March 1983
Also numbered AIM-351

Report No. STAN-CS-83-961

(2)

Automated Stereo Perception

by

R. David Arnold

Department of Computer Science

Stanford University
Stanford, CA 94305

N00039-82-C-0250

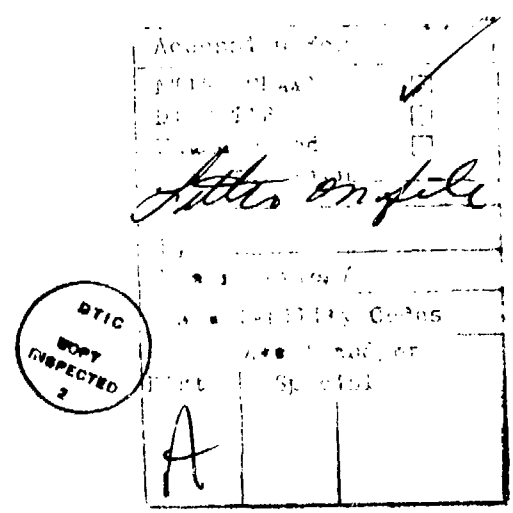
AUTOMATED STEREO PERCEPTION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By

R. David Arnold

March 1983



I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Thomas O. Binford

(Principal Adviser)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John McCarthy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Thomas M Cover

(Electrical Engineering)

Approved for the University Committee on Graduate Studies:

(Dean of Graduate Studies & Research)

Acknowledgments

This thesis represents the work of more than just the author; many people have contributed programs, ideas and encouragement over a number of years. Hans Moravec, Don Gennery and Dave Lowe have supplied programs for processing images and graphics. Sid Liebes, Harlyn Baker and Tom Binford formed the core of a "stereo group" at Stanford that spent many hours discussing the stereo problem and the various approaches we were all trying. Tom Binford, my advisor, has been a constant source of ideas and guidance. I would like to thank these and all the other people at Stanford who have made my work there enjoyable. Finally, I would like to thank Kathy, my wife, whose patience and support have made this thesis possible.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 The Problem Domain

This research aims to advance the science of computer stereo vision – reconstruction of 3-dimensional scenes from 2-dimensional images. The problem is to establish a correspondence between features or regions in two or more images from which we can calculate positions in 3-space. In ~~our~~ research, ~~we~~ choose to match features, rather than use the traditional methods of area correlation. The features ~~we~~ use are extended edges, or more precisely, the image curves which are the projections of edges in the scene. ~~We assume~~ a preprocessing stage which can extract such edges by first applying an edge operator and then linking edge elements into extended image curves. The choice of feature-based stereo over area-based stereo offers advantages in speed and accuracy, as well as avoiding some fundamental problems.

In an edge-based system, computation effort can be concentrated on the edges. Depth information about surfaces can be inferred from surface boundaries, which are visible as edges. If high speed, specialized processors are used for edge operators [Nudd 1977], overall computation can be cut significantly.

Typically, edge-based techniques offer a factor of 10 improvement in accuracy over correlation methods. In correlation, accuracy near a boundary is limited to a fraction of the width of the correlation window (typically 8x8). An edge operator, however, can provide measurements to a fraction of a pixel. Edge-based systems also have an advantage with small objects whose total size is smaller than a correlation window. Similarly, long, thin objects such as poles are prominent features, but are too small for correlation windows.

Area correlation systems depend ultimately on image intensity, which can be affected by several factors. Film and camera sensitivity may vary over an image or from one image to the next and scene illumination may change for images taken at different times. The reflectivity of objects may depend strongly on viewer position, as in specular reflections. Many correlators automatically compensate for constant gain and bias differences in the images, but edge position and orientation are much more stable than photometric quantities because the conditions listed above will not significantly affect them.

A serious deficiency of area correlation is failure at surface discontinuities. Simple area correlation techniques inherently fail in the vicinity of surface discontinuities because the edge of an object appears against a different background area in each view of the stereo pair. It is important to locate surface discontinuities, since it is precisely the boundaries of objects where accurate measurements are most important. Surface discontinuities are typical of most scenes containing man-made objects such as buildings and vehicles.

Fine textures and smooth surface slopes are typical of natural surfaces such as rock, grass and vegetation. In such regions, area correlation can be quite effective. On the other hand, regions of totally uniform intensity provide no signal for an area correlator, and the only hope is to locate the boundaries and interpolate the interior.

Stereo vision systems have applications in mapping, aircraft and missile guidance, autonomous robot vehicles [Moravec 1980], planetary exploration [Gennery 1980], and industrial inspection and assembly. Some applications favor area-based systems and some favor edge-based systems. Thus, edge-based and area-correlation approaches are in a sense complementary. A general purpose stereo vision system should include both.

1.2 Results and Contributions

This thesis approaches the stereo problem in two steps. We first use epipolar geometry to reduce the problem to a one-dimensional matching. Given the geometry of the two cameras, a point in one image may be projected to a line in an image from a different viewpoint. Any corresponding point must lie on that ***epipolar line***. We then demonstrate the use of edge continuity and context in combining matches along adjacent epipolar lines to produce a match over an entire image.

We derive a series of important geometric constraints for matching edges in the one-dimensional problem. However, edges are not matched in isolation – they must fit a global interpretation. Occlusion constraints require an explanation for each occluded edge or surface and ensure a consistency across the whole epipolar line. If an edge is occluded, there must be a surface in a position to block the view of one camera. Conversely, if an edge is not occluded, there must be no surface blocking the view of either camera. We have modified a dynamic programming algorithm, the Viterbi algorithm, to incorporate these constraints and the special conditions of stereo matching. The algorithm determines the highest scoring one-dimensional match that satisfies these occlusion constraints.

We have derived two analytic results concerning constraints on ***interval length*** and ***edge angle*** for stereo matching [Arnold 1980]. The ***interval length***, or distance between adjacent edges on an epipolar line, is a function of surface orientation. The projected dimensions of a surface will vary in two views according to the orientation of that surface. Similarly, edge orientation in the scene determines the projection of different ***edge angles*** in the two views.

These results allow a distribution function in the object space to be translated to a distribution function in the image space. In the simplest case, we can assume edges and surfaces to be uniformly distributed over all orientations in the object space. We can then calculate the likelihood that an arbitrary pair of edges or intervals from the two

images correspond. The functions are sharply peaked even for the 60 degree vergence angles used in aerial photography. When baselines corresponding to human vision are used, the conditions are extremely strong. We have used the results of these functions as components of an evaluation function for stereo matching.

While the usual purpose of the Viterbi algorithm is to find an optimal match, it is a good strategy not to discard options too early. A globally optimal match may be suboptimal in the limited context of a single epipolar line. It is an advantage to keep a list of several of the best matches of each line to be filtered later by two-dimensional consistency relations. For this reason we have developed a significant extension to the Viterbi algorithm that produces a list of all matches scoring within a preselected range of the optimal match. This list is then filtered by an iterative process that enforces consistency among adjacent epipolar lines.

In 1978 we introduced an edge-based stereo system that used the concept of edge continuity and context to reduce ambiguity [Arnold 1978]. Edge matches based on simple local measures such as contrast and angle were filtered by requiring matched edges to be continuous in 3-space. If an edge extended continuously in one view, its match was required to have a continuous extension in the other. This system used unlinked edge elements (edgels), and succeeded in correctly correlating about 90% of the edgels in an image.

Our most recent system operates on linked edgels, or extended edges, and makes use of more powerful techniques to do the one-dimensional matching. It then applies the constraint of edge continuity iteratively with the epipolar matching to derive a globally consistent match.

The principal contributions of this research are the first use of edge continuity in the context of adjacent epipolar lines for determining matches; the use of occlusion constraints; the analytical functions for *interval* and *angle* constraints; and the modified Viterbi algorithm that includes suboptimal matches.

1.3 State of the Art

This section briefly describes some recent work on stereo vision systems by others that have influenced this thesis. Some of these systems use techniques that we have adopted in our work, while others provide interesting alternatives. This survey concentrates on *feature-based* stereo.

Moravec's vision system [Moravec 1980] is the input to a navigation and obstacle avoidance system for a computer controlled-vehicle. The stereo correlation in this system is area-based, but the initial correlation is driven by a collection of feature points resulting from an *interest operator*. The *interest operator* selects points with a locally maximal value of an interest measure. The interest measure is the minimal directional variance taken in four directions over a small square window. Thus "interesting" points are those whose position is easy to determine in more than one direction (e.g., intersecting edges). The points from the interest operator are matched with a binary correlator that uses an iterative technique with increasing resolution to narrow the search at each step. Stereo images are taken from nine camera locations along a common baseline, and correlations from all possible pairs of images are combined to determine the final depth map.

Gennery's stereo system [Gennery 1980] is also designed to provide input to an autonomous vehicle. This system uses Moravec's *interest operator* and *binary correlator* as inputs to a camera model solver and ground plane finder. With an accurate camera model, the system then applies a high resolution (area-based) correlator capable of subpixel positioning. Obstacles are defined relative to the ground plane.

Control Data Corporation has developed a *Broken Segment Matcher* [Henderson 1979] that is designed to produce structural models of buildings and other cultural scenes from aerial imagery. Their approach combined edge- and area-based techniques, with edges serving to bound regions in which correlation is based on image intensities. The images are transformed to make use of a simplified epipolar geometry, and one epipolar

line pair is 'seeded' by providing a manual matching of edges crossing that line pair. All matching is done in one dimension, along epipolar line pairs proceeding outward from the initial line. Edge match information is propagated from line to line and edited automatically as some edges end and others begin. In a later version, the system assumes that scenes are composed of rectilinear structures; surfaces must have one of three orthogonal orientations, and all edges are straight.

More recently, Control Data Corporation has developed algorithms for stereo matching that employ a ***structural syntax*** for symbolic matching of geometric units [Panton 1981]. This system works from line drawings, and matches edges or figures composed of edges. Knowledge of scene geometry is built into the algorithm or entered by hand and serves to filter ambiguous edge matches. Scenes are restricted to right parallelepipeds (simulated urban structures) and matching is restricted to the horizontal tops of these objects (roofs). The geometric knowledge used includes clustering of parallel lines on opposing figure boundaries, known allowable edge orientation (vanishing points entered manually) and ***a priori*** limits on stereo disparity (based on building heights).

Researchers at MIT have developed a computational algorithm for human stereo vision [Marr 1977] which has been implemented by Grimson [Grimson 1980]. This system convolves the image with spatial frequency filters (an edge operator), and bases its matching on the zero crossings of these filters, together with contrast and edge orientation estimates. The filters used have varying resolution, and matching proceeds generally from low to high frequency. An initial ***vergence*** or disparity is set manually, and the low frequency filter output is used to drive fine adjustments to this ***vergence*** until a match is achieved with the high frequency filters over a significant local region. The depth information from each region of correspondence is then interpolated and smoothed into a full depth map.

