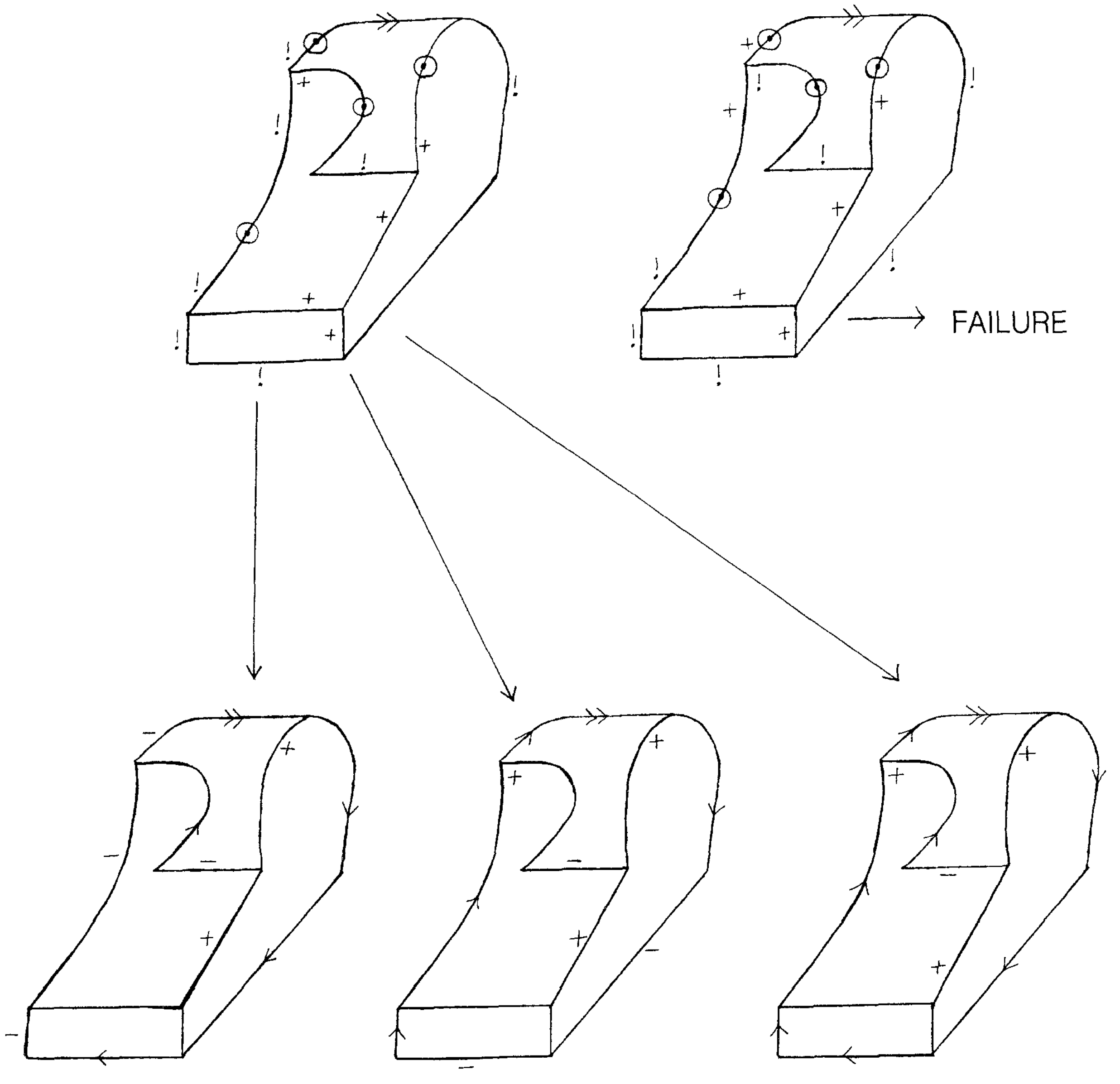

Fig. 28. Tracing through the algorithm contd.

labels. This is a straightforward process using the catalog in figure 17 augmented with the labelling possibilities for a phantom node from figure 24.

6. Perform a node and arc consistency filtering followed by a backtrack search to generate all labellings. Figure 28 shows the results. Left to right, the three labellings correspond to our intuitive interpretations—an object (a) lying on its side, (b) resting on its bottom face, and (c) floating in air.

The algorithm described above was implemented in Zetalisp on the Symbolics-3600. The implementation is fairly straightforward. A line drawing is represented as a set of points with their coordinates and a set of curves with appropriate directional and connectivity information. This information is read in from a file and the image structure graph constructed. In this process, each junction is classified appropriately as an arrow, three-tangent, etc. The behaviour of the program after this stage is aptly described by the algorithm stated above.