Baker's stereo system [Baker 1981] combines several of the techniques used by earlier systems and uses both feature- and intensity-based matching. This system begins

with low resolution images and matches linked edges to get a rough disparity limit for subsequent higher resolution matchings. The matching process uses a Viterbi dynamic programming algorithm applied to individual epipolar line pairs. It maximizes a metric based on local edge properties including contrast and angle, and uses the edge angle and edge interval measures described in this thesis. Some edges are allowed to remain uninterpreted in this step. A cooperative process then removes edge correspondences that violate a three-dimensional continuity constraint across epipolar lines. Another edge matching process is applied to attempt to match unassigned edges bounded by pairs of matched edges. A final dynamic programming process matches intensity data bounded by matched edges and results in a full disparity map of the image pair.

1.4 Other references

This research touches on many disciplines; for the reader who wishes to pursue some of these further or to develop a background for reading this thesis, we provide a short list of references.

Many of the images processed by stereo vision programs originate in aerial photography, where stereo images have been used for years in making topographic maps. The textbook by Burnside [Burnside 1979] provides an introduction to the main theoretical elements of photogrammetry, while the more massive reference from the American Society of Photogrammetry [Slama 1980] covers the subject in more detail. Both books cover the geometry of aerial photographs, from the principles of central perspective projection to corrections for typical aircraft alignment and tilt problems. They also include information that is of practical use to a researcher seeking images from an aerial survey company. For a theoretical treatment of perspective transformations and coordinate systems, an introductory text in projective geometry such as Wylie [Wylie 1970] is recommended.

A central algorithm in this thesis is the Viterbi algorithm, which is one result from a field of research called dynamic programming. The first complete text in this area was by Bellman [Bellman 1957]. Dynamic programming has since become a well established discipline with many textbooks following [White 1969, White 1978, Viterbi 1979, Denardo 1982]. The first published account of the Viterbi algorithm was in 1967 [Viterbi 1967] as a decoding algorithm for convolutional codes, but the algorithm has since been used in a variety of applications. Forney [Forney 1973] gives a good tutorial and survey.

The psychology of human stereo vision is an interesting area because the phenomena can be personally experienced. Many experiments and unusual examples of stereo effects are described by Julesz [Julesz 1971, Julesz 1975]. His experiments in random dot stereograms have been used as test cases for computer stereo programs.

Computer vision is a much m re recently developed discipline, and until recently there have been few books on the subject. Ballard and Brown [Ballard 1982] have just published a comprehensive book on computer vision that is designed as a textbook and provides a good survey of this field. David Marr [Marr 1982] has taken a different approach and believes the "overall goal is to understand vision completely". Marr presents his group's research efforts to model *human* vision computationally. Both books provide good bibliographies.

Chapter 2

THEORY

2.1 Profiles

This section introduces the subproblem of one-dimensional stereo matching. We describe the geometry and develop a notation that will be used later in the presentation of the dynamic programming algorithm.

2.1.1 – Geometry

We will use the stereo camera geometry of Figure 2-1. The projective center of the left camera is the origin and the projective center of the right camera lies on the x axis. The baseline, B, is the distance between the projective centers. The two image planes are coplanar and are perpendicular to the z axis. The image distance, f, is the distance from the projective center to the image plane, and is the same in both cameras. This normal camera model is for side-by-side cameras.

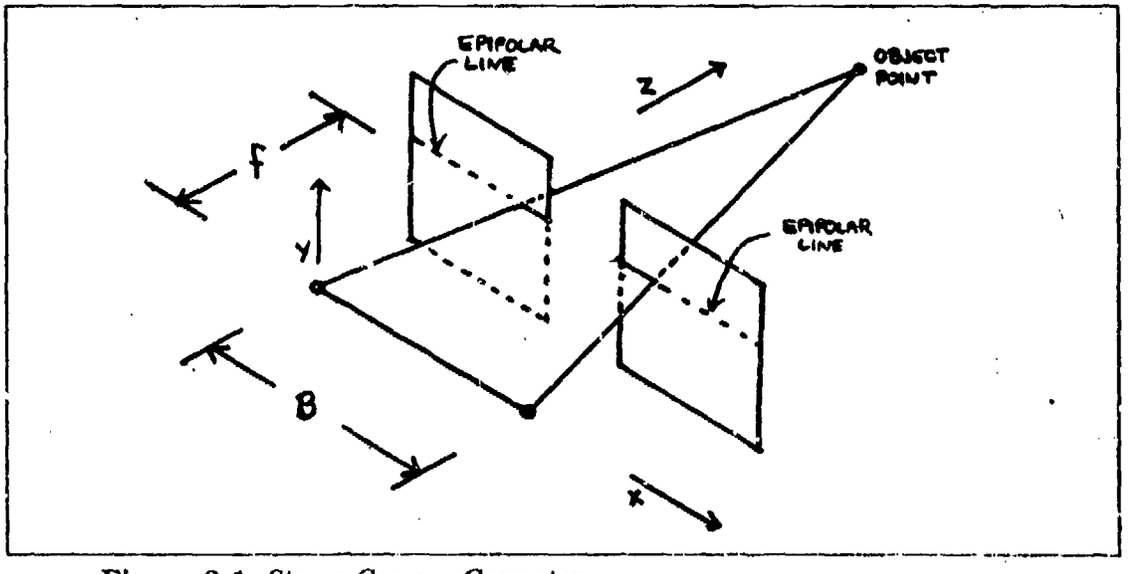


Figure 2-1: *Stereo Camera Geometry.*

If an actual stereo image pair were taken with a slightly different geometry, we could use a few simple transformations on the images to produce a pair consistent with this model. Alternatively, we could access the images through a coordinate transform that accounted for these differences. Gennery describes a camera model program that can automatically compute such a transform if given a collection of corresponding stereo points [Gennery 1980], and has separately produced a version specifically for the camera model we present here.

This model corresponds to the usual examples of stereo, such as human vision, where the baseline is roughly perpendicular to the line of sight. It is also a good model for aerial photography where a single camera is used; the line of sight is perpendicular to the flight path, which determines the baseline. It does not cover the case where the line of sight and the baseline are approximately collinear. This could arise from a moving vehicle taking successive pictures looking forward along its path (see [Moravec 1980]). Such geometry is a degenerate case of our model. Our camera model is also not suited to panoramic sensors where there is no projective center (e.g., the stereo cameras used in the Viking Lander (see [Liebes 1977]).

Given an arbitrary camera orientation and a point on any object visible to both cameras, we can define an ***epipolar plane*** as that plane determined by the object point and the two projective centers. The epipolar plane intersects the image planes, defining an ***epipolar line*** in each. It follows by projection that the image of the object point must lie on the epipolar line. Furthermore, the image of the same object point from the other view must lie on the corresponding epipolar line. This reduces the correspondence problem to one dimension. In our normal camera model, epipolar lines are always parallel to the x axis, thus corresponding image points must have the same y-coordinate in both left and right images.

In the discussion that follows, we will be concerned with matching features in a given left epipolar line with features in the corresponding right epipolar line. The

features of interest are scene edges, which project as curves in the image planes. For the one dimensional case, we are interested in the points at which image curves intersect the epipolar line. These intersection points serve to segment the epipolar line into intervals, which are ordered by their occurrence in the image from left to right. The matching problem is one of mapping between a sequence of intervals or intersection points from one view and a similar sequence in the other.

Consider a pair of corresponding epipolar lines from a stereo image pair. We locate intersection points of image curves with the epipolar lines and want to match these points to reconstruct the original scene. If we back-project these points we get for each view a set of rays from the camera's projective center, through the intersection points. Every ray from each image lies in the epipolar plane. If we view this plane from the side, we see that the rays from the left and right cameras intersect to form a grid or lattice. This lattice is bounded by a region obtained by projecting rays through the image boundaries. We will refer to this region as the *stereo zone*, for only objects within this zone can be seen in stereo. (See Figure 2-2a). In general, the image boundaries need not be symmetric with respect to the projective center. Although this is true for most cameras, it is common to digitize the film off-center in order to improve the stereo overlap (see Figure 2-2b). Thus, the stereo zone may extend to infinity or may be a closed quadrilateral, depending on how the image boundaries are defined in the film plane.

Each lattice point corresponds to a potential match between a feature in the left and a feature in the right. If such a match were correct, then the object must have been at the point in space represented by that lattice point. We will use this lattice as our coordinate system and attempt to reconstruct the original scene within it. The two axes of the grid are labeled with the left and right feature sequences. The reconstructed surfaces of the scene, or rather their intersection with the epipolar plane, will be called a *profile*.

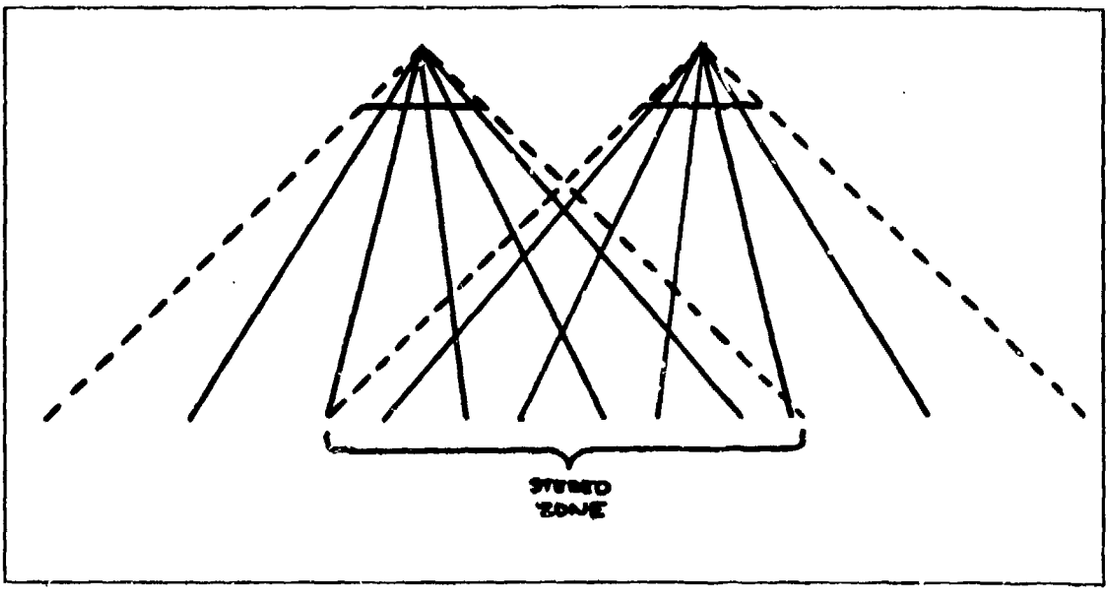


Figure 2-2a: *The Stereo Zone is determined by the camera geometry and image boundaries.*