15 Some Typical Results

The computer implementation of the algorithm was tested on several hand-input line drawings. The objective of this was to seek answers to the following questions:

1. Do phantom nodes cause a combinatorial explosion in the number of labellings generated? One could argue that, since potentially for each curved edge the label could be different at the two ends, what used to be the main source of constraint propagation in polyhedral scene labelling is no longer available.
2. Is the "minimum number of surfaces at a vertex" rule (section 10.1) really useful? For junctions with three lines or less, this reduces to Huffman–Clowes labelling which gives good results where it is applicable. What about higher-order junctions? We would like the algorithm to come up with labellings which correspond to the interpretations found by human observers.
3. In the worst case, the time complexity of our algorithm is exponential. How good is it in practice? How useful is the idea of using collapsed label sets to cut down search?

Figures 29–34 give the labellings found for some scenes. Table 3 presents the results of these runs. For figure 31, which is a multiple-object scene, the object analyzed is the house. For figure 33, the object analyzed is the "pinched pyramid"—the figure which has an eight line junction in it. Recall that, for multiple object scenes, the algorithm labels each line drawing component independently, and hence it is only worthwhile to ask the questions mentioned above for single objects rather than whole scenes.

First, let us study whether the labellings found satisfy the desirable requirement of being few and corresponding to human intuition. The most typical ambiguity is that between concave edges

[10]This construction was pointed out to me by Christos Papadimitriou.

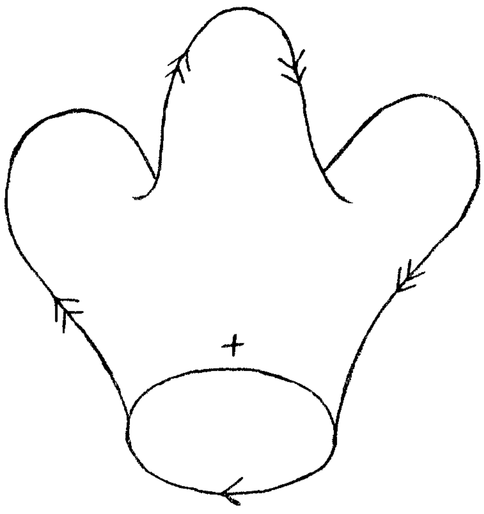

Fig. 29. Possible labellings of a curved object.

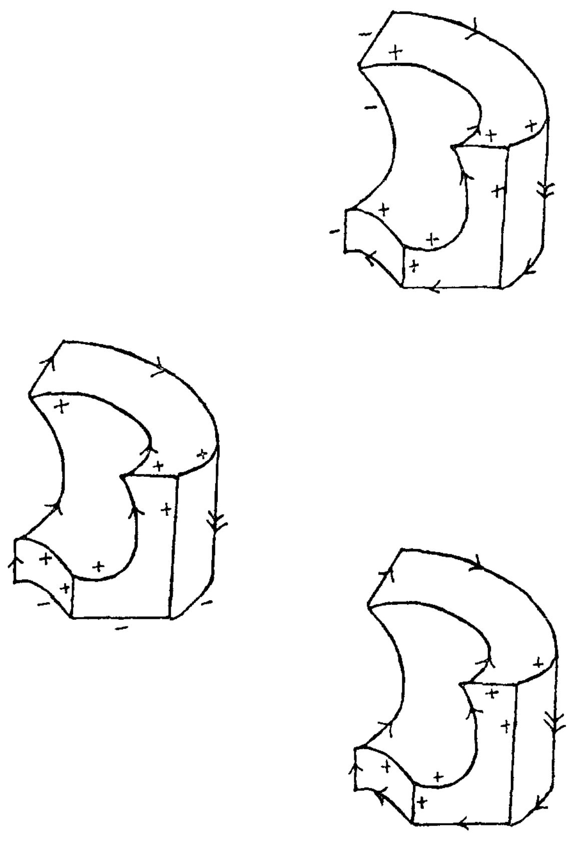

Fig. 30. Possible labellings of a curved object.