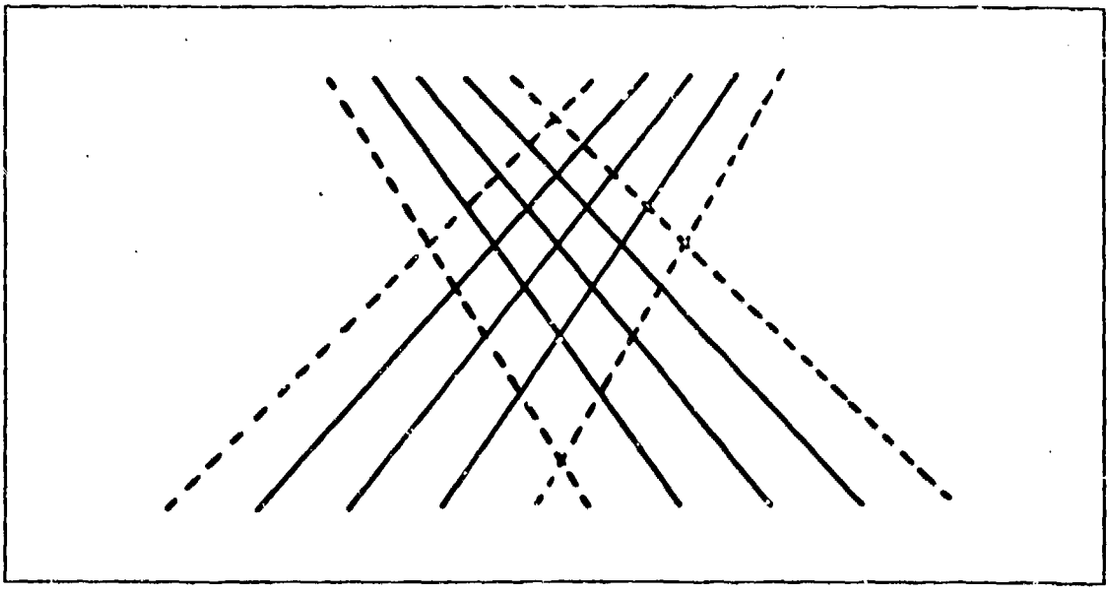


Figure 2-2b: *Image boundaries may be restricted for better overlap and a finite Stereo Zone.*

2.1.2 – Basic Assumptions

We now make some limiting assumptions in order to precisely define the problem we intend to solve. First, we make the general assumption that the scene is independent of the viewer. While the stereo camera model and the objects in the scene may be aligned with respect to a common reference such as gravity, individual features in the scene should have no dependence on camera angle. That is, small shifts in camera position should not cause significant changes in the image.

We assume that surfaces are bounded by visible edges. That is, if a surface or slope discontinuity exists, it will produce a curve in the image which will be detected by our edge operator. This ensures that the reconstructed profile will have all its boundaries at lattice points. There may also be edge curves in the image that do not result from discontinuities but from surface markings. Note that this does not mean that every feature in one view must match some feature in the other. Occlusion by intervening surfaces can block features from one or both cameras. We merely require that a feature is detected if and only if it is not occluded. This assumption is equivalent to perfect edge detection; performance with imperfect data is discussed in a later section.

We assume that the profile consists of straight line segments. A sufficient condition for this is that in the original scene all surfaces are planes. This restriction is not a severe one in cultural scenes, where man-made surfaces usually are planar. In fact, the interpretation of a curved surface as flat may still allow an accurate estimation of its boundary (see Figure 2-3). However, with curved surfaces, tangent discontinuities are possible (see Figure 2-4). Since such features are dependent on camera position, it is possible for a surface boundary in one view to have no counterpart in the other view.

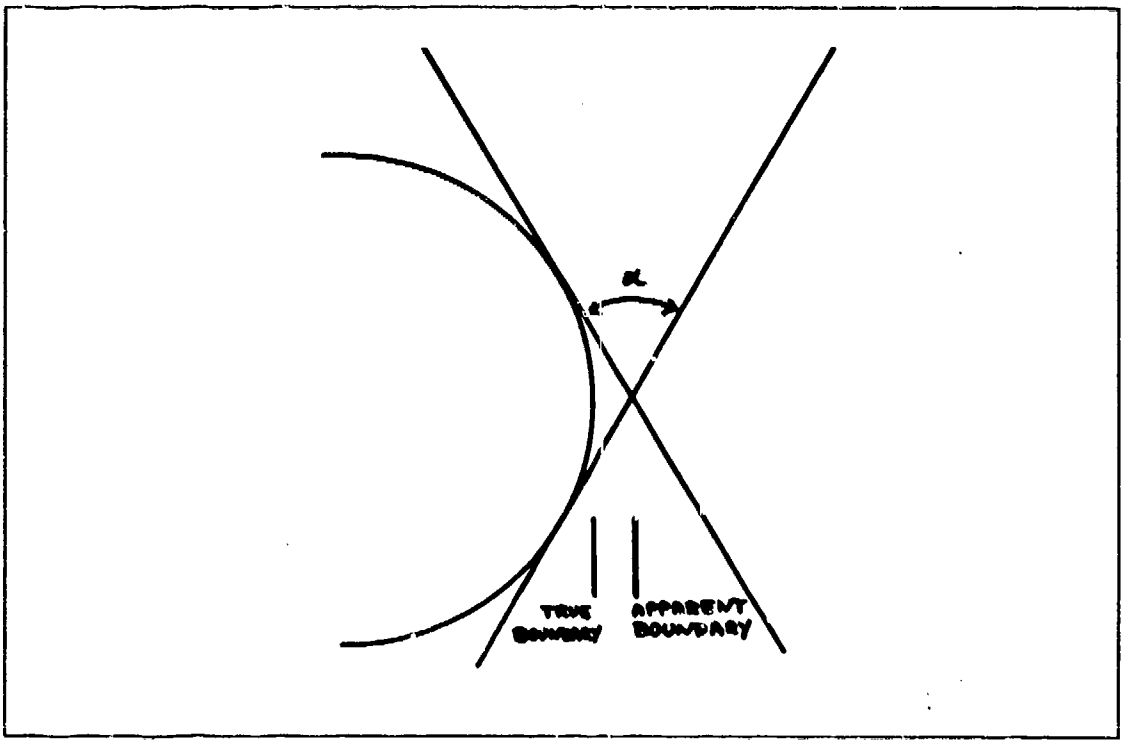


Figure 2-3: *The distance, d, between the true boundary and the apparent boundary of a circle in the epipolar plane is small for most vergence angles. Its value may be calculated from* $d = R(\sec(\alpha/2) - 1)$ *where* R *is the radius of the circle and* α *is the stereo vergence angle. For angles of* 60°, 12° *and* 4°, *the errors are* $.15R$, $.006R$ *and* $.0006R$, *respectively.*

Finally, it will be a necessary condition of the dynamic programming methods to be introduced later, that the two image sequences match monotonically:

> Let a_l and b_l be elements of the left sequence and a_r and b_r be elements of the right.
>
> Assume a_l matches a_r and b_l matches b_r.
>
> If a_l occurs to the left of b_l then a_r must occur to the left of b_r.

In other words, there can be no *order reversal* in mapping one sequence to the other.

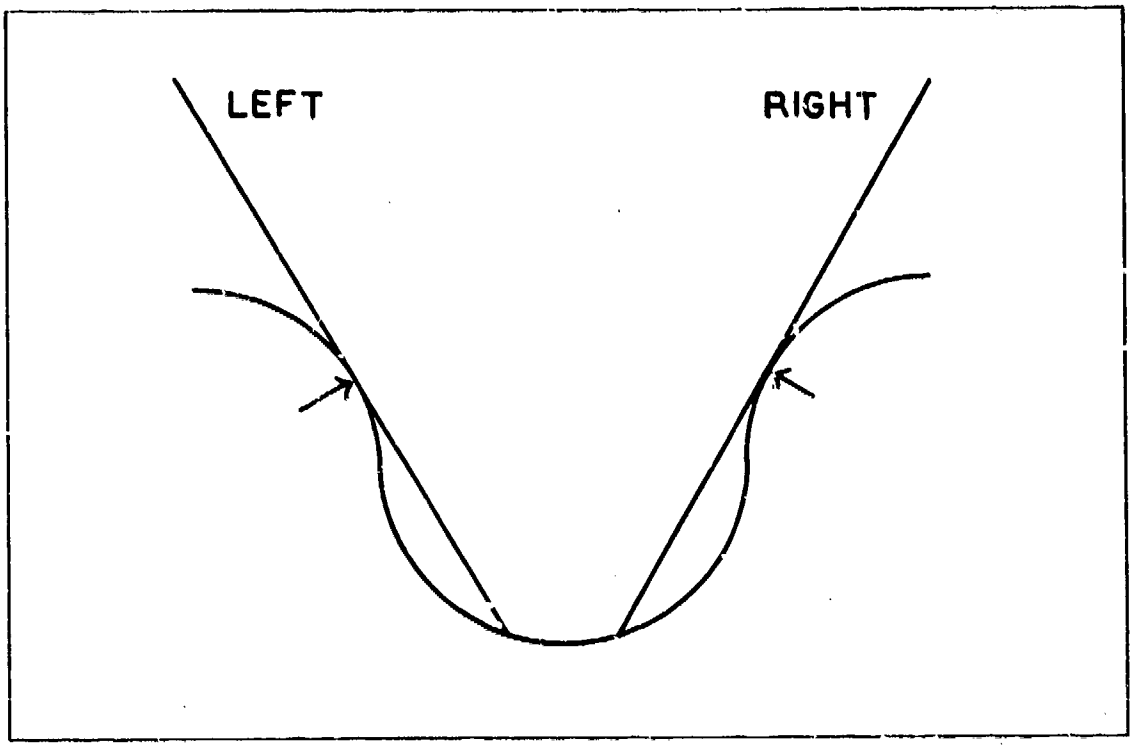


Figure 2-4: *Tangent discontinuities are viewer-dependent "features" that have no stereo correspondence even though there may be no occlusion.*

Order reversals may occur whenever there is an *overhang*, which we define as a disconnected profile. Figure 2-5 illustrates the classic case, where an overhanging surface is far enough above the background surface that the cameras can "see under it". Not all overhangs produce order reversals. It is necessary for there to be a feature within the wedge-shaped zone (cross-hatched area in Figure 2-5) beneath the overhanging surface. The geometry of this zone depends on the camera model and the width and altitude of the upper surface. Overhangs usually result from something like a wire stretched across the scene, or from oblique views of thin objects. Figure 2-6 shows how an aerial view of a building can cause an overhang. While overhangs are common in many scenes, they do not usually result in order reversals.