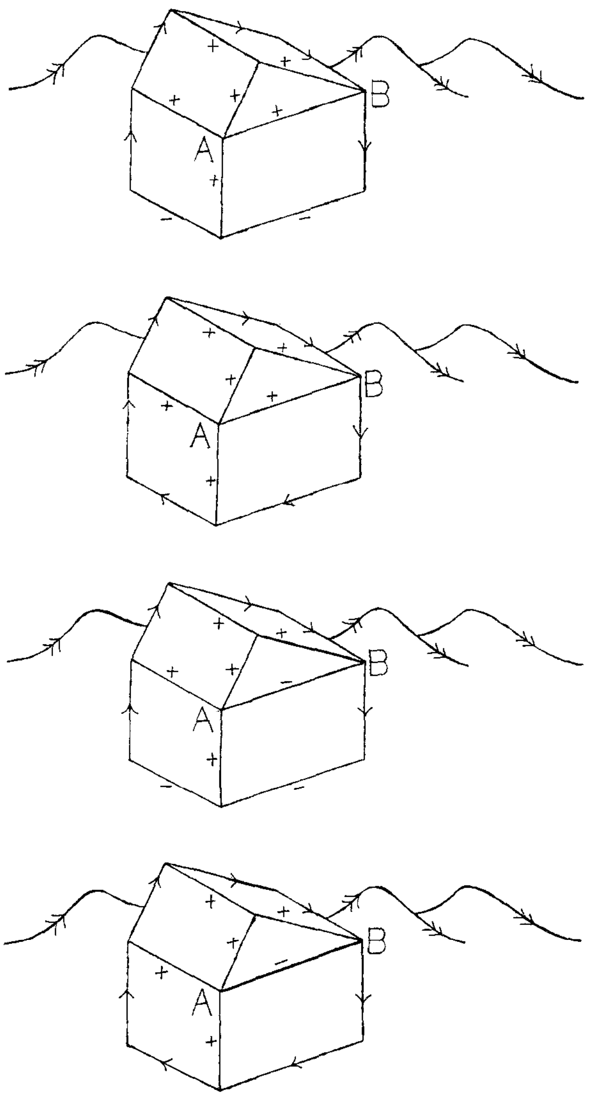

Fig. 31. Possible labellings of a curved object.

and convex occluding edges. A tetrahedron stuck to a wall, a tetrahedron stuck to the table, and a tetrahedron floating in air can give rise to the same line drawing but correspond to three different labellings. This is a fundamental ambiguity and cannot be resolved without appealing to support reasoning. The drawing in figure 30 is a good example. The house in figure 31 has four labellings—two which correspond to the house resting on the ground and two which correspond to the house floating in air. For each of these cases, the two labellings correspond to the edge AB being convex or concave—certainly a very intuitive result. Figure 32a and b are variations of the same object—in figure 32a the edges 1, 2, and 3 are straight and, in Figure 32b, they are curved. In both the figures, the left-most labelling is the intuitive one, but one can visualize physical realizations for the other interpretations. Figure 33 with its single labelling was a pleasant surprise. Figure 34 has the maximum number of labellings found—ten. Of these, the top six are just instances of the concave/occluding edge ambiguity. The bottom four are more unusual. They correspond to an invisible junction in the edge AB. The physical objects corresponding to these four labellings are quite counter-intuitive but constructible[10] nevertheless. We will try to help the reader visualize an object corresponding to the labelling L in figure 34. In figure 35, is shown a dotted line corresponding to the position of the invisible limb corresponding to the invisible junction on AB. Figure 36 shows what the object looks like when viewed from another direction—along the arrow in figure 35.

Now let us study the time complexity of the algorithm. This is done in table 3. The times given are in seconds for the Symbolics-3600 implementation. In the column called "useful coarse labels," the notation *n* of *m* means that of the *m* coarse labellings obtained at step 4 of the algorithm (section 14), *n* yielded a legal final labelling. There is a marked increase in time taken with the number of edges meeting at a junction in figures 32 and 33. Figure 32a and b differ only in that three of the stright lines in figure 32a become curved in figure 32b, resulting in the introduction of more pseudo-junctions by the algorithm. That adds three extra nodes and three extra arcs to the image structure graph with consequent increase in time complexity. Also to be noted is the usefulness/lack of usefulness of coarse labels. In figure 34, so many phantom nodes need to be introduced that there are no long arrow–Y subgraphs. Coarse labels prove pretty useless under these circumstances.

16 Performance Evaluation

We will now evaluate the labelling scheme with respect to its performance on a set of "self-evident" criteria.