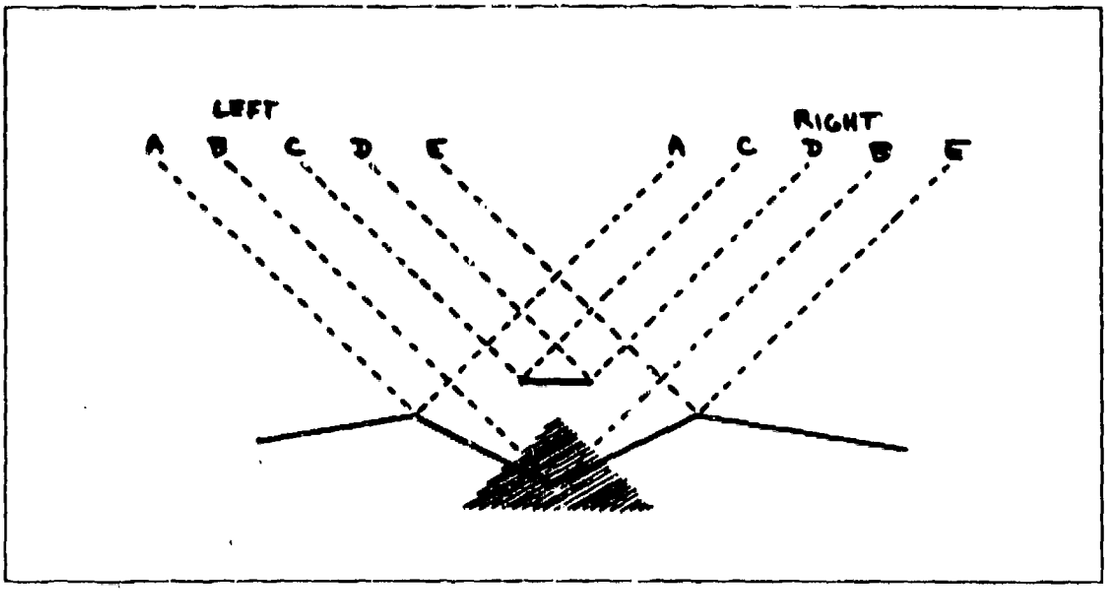


Figure 2-5: *A classic case of order reversal caused by an overhang. Any scene feature falling within the cross-hatched zone will generate an order reversal.*

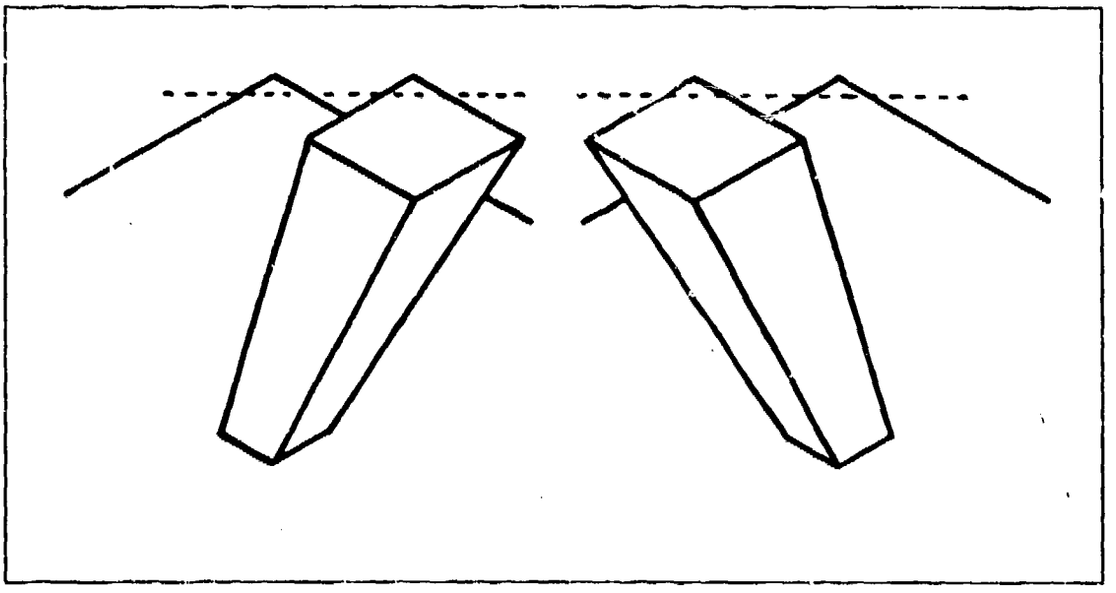


Figure 2-6: *Even a simple block can produce an overhang and an order reversal, but most overhangs do not result in order reversals.*

In summary, we assume that the profile resulting from the intersection of the epipolar plane with the scene consists of one continuous, connected path of straight line segments representing surface intervals. The bounding edges and possible interior markings of these surfaces are detected in each camera except when occluded by an intervening surface. Any such intervening surface must be another part of the same profile.

2.1.3 – Notation

We now introduce some notation and conventions that will be used later in the discussion of profiles. First, we classify the surface intervals which compose a profile. A given interval will fall into one of three classes according to whether it is visible to the left camera only, to the right camera only or to both cameras. The class visible to both can further be divided into four groups according to the visibility of the edges that form its endpoints. Thus we have six types of profile intervals:

1) The surface and both edges are visible to both cameras.

2) The surface and its left edge are visible to both cameras, but its right edge is occluded.

3) The surface is visible only to the left camera

4) The surface is visible only to the right camera.

5) The surface and its right edge are visible to both cameras, but its left edge is occluded.

6) The surface is visible to both cameras, but its left and right edges are occluded.

Figure 2-7 illustrates the coordinate system for profiles, showing a lattice for a profile with three edges visible to the left and three edges visible to the right. Lattice points are identified by ordered pairs, (a, b), where a is the right camera ray number and b is the left camera ray number. The intervals are identified by ordered triples, (a, b, c), defined as:

a) Right camera ray at right end of interval.

b) Left camera ray at right end of interval.

c) Type of interval (one of the six described above).

Notice that type 3 intervals are aligned with rays from the right camera, thus are invisible to it. Similarly, type 4 intervals are aligned with left camera rays. Types 1, 2, 5 and 6 are aligned so as to be visible to both cameras.

The stereo zone in Figure 2-7 includes only nine lattice points, the intersections of $L1$, $L2$, $L3$ with $R1$, $R2$, $R3$. In general, a profile starts somewhere to the left of $L1$ and $R1$ and ends somewhere to the right of $L3$ and $R3$. It may enter the stereo zone at any point on the left and leave at any point on the right, or, if the images don't overlap, it may not enter at all. In effect, the edge of the image serves as a surface which can occlude features in the scene. We term this effect *windowing* and have added rays $L0$, $L4$, $R0$, and $R4$ to represent it. The mechanism of expanding the lattice by one in each direction allows us to describe all the windowing effects in a convenient way. Intervals which are occluded by windowing effects simply appear along one of the four added rays. Thus, we may assume all profiles begin and end at special intervals $(0, 0, 1)$ and $(5, 5, 1)$, which are not themselves visible to either camera.

So far, we have discussed profile intervals whose edges are always on lattice points. This is actually true only for intervals whose edges have no degrees of freedom, i.e., are visible to both cameras and thus must be fixed in space. If an edge is visible to only one camera, it has one degree of freedom; it is free to slide along the ray from the camera

that can see it. Usually its range is bounded by a ray from the other camera, above which it would cease to be occluded.

Some edges, due to windowing effects, are free to slide in either direction indefinitely. Finally, an edge may have two degrees of freedom if it is visible to neither camera. This happens at the start or end of the whole profile, or in the case where an invisible joint must occur between two adjacent intervals that are required to have different slopes. (See the ***valley transition*** in next section). Usually, the two degrees of freedom are bounded by left and right rays with the result that the point may occur anywhere within an infinite wedge defined by those rays.

Thus, profiles which contain degrees of freedom are actually families of profiles, all of which have identical interval types and which produce identical images. We present the following notation for representing such a family of equivalent profiles. See Figure 2-8 for some examples.

A fully constrained point is indicated by a dot. A point with one degree of freedom is drawn at the limiting position with an arrow pointed in the direction in which the point may slide. If the point's range is unbounded, two opposing arrows are used. If a point has two degrees of freedom, the arrows are drawn to define the wedge in which it may be located, and the point is drawn at the vertex of the wedge. Finally, it may occur that two different edges are drawn at the same position, one with a dot and one with an arrow. In this case, the direction of the arrow will make it clear which edge belongs to which surface. In visualizing these profiles, you should imagine an elastic string tied to the fixed dots, but free to be pulled along any of the arrows. The result will be a continuous profile which when viewed by the two cameras will produce the original sequences.

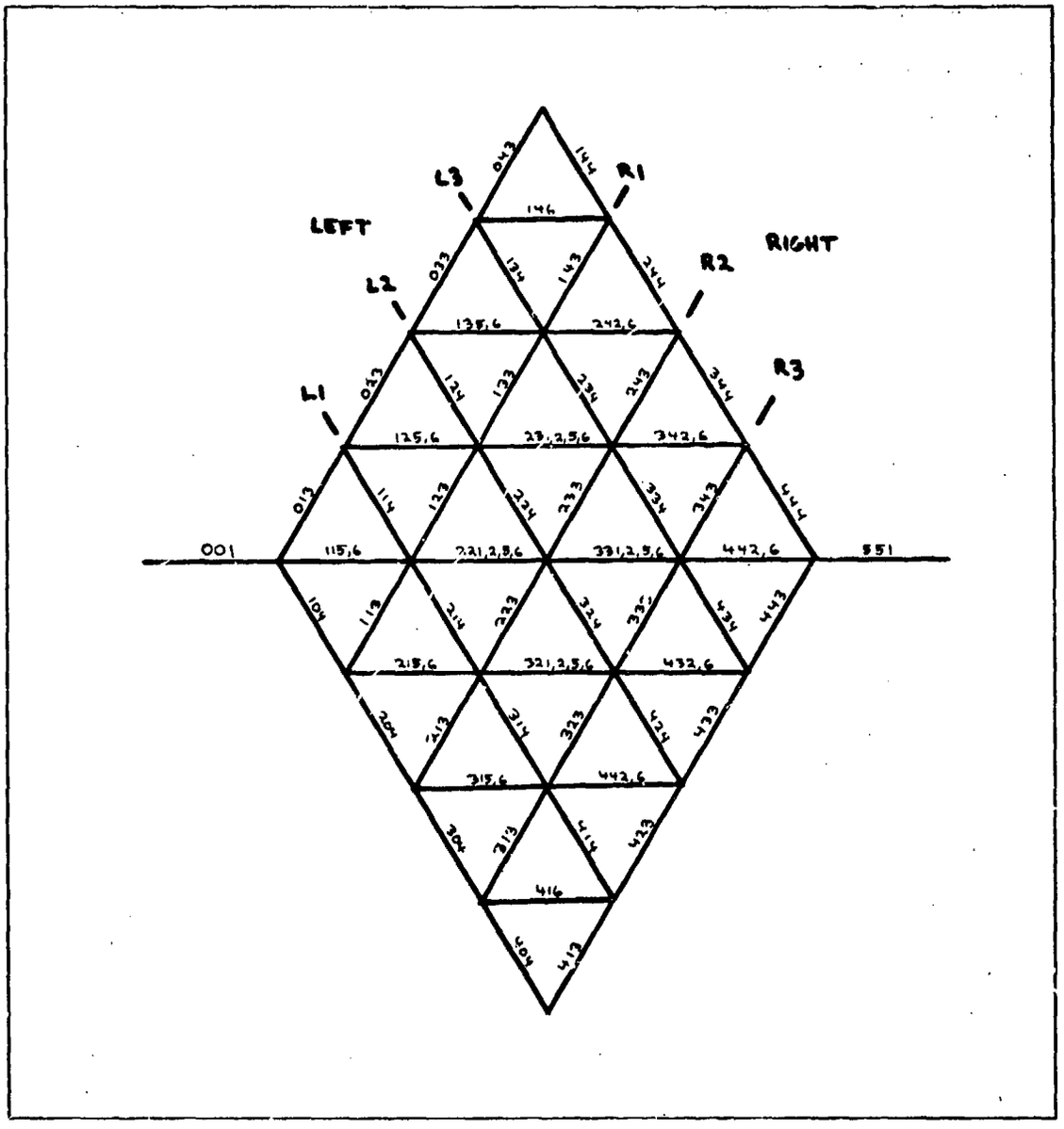


Figure 2-7: *The coordinate system used for profiles is based on rays from each of the cameras, passing through features (edges) in the scene, and through image boundaries. The rays form a lattice of intersection points that cover the stereo zone. Profile intervals join lattice points from left to right, with most points having three nearest neighbors on each side.*

that can see it. Usually its range is bounded by a ray from the other camera, above which it would cease to be occluded.