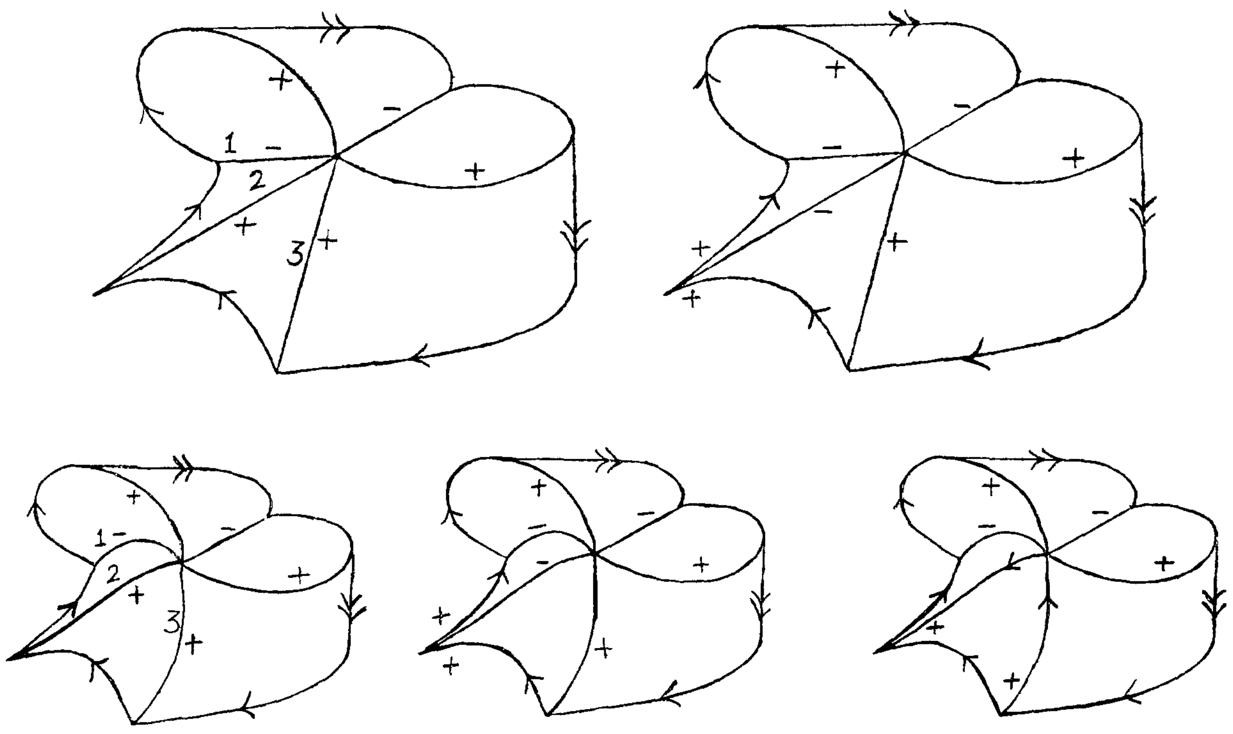

Fig. 32. Possible labellings of a curved object.

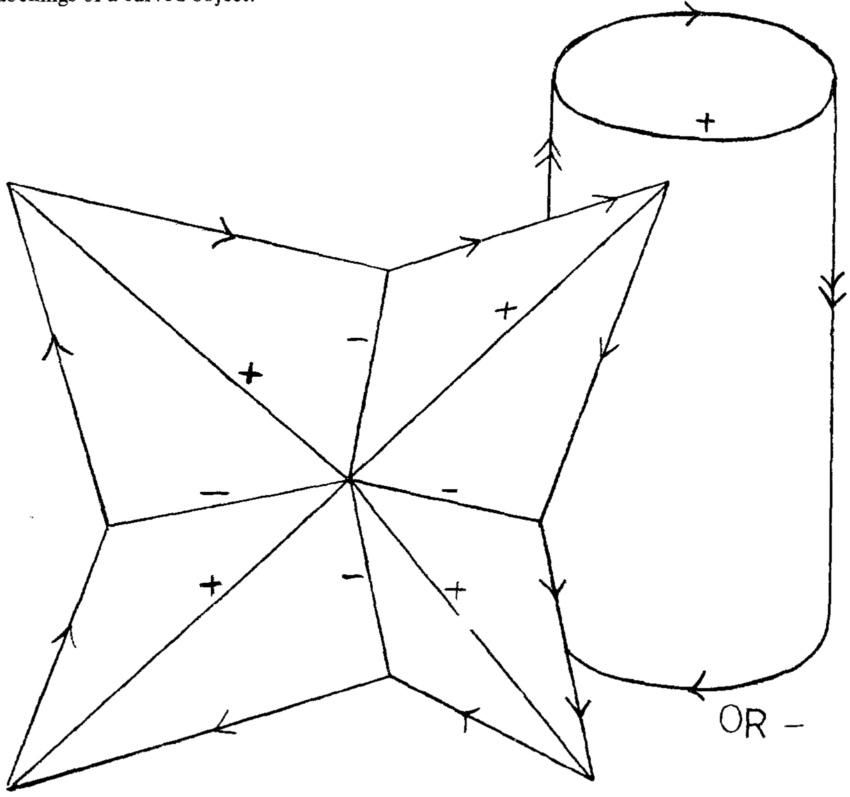

Fig. 33. Possible labellings of a curved object.

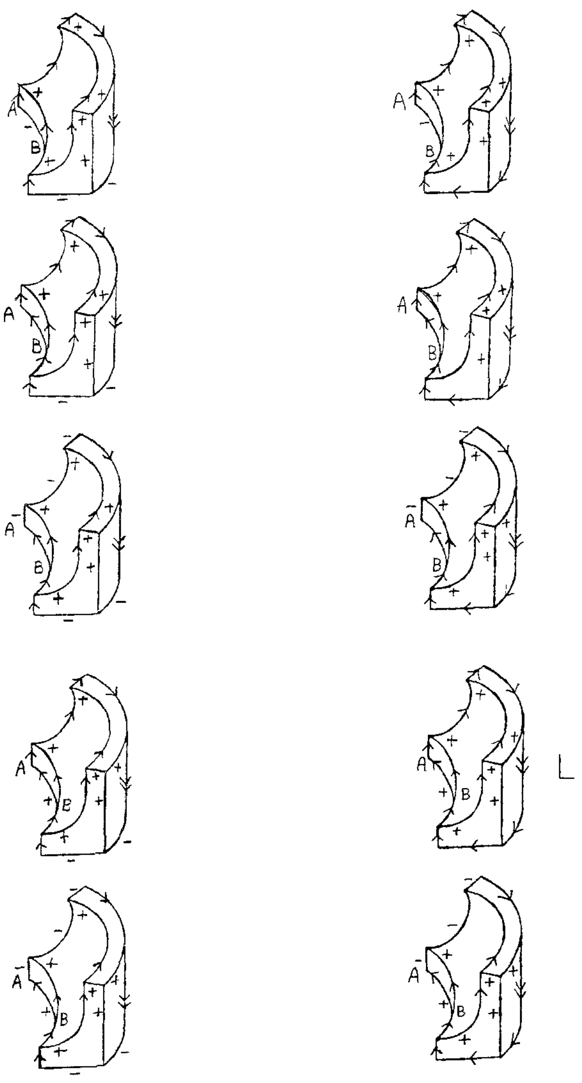

Fig. 34. Possible labellings of a curved object.

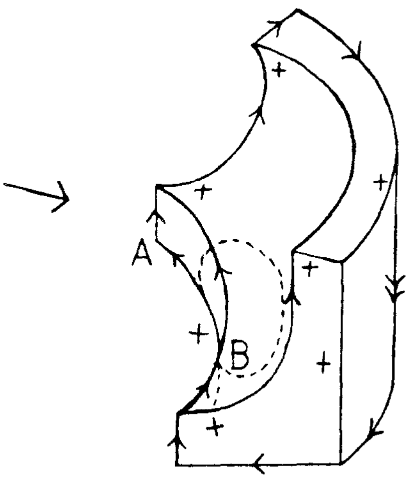

Fig. 35. A hard to visualize labelling.

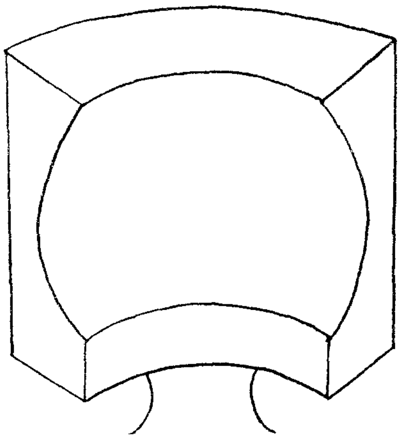

Fig. 36. The object in figure 35 viewed from another angle.

1. *We should be able to handle a broad range of objects—*n *faces meeting at a vertex with each face a portion of a general surface.* The class of objects that can be handled is defined in section 5. We will let the reader evaluate our success—an extended discussion may be found in Malik [24]. Two particular problem cases are interesting:
   a) *The apex of a cone:* Consider the labelling shown in figure 37. Our scheme regards this as an illegal labelling. If S_3 is a flat surface, and S_1,S_2 are conical cavities, this is a valid labelling. When conical "surfaces" are allowed, one can generate almost any labelling. Clearly this would be an unacceptable state of affairs. It is our opinion that conical "surfaces" may have to be handled by processing at some other level, using other line drawing cues or image intensity information. The problems arise only when the apex of the cone is included—conical surfaces minus the apex are acceptable.
   b) *General viewpoint:* The upper half of figure 38 shows line drawings of two objects viewed under general viewpoint and a sufficiently high-precision imaging situation. When the line drawing is obtained as the

Table 3. Results of some test runs.