Some edges, due to windowing effects, are free to slide in either direction indefinitely. Finally, an edge may have two degrees of freedom if it is visible to neither camera. This happens at the start or end of the whole profile, or in the case where an invisible joint must occur between two adjacent intervals that are required to have different slopes. (See the ***valley transition*** in next section). Usually, the two degrees of freedom are bounded by left and right rays with the result that the point may occur anywhere within an infinite wedge defined by those rays.

Thus, profiles which contain degrees of freedom are actually families of profiles, all of which have identical interval types and which produce identical images. We present the following notation for representing such a family of equivalent profiles. See Figure 2-8 for some examples.

A fully constrained point is indicated by a dot. A point with one degree of freedom is drawn at the limiting position with an arrow pointed in the direction in which the point may slide. If the point's range is unbounded, two opposing arrows are used. If a point has two degrees of freedom, the arrows are drawn to define the wedge in which it may be located, and the point is drawn at the vertex of the wedge. Finally, it may occur that two different edges are drawn at the same position, one with a dot and one with an arrow. In this case, the direction of the arrow will make it clear which edge belongs to which surface. In visualizing these profiles, you should imagine an elastic string tied to the fixed dots, but free to be pulled along any of the arrows. The result will be a continuous profile which when viewed by the two cameras will produce the original sequences.

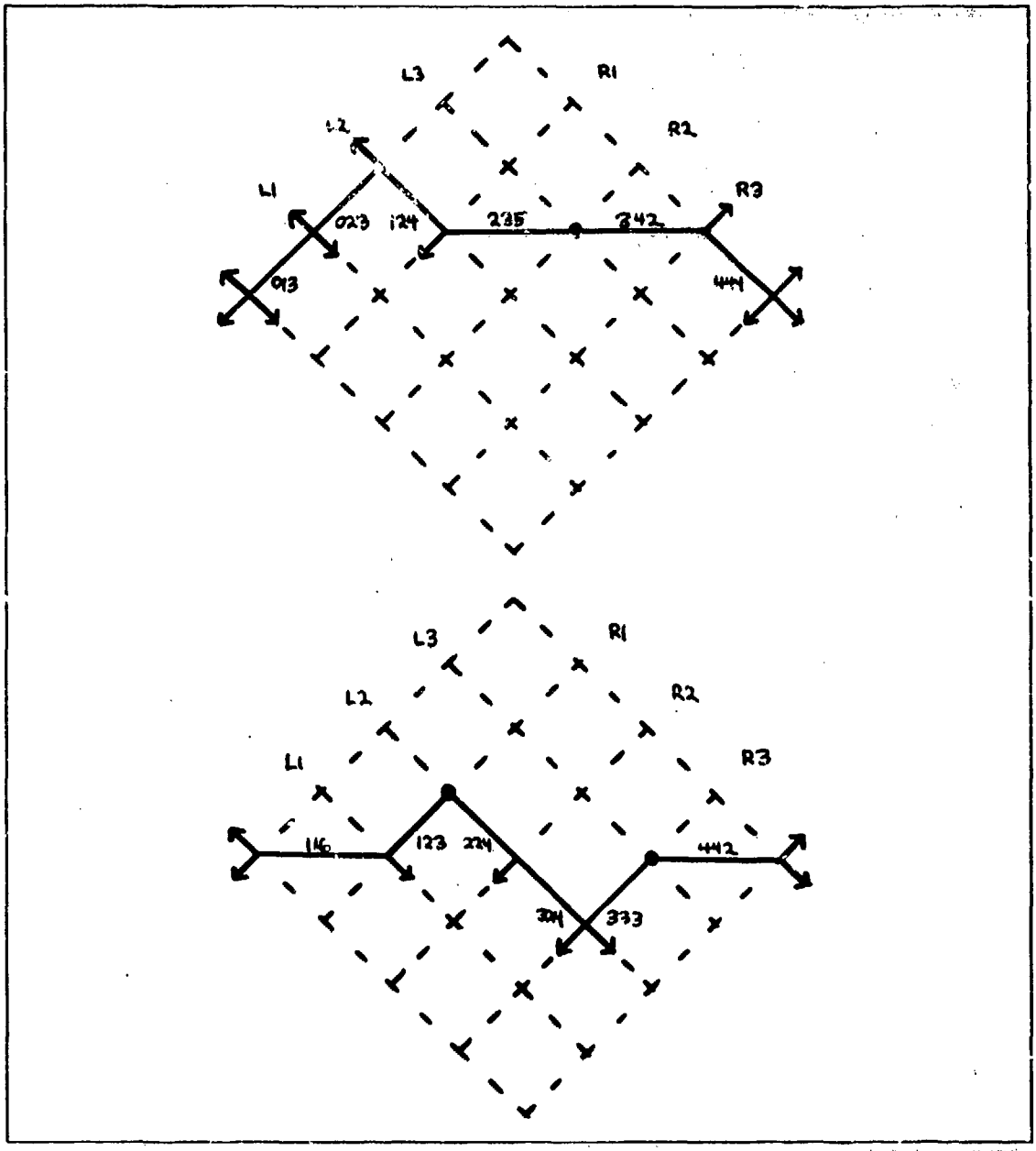


Figure 2-8: *Two examples of profiles indicating the notation for various interval types. The dots represent fully constrained feature points that are fixed in space. The arrows represent degrees of freedom – a feature may "slide" in the direction indicated without changing the projected image.*

2.2 Constraints

Figure 2-9 is an example of reconstructing a profile from epipolar lines. The dotted lines at the top of the figure represent epipolar lines in the left and right views of a stereo image. If we extract these lines, together with the image line intersections, we can then back-project to get our grid. The task then is to reconstruct a profile passing through the lattice points of the grid. The particular profile shown is the one we had in mind when drawing the original images, but in general we must identify the correct match from all possible matches. At this point geometric constraints enter the discussion.

Some matches imply unlikely geometry in the scene; others are simply impossible under our basic assumptions. Several useful constraints are suggested by this example. The length of the intervals between intersection points can be used, as short intervals are more likely matches for other short intervals. A good match criterion is edge angles, i.e., the angle of the image lines where they intersect the epipolar line. On this basis alone, $L4$ should match $R3$. The length of the extended edges would help distinguish $L3$ from $L2$ as a match for $R2$.

In this section we discuss several of these constraints and attempt to quantify them for use in evaluating a match. The evaluation function thus developed will be used in the dynamic programming algorithm to provide solutions to one-dimensional stereo matching problems.

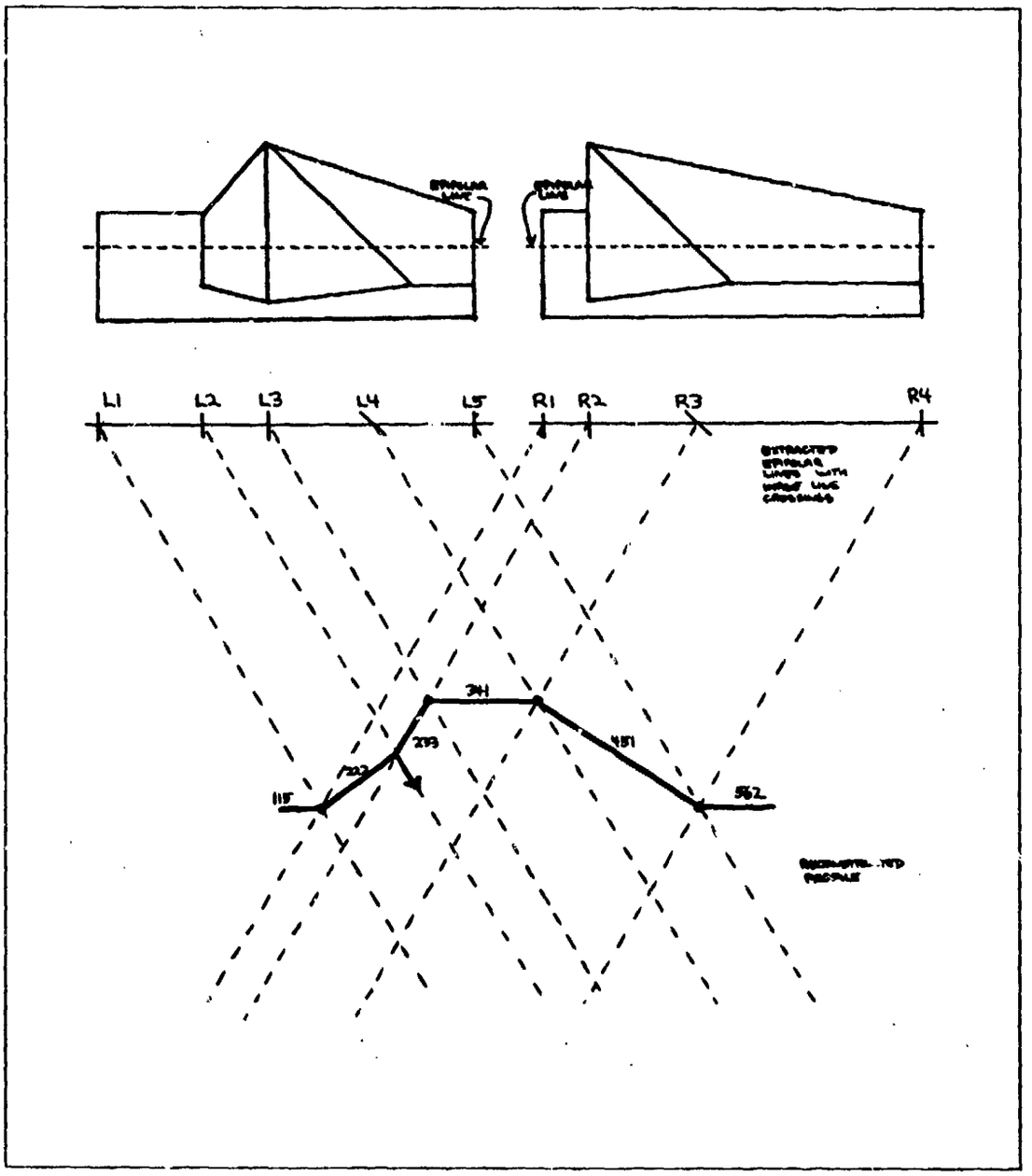


Figure 2-9: *Two epipolar lines are indicated in the sample stereo image at the top. Below the image, and to the same scale, these lines are stripped of all information but the edge intersection points and the angles of those intersections. The profile shown is one of many possible reconstructions from this data.*

2.2.1 – Occlusion

The profile in Figure 2-9 has been labeled with the ordered triples we use to identify intervals. These labels cannot be assigned arbitrarily. In particular, consider $L2$, the second edge in the left image. We are interpreting that edge as visible only to the left camera; there is no edge in the right image to match it. In order to block this edge from view, the surface extending from it toward the right, $(2, 3, 3)$, must have a slope greater than or equal to the slope of ray $R2$ (type 3). On the left side of that same edge, the interval $(2, 2, 2)$, which is in part visible to both cameras, must have its right edge occluded (type 2). Thus the premise that edge $L2$ is occluded has placed constraints on its adjacent surfaces or intervals.

We now consider all possible joints or *transitions* between two intervals. If we look at only the part of an interval next to the transition, we find that our six interval types lead to four possibilities:

1) surface visible to both, edge visible to both,
2) surface visible to both, edge visible to only one,
3) surface visible to right only,
4) surface visible to left only.

If we made these choices independently on each side of a edge, there would be sixteen transition types. However, occlusion constraints make five of these types impossible under our assumptions. They are excluded because either there would be no surface to occlude an edge which shouldn't be visible, or there would be a surface that must occlude a edge that should be visible. The remaining eleven transitions are illustrated in Figure 2-10. Again, the dots indicate edges visible in both views, and arrows indicate degrees of freedom. Note the two degrees of freedom in a *valley*, where a joint must be present, but is visible to neither camera. Also, in a *right cliff*, for example, the interpretation is that the visible edge belongs to the left surface, while the right surface is free to slide along the arrow. We hypothesize an invisible surface and joint connecting the two to preserve profile continuity.

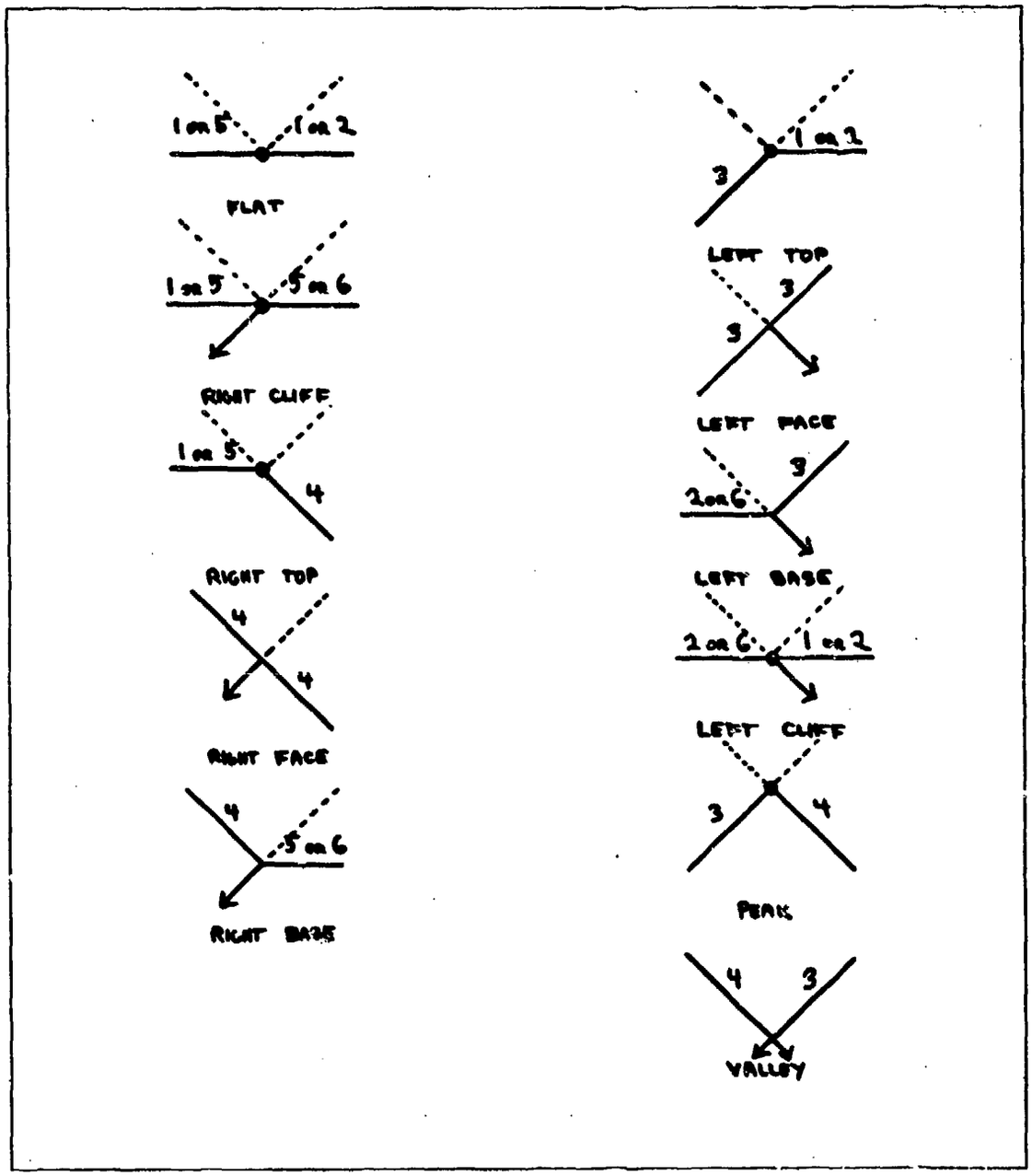


Figure 2-10: *Eleven transition types satisfy the occlusion constraint. In our notation, these are the only interval types that may be adjacent in a profile.*

2.2.2 -- Edge Intervals

Given an object surface, its image at a particular epipolar line will generally consist of two bounding edges and the interval between them. If the object surface is visible to both cameras, there will be a corresponding interval in each image. The lengths of these intervals are related to the angle of the surface and to the camera geometry. The lengths can take on any values, but for moderate or small baselines they are usually comparable. In this section we describe a method for quantifying this relation; the next section presents the detailed mathematics.

Under our assumptions, each epipolar plane cuts a continuous profile in the scene. Now consider the case where the profile consists of a central small surface flanked by two larger ones extending off to the left and right (see Figure 2-11). We want to vary the orientation of the small surface and see what happens to its image. In general, the left and right images will show an interval between two edges. The length of the interval will depend on the orientation of the surface and its position with respect to the cameras. For some orientations, one of the edges may be occluded and the small surface may not be visible.

We see immediately that there is a simple function mapping orientation, ϑ, to projected interval lengths, p_l and p_r. Since we are working to reconstruct the scene from the image, we need an inverse function mapping some image parameter to ϑ. To do this, we define a ratio, $R = p_r/p_l$. This has the advantage of reducing the information from the image interval lengths into a single number while eliminating the dependency on segment length, d. (The derivation is presented in the next section.) Now we can easily invert the function and take the derivative. The resulting $d\vartheta/dR$ is a scale factor which indicates how much a unit length in "R-space" is stretched in mapping to "ϑ-space". This allows us to translate probability densities. For example, suppose an interval ratio a maps to an

orientation A, the derivative of the mapping at a is $D(a)$, and the probability density at orientation A is $P(A)$. Then the probability density for ratio a is $D(a)P(A)$.

The derivative D is normalized and plotted against R in Figure 2-12a. ϑ ranges from -90 to +90 degrees while the domain of R extends from $-\infty$ to $+\infty$. A ratio of zero corresponds to a surface exactly in line with the right camera, while a ratio of $\pm\infty$ corresponds to alignment with the left camera. Negative ratios result when the surface presents a different face to each camera. If the surfaces are opaque, this condition corresponds to occlusion.

Given this mapping, we are now able to translate probability distributions in one domain to probability distributions in the other. For example, we are interested in the following problem: assuming a particular distribution of surface angles in the scene, what

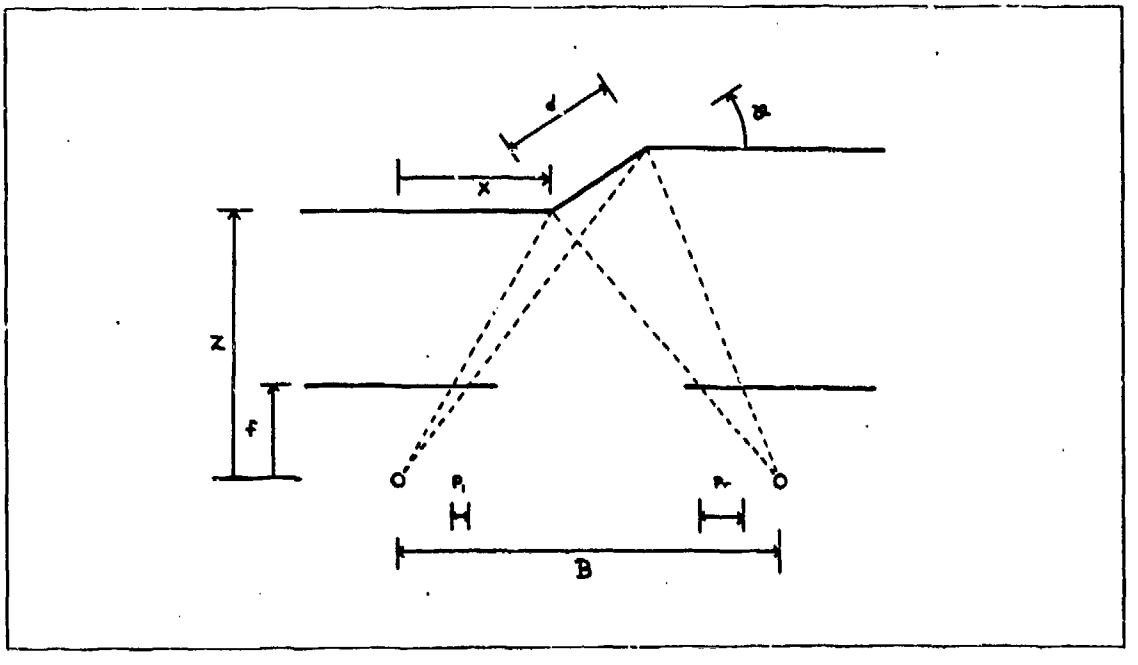


Figure 2-11: *The calculation of the Edge Interval Constraint is based on the camera geometry shown here, viewed along the y-axis. The ratio of projected image intervals, p_r/p_l, is a function of surface orientation, ϑ.*

distribution of interval length ratios can be expected in the image? Knowing the answer to this question will help to discriminate among potential stereo matches.

If we assume that the small segment in Figure 2-11 takes on all orientations uniformly, then D exactly equals the probability density for interval ratios. The peak near $R = 1$ indicates that under such an assumption, most intervals tend to have comparable lengths. This peak becomes much sharper for narrower baselines (see Figure 2-12). Human stereo at a range of 1 meter uses a baseline of $B/z = 0.07$. With that geometry, half of all ratios lie between 0.93 and 1.07. Note that integrating the probability function between $-\infty$ and 0 gives the range of angles for which occlusion may occur. When normalized, this is the probability of occlusion.