Object	Nodes	Arcs	Useful coarse labels	Backtrack failures	Time	Labellings
Fig. 29	6	7	1 of 1	0	0	1
Fig. 30	11	15	1 of 8	2	0.13	3
Fig. 31	11	15	1 of 4	6	0.10	4
Fig. 32a	9	14	2 of 4	7	0.52	2
Fig. 32b	9	14	2 of 4	10	1.27	3
Fig. 33	11	18	1 of 4	1	1.87	1
Fig. 34	11	16	1 of 16	10	0.32	10

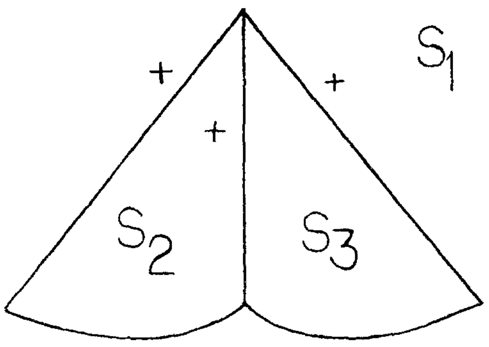

Fig. 37. Allowing conical "surfaces".

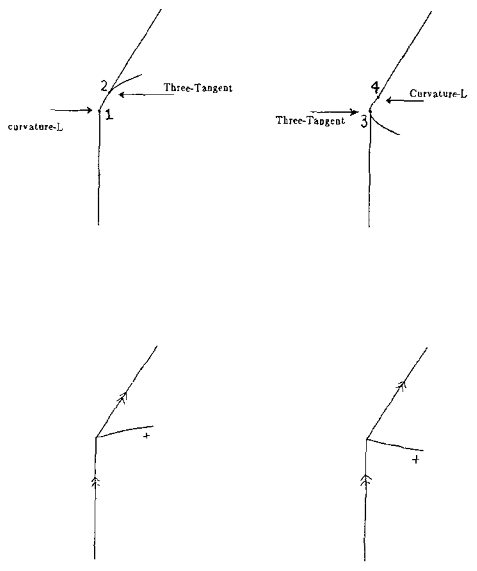

Fig. 38. Merging junctions.

output of a process with limited resolution, junctions 1 and 2 would get merged and so would junctions 3 and 4. This would result in the drawing in the lower half of figure 38. This drawing would not be labelled correctly by our scheme.

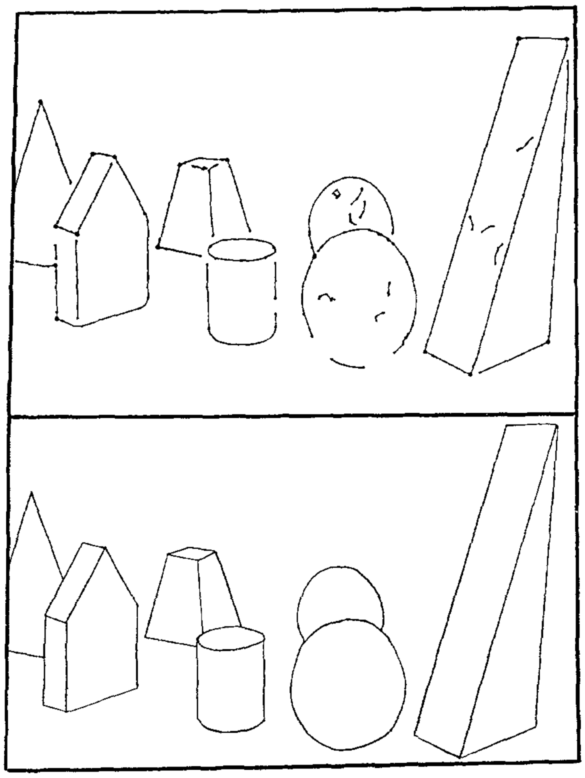

Fig. 39. Edge detector output compared with idealized line drawings.

To solve this problem in the context of known objects, one could suitably augment the junction catalog. This is clearly not a satisfactory solution for handling general scenes. Another approach would be to consider variants of the general viewpoint assumption. Our definition of general viewpoint was that the vantage point could move in a spherical neighborhood without changing the line drawing. One could relax this by allowing viewpoints confined to a plane. Alternatively, one could consider some numerical measure of stability of viewpoint. More work is needed on this problem.

2. *When given a line drawing of a legal scene, the scheme should find (a) all the legal labellings, and (b) only legal labellings.* Our construction of a junction catalog for the class of scenes defined in section 5 is as rigorous as the Huffman–Clowes construction for the trihedral world. We could therefore make the same claim—all the legal labellings will be found—except for one significant difference. Our scheme can find only the subset of legal labellings which correspond to a minimum number of faces interpretation at each vertex. It is not too difficult to modify the scheme so as to ignore this restriction. However, we feel that the resulting gain in geometric adequacy is not worth the loss of practical usefulness.