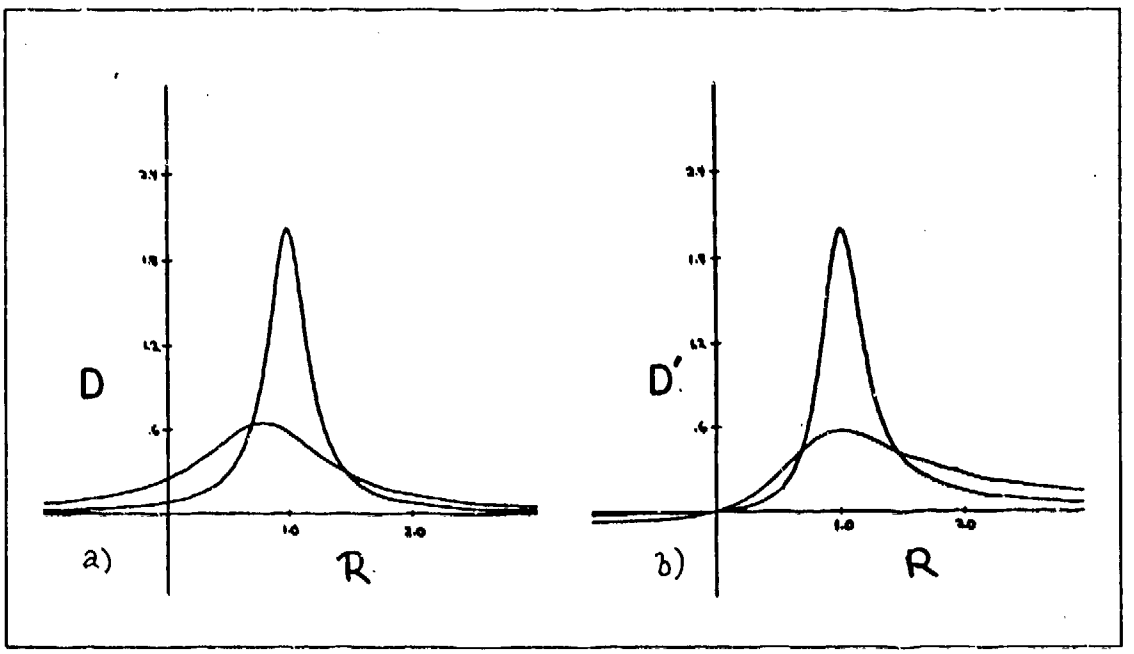


Figure 2-12: *Probability density is plotted against interval length ratio to show which ratios are most likely to occur. The curves peak near $R = 1$, indicating that the lengths in the two images are most likely to be very similar. The shape of the curves varies with the camera geometry. In this Figure, the more sharply peaked curve results from a narrow baseline ($B/z = .07$), while the broader curve is for a wide baseline ($B/z = .2$). The same two curves have been scaled in b to satisfy the symmetry condition that requires identical values for ratios that are inverses.*

There is one problem with using the results of Figure 2-12a directly. While the function gives the true probability density per unit R, dR is a nonuniform unit varying in length from 0 to ∞. Consequently, orientations which are simple reflections, i.e., ϑ and $-\vartheta$, yield different values. To adjust for this, we scale the derivative by a factor of R, yielding the function in Figure 2-12b. This function satisfies the conditions of symmetry, in that symmetric orientations now have identical values. Another way to get this same result is to use $\log R$ as the image parameter and take the derivative of ϑ with respect to $\log R$.

2.2.3 – Interval Constraint Derivation

Referring to Figure 2-11, we assume that x, z, B, d, and ϑ are given. The projected interval lengths, p_l and p_r are determined:

$$\frac{p_l z/f + x}{d\cos\vartheta + x} = \frac{z}{d\sin\vartheta + z}$$

$$p_l = \frac{fd}{z}\ \frac{z\cos\vartheta - x\sin\vartheta}{d\sin\vartheta + z} \qquad (2-1)$$

$$\frac{p_r z/f - (B-x)}{d\cos\vartheta - (B-x)} = \frac{z}{d\sin\vartheta + z}$$

$$p_r = \frac{fd}{z}\ \frac{z\cos\vartheta + (B-x)\sin\vartheta}{d\sin\vartheta + z}. \qquad (2-2)$$

Letting $R = p_r/p_l$, $a = B/x$ and $b = z/x$:

$$R = \frac{z\cos\vartheta + B\sin\vartheta - x\sin\vartheta}{z\cos\vartheta - x\sin\vartheta}$$

$$= 1 + \frac{B/x}{\cot\vartheta - x/z}$$

$$= 1 + \frac{a}{\cot\vartheta - b}. \qquad (2-3)$$

Now we need ϑ as a function of R:

$$\vartheta = \cot^{-1}\left(\frac{a}{R-1} + b\right). \qquad (2-4)$$

This gives a function mapping the image parameter $R = p_r/p_l$ to the object space parameter ϑ. The mapping of probability densities requires the derivative of this function. Using:

$$\frac{d}{dx}\cot^{-1} u = -\frac{1}{1+u^2}\frac{du}{dx}$$

we substitute to find:

$$D(R) = \frac{d\vartheta}{dR} = \frac{a}{(a + b(R-1))^2 + (R-1)^2}. \qquad (2-5)$$

As noted in the text, D is not symmetrical in its use of R for the case where the object is halfway between the two cameras ($b = .5a$). By noting that $dR/d\log R = R$, we have:

$$D'(R) = \frac{d\vartheta}{d\log R} = \frac{aR}{(a + b(R-1))^2 + (R-1)^2}. \qquad (2-6)$$

This function is symmetrical in R ($D'(1/R) = D'(R)$) whenever $b = .5a$, as shown by:

$$\frac{a/R}{(a + .5a(1/R - 1))^2 + (1/R - 1)^2} = \frac{aR}{(aR - .5a(R-1))^2 + (R-1)^2}$$
$$= \frac{aR}{(a + .5a(R-1))^2 + (R-1)^2}.$$

Finally, we locate the extrema of this function by taking the first derivative:

$$\frac{dD'}{dR} = \frac{a}{((a + b(R-1))^2 + (R-1)^2)^2}(-(b^2+1)R^2 + (a-b)^2 + 1).$$

Setting this equal to zero and solving for R gives the values for which $D'(R)$ reaches a maximum and minimum:

$$R = \pm\sqrt{\frac{1 + (a-b)^2}{1 + b^2}}. \qquad (2-7)$$

2.2.4 – Edge Angles

Given a corresponding pair of edge curves, one in the left image and one in the right image, we are interested in how their angles are related (or more precisely, the angle of intersection with a given epipolar line). In general, the two angles may take on any values, but we intuitively expect them usually to be similar, especially for moderate or small baselines. This is in fact the case, as we will now show. (The next section will give the detailed derivation.)

Consider an object edge passing through a scene point (x, y, z). The edge at that point has an orientation in three dimensions which can be characterized as a point

on the surface of a unit sphere, whose origin is (x, y, z). This is known as the ***gaussian sphere***, and points are located on its surface in terms of spherical coordinates θ and φ (see Figure 2-13). The spherical coordinate axis is parallel to the z axis and θ corresponds to *longitude*, measured counter-clockwise from the x axis when viewed from the cameras. φ corresponds to *latitude* and is measured from the sphere's axis.

The object edge projects to a line intersecting the epipolar lines in the two images determined by (x, y, z). Let the angle of the image curve in the left image be θ_l and in the right image be θ_r, measured counter-clockwise from the x axis. A continuous function maps points on the gaussian sphere to pairs of image angles, (θ_l, θ_r). Similarly, there is an inverse function which maps points in the space $\theta_l \times \theta_r$ to points on the gaussian sphere. This inverse function is defined everywhere except at $(0, 0)$. This is because the great circle of points on the sphere for which $\theta = 0$ all map to $(0, 0)$, and the function is not invertible at that point.

This inverse function allows us to translate probability distributions in one domain to probability distributions in the other. If, for example, there is a uniform distribution on the gaussian sphere, we could calculate the expected distribution of image angles. In other words, if all object edges are randomly and uniformly distributed in orientation, are some combinations of (θ_l, θ_r) more likely than others?

We know the mapping from $\theta_l \times \theta_r$ to $\theta \times \varphi$. The determinant of the matrix of partial derivatives (Jacobian matrix) is the scale factor for area under the mapping, and thus is the scale factor for probability density. Suppose point (a, b) in $\theta_l \times \theta_r$ maps to (A, B) in $\theta \times \varphi$, and that the determinant of the Jacobian at (a, b) is $D(a, b)$. Then a small patch around (a, b) maps to a patch around (A, B) with $D(a, b) \sin \varphi$ times the area. The $\sin \varphi$ term compensates for the area distortion of the spherical coordinates. If the probability density at (A, B) is $P(A, B)$, then the probability density at (a, b) is

$$D(a, b)P(A, B) \sin \varphi$$

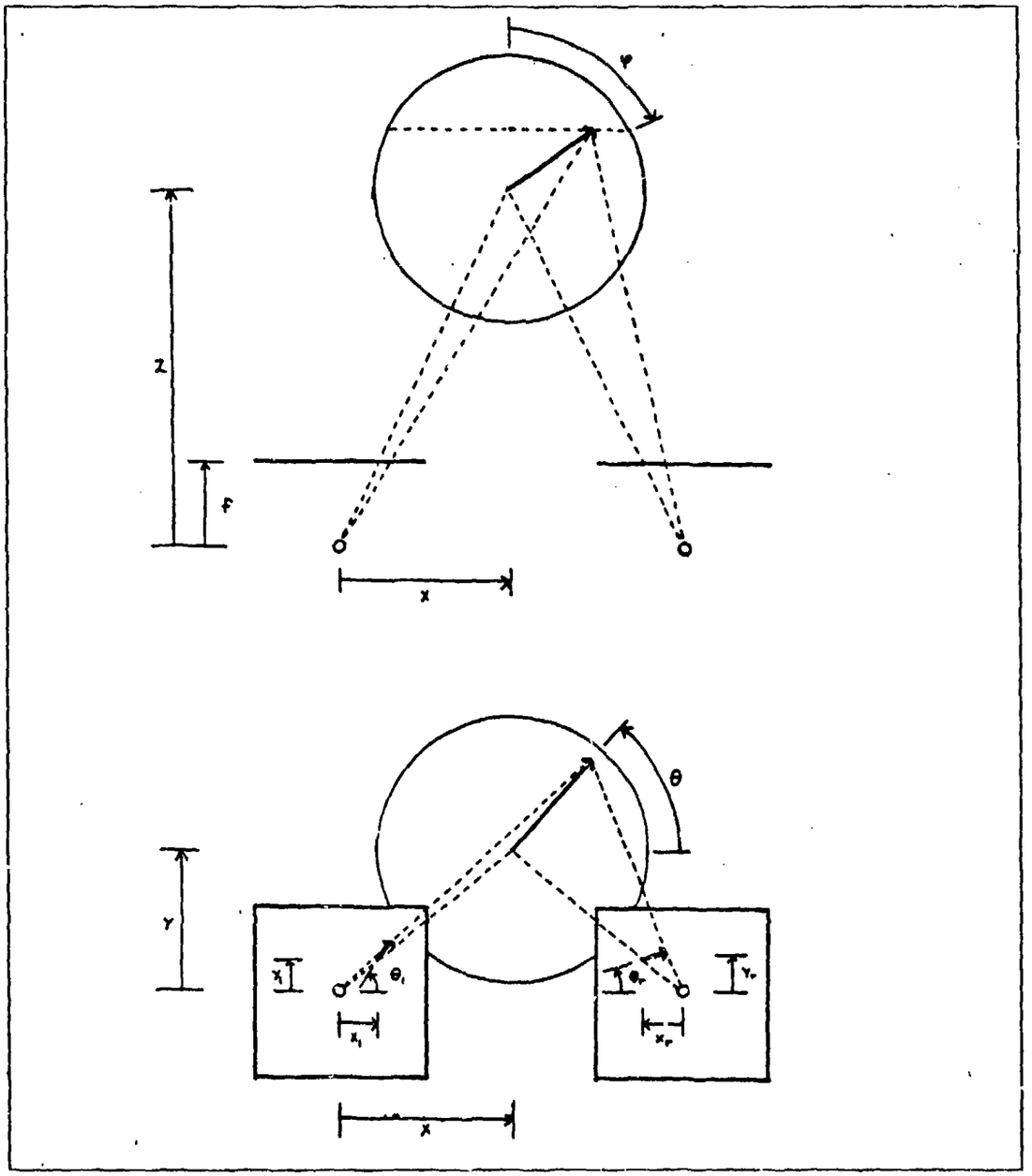


Figure 2-13: *The calculation of the edge angle constraint is based on the camera geometry shown here. The image angles, θ_l and θ_r, are functions of the edge orientation given by spherical coordinates θ and φ.*