We conjecture that our scheme finds only legal labellings; i.e., corresponding to any sparse labelling found by our algorithm, some legal scene can be found. We do not have a rigorous result to prove this. However heuristic arguments and the results of all our test runs support this conjecture.

3. *The scheme should be robust with respect to errors in the input line drawing.* This is the major weakness of our algorithm at least in the basic form presented in section 14. Our algorithm was tested on hand-input line drawings as opposed to real edge data. To deal with the output of current edge detectors, the scheme should be able to cope with missing edges, spurious edges, missing junctions, improperly classified junctions, etc. Figure 39 illustrates this—the top half shows the output from an edge detection and linking scheme [27], and the bottom half shows what output we would like to have.

Certain kinds of junctions are more troublesome than others. To distinguish between an L-junction and a curvature-L junction is really asking for too much. We can adapt our algorithm to handle this in a fairly easy way. Instead of having entries for two kinds of junctions—the L and the curvature-L—consider a hybrid junction which has the labelling possibilities of both the L and the curvature-L (just take the union of the appropriate rows in figure 17). The use of this modified catalog would result in our getting more legal labellings for the drawing, compared with the scheme described in the rest of this thesis. Another source of error could be the misclassification of junctions. An arrow junction, a Y-junction, a three-tangent junction and a T-junction are discriminated by looking at the angles formed by the lines at the junction. With linking and curve fitting, a good edge detector should get the correct classification most of the time. If (by estimating the signal-to-noise ratio and some additional calculation) one can come up with numbers for the reliability of each junction classification, then one can use probability combination rules/fuzzy logic techniques to compute estimates of the reliability of various labellings of the line drawing.

The labelling algorithm developed in section 14 lacks robustness primarily because of the 100% confidence it assumes in the reliability of each junction. However, the junction catalog could be used as part of a more realistic algorithm, along the lines of earlier work [9,32] for polyhedral labelling. The ideas mentioned in the previous paragraphs could be incorporated into such an algorithm. Our junction catalog is small enough to permit the trial-and-error process inherent in a heuristic scheme. Needless to say, our statements here are mere statements of hope, until they are corroborated by an implementation tested on realistic input.

We would like to point out that, to appreciate the work in this paper, one has to make a distinction between two steps: (a) analyzing the geometric constraints and (b) developing an algorithm to exploit the constraints. In this paper, these two steps were performed in sections 7–10 and section 14, respectively. While the two steps are closely related, they are in fact logically distinct. Lack of robustness in the presence of noisy input is a serious, perhaps fatal, problem for the algorithm outlined in section 14; it is irrelevant as far as the analysis of geometric constraints is concerned.

Acknowledgments

I wish to thank Tom Binford and Christos Papadimitriou for many useful discussions during the development of the ideas in this paper. I also wish to thank them, along with Rod Brooks and Brian Wandell, for carefully reading through various drafts of my thesis. Their suggestions and criticisms clarified and improved the content and presentation of this work. The comments of the anonymous referees were also useful. Isha Ray drew most of the diagrams and was a great source of support throughout this work.

I was supported by an IBM doctoral fellowship during the academic years 1983–1985. Additional financial support for this work was provided by ARPA contract N00039-84-C-0211 and NASA contract MEA 80-19628.