Figures 2-14a and 2-14b show the function D plotted for a stereo baseline typical of aerial photographs, $B/z = 0.7$. For the uniform distribution assumption, this surface corresponds directly to probability distribution over $\theta_l \times \theta_r$. The surface forms a high, narrow saddle along the line $\theta_l = \theta_r$, with a singularity at $(0, 0)$. This corresponds to the intuitive notion that left and right angles are usually similar, but the sharpness is surprising. Half width at half m ximum (HWHM) at the center is 30°. As Figures 2-14c and 2-14d show, probability functions for narrower baselines are even sharper. For $B/z = 0.07$, which corresponds to human vision at a range of about 1 meter, the HWHM at the center is 3°.

Another way of looking at the data is to consider the distribution of "wrong matches". Suppose we choose an edge at random from the left and from the right, and try to interpret them as corresponding. If we do this for a large set of edges we will produce a distribution of edges in 3 dimensions, i.e., on the surface of the gaussian sphere. The nature of the distribution will depend on the distributions of θ_l and θ_r.

We originally assumed a uniform distribution over the gaussian sphere. For this case, it is easy to show that θ_l and θ_r are also uniformly distributed. For each value of θ_l in the image, there is a corresponding set of points on the gaussian sphere. This set of points forms a great circle, that is, a circle of unit radius. The probability of a particular value of θ_l occuring depends on the integral of the gaussian sphere probability distribution over that circle. If we assume a uniform distribution on the sphere, then all circles will yield identical integrals. Similarly, θ_r will be uniformly distributed.

If we choose unrelated left and right edges from these uniform distributions and project back to the gaussian sphere, we get the distributions shown in Figure 2-15. The distributions, which are actually on the surface of the sphere, have been cut in half and projected onto the plane of the image for display. The result is a sharply double peaked

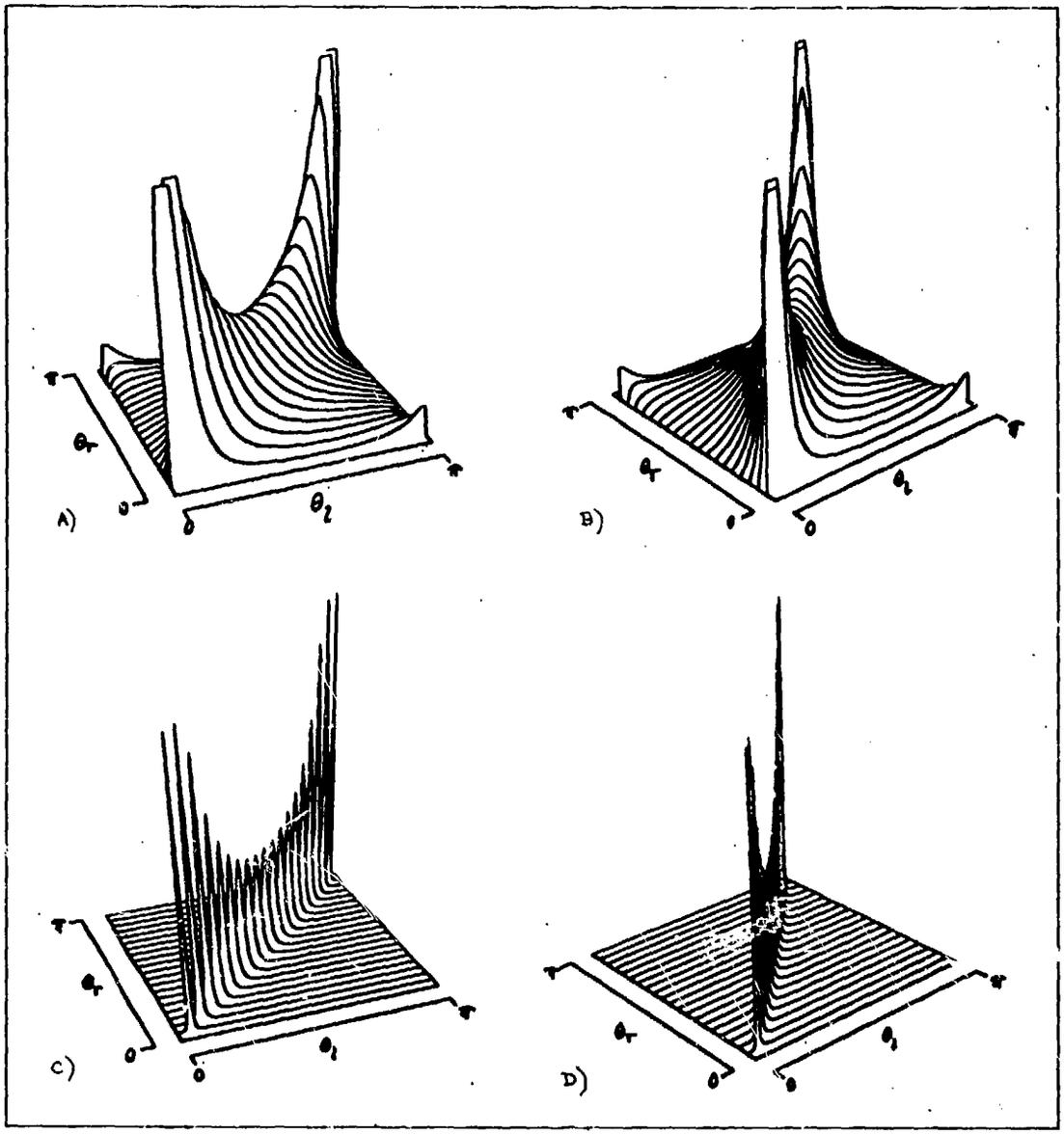


Figure 2-14: *Probability density is plotted against image angles to show which combinations of angles are most likely to occur. The curves peak near the line $\theta_l = \theta_r$, indicating that the angles in the two images are most likely to be very similar. The shape of the curves varies with the camera geometry. Graphs a and b are two views of the function for a wide baseline ($B/z = .7$). Graphs c and d show the much sharper function for a narrow baseline ($B/z = .07$).*

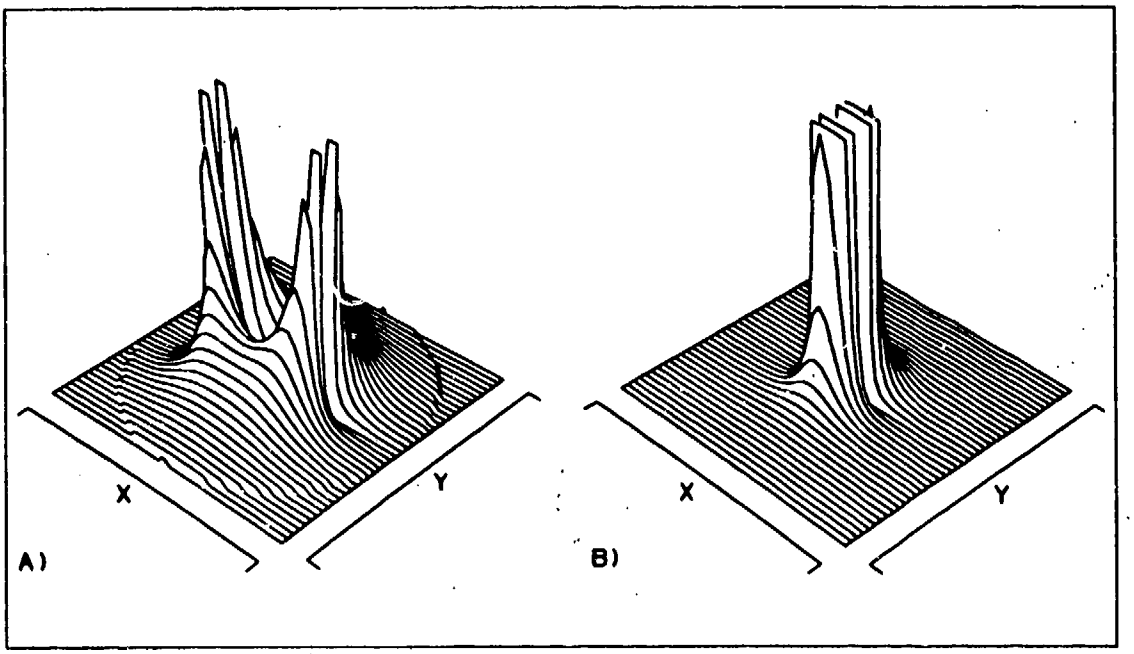


Figure 2-15: *If edges from the left and right images were matched at random, the 3-dimensional orientations of the reconstructed scene edges would be distinctly non-uniform. A uniform distribution of image edge angles maps to a distribution on the gaussian sphere that is strongly peaked along the line of sight of the two cameras. Surfaces a and b result from baselines of $B/z = .7$ and $B/z = .07$, respectively.*

distribution, with each peak oriented toward (and the missing half away from) a camera. This violates the assumption that the scene should be independent of the observer. Such a distribution could be used to identify wrong matches.

Figure 2-15a results from a wide baseline of $B/z = .7$. The twin peaks are quite clear in this graph; each contains a singularity at $\varphi = 19.29°$ ($\tan\varphi = .35$) and $\theta = 0°$ or $\theta = 180°$ that has been clipped to limit the height of the graph. The distribution has a value of zero along the "equator" where $\theta = 0°$ or 180°, and a value of .7 at the "poles" where $\theta = \pm 90°$. This rises to a value of 5.6 at the saddle between the peaks.

Figure 2-15b, based on $B/z = .07$, is similar in shape but more extreme in value. It is graphed at the same scale as Figure 2-15a for comparison. The singularities are o..

the equator at $\varphi = 2.00°$ ($\tan\varphi = .