References

1. H.G. Barrow and J.M. Tenenbaum, "Interpreting line drawings as three-dimensional surfaces," *Artificial Intelligence* **17**, pp. 75–116, 1981.
2. T.O. Binford, "Inferring surfaces from images," *Artificial Intelligence* **17**, pp. 205–244, 1981.
3. J.M. Brady and A. Yuille, "An extremum principle for shape from contour," in *Proceedings of IJCAI-8*, Karlsruhe, 1983, pp. 969–972.
4. I. Chakravarty, "A generalized line and junction labelling scheme with applications to scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1**, pp. 202–205, 1979.
5. M.B. Clowes, "On seeing things," *Artificial Intelligence* **2**, pp. 79–116, 1971.
6. H.S.M. Coxeter, *Regular Polytopes*, New York: Dover, 1973.
7. S.W. Draper, "Reasoning about depth in line-drawing interpretation," PhD thesis, Sussex University, 1980.
8. S.W. Draper, "The use of gradient and dual space in line-drawing interpretation," *Artificial Intelligence* **17**, pp. 461–508, 1981.
9. G. Falk, "Interpretation of imperfect line data as a three-dimensional scene," *Artificial Intelligence* **4**, pp. 101–144, 1972.
10. A. Guzman, "Computer recognition of three-dimensional objects in a scene," MIT Tech. Rept. MAC-TR-59, 1968.
11. D.A. Huffman, "Impossible objects as nonsense sentences," *Machine Intelligence* **6**, pp. 295–323, 1971.
12. D.A. Huffman, "Realizable configurations of lines in pictures of polyhedra," *Machine Intelligence* **8**, pp. 493–509, 1977.
13. T. Kanade, "A theory of the Origami World," Tech. Rept. CMU-CS-144, Computer Science Department, Carnegie Mellon University, 1978.
14. T. Kanade, "Recovery of the three-dimensional shape of an object from a single view," *Artificial Intelligence* **17**, pp. 409–460, 1981.
15. L. Kirousis and C.H. Papadimitriou, "The complexity of recognizing polyhedral scenes," Tech. Rept. STAN-CS-84-1015, Computer Science Department, Stanford University, 1984.
16. J.J. Koenderink and A.J. van Doorn, "The shape of smooth objects and the way contours end," *Perception* **11**, pp. 129–137, 1982.
17. J.J. Koenderink, "What does the occluding contour tell us about solid shape?" *Perception* **13**, pp. 321–330, 1984.
18. S.J. Lee, R.M. Haralick, and M.C. Zhang, "Understanding objects with curved surfaces from a single perspective view of boundaries," *Artificial Intelligence* **26**, pp. 145–169, 1985.
19. A.K. Mackworth, "Interpreting pictures of polyhedral scenes," *Artificial Intelligence* **4**, pp. 121–137, 1973.
20. A.K. Mackworth, "Consistency in networks of relations," *Artificial Intelligence* **8**, pp. 99–118, 1977.
21. A.K. Mackworth and E.C. Freuder, "The complexity of some polynomial network consistency algorithms for constraint satisfaction problems," *Artificial Intelligence* **25**, pp. 65–74, 1985.
22. A.K. Mackworth, J.A. Mulder, and W.S. Havens, "Hierarchical arc consistency: exploiting structured domains in constraint satisfaction problems," *Computational Intelligence* **1**, pp. 118–126, 1985.
23. J. Malik, "Labelling line drawings of curved objects," in *Proceedings of the ARPA Image Understanding Workshop*, Miami, 1985, pp. 209–218.
24. J. Malik, "Interpreting line drawings of curved objects," Tech. Rept. STAN-CS-86-1099, Computer Science Department, Stanford University, 1985.
25. D. Marr, *Vision*, San Francisco: W.H. Freeman, 1982.
26. R.S. Millman and G.D. Parker, *Elements of Differential Geometry*, New Jersey: Prentice-Hall, 1977.
27. V.S. Nalwa and E. Pauchon, "Edgel aggregation and edge description," in *Proceedings of the ARPA Image Understanding Workshop*, Miami, 1985, pp. 176–185.
28. J.M. Rubin and W.A. Richards, "Color vision and image intensities: When are changes material?" *Biological Cybernetics* **45**, pp. 215–226, 1982.
29. S. Shafer, "Using color to separate reflection components," Tech. Rept. TR 136, Computer Science Department, University of Rochester, 1984.
30. S. Shafer, T. Kanade, and J. Kender, "Gradient space in orthography and perspective," *Computer Vision, Graphics and Image Proceeding* **24**, pp. 182–189, 1983.
31. R. Shapira and H. Freeman, "Computer description of bodies bounded by quadric surfaces from a set of imper-

fect projection," *IEEE Transactions on Computers*, pp. 841–854, 1978.

32. Y. Shirai, "Recognition of man-made objects using edge cues," in *Computer Vision Systems*, A. Hanson and E. Riseman (eds), New York: Academic Press, 1978.
33. K.A. Stevens, "The visual interpretation of surface contours," *Artificial Intelligence* **17**, pp. 47–73, 1981.
34. K. Ikeuchi and B.K.P. Horn, "Numerical shape from shading and occluding boundaries," *Artificial Intelligence* **17**, pp. 141–185, 1981.
35. K. Sugihara, "Mathematical structure of line drawings of polyhedra—towards man-machine communication by means of line drawings," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **4**, pp. 458–469, 1982.
36. K.J. Turner, "Computer perception of curved objects using a television camera." PhD thesis, Edinburgh University, 1974.
37. D. Waltz, "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision*, P.H. Winston (ed.), New York: McGraw-Hill, 1975.
38. H. Whitney, "Singularities of mappings of Euclidean spaces. I: Mappings of the plane into the plane," *Ann. Math* **62**, pp. 374–410, 1955.
39. A.P. Witkin, "Intensity-based edge classification," in *Proceedings of AAAI-82*, Pittsburgh, 1982, pp. 36–41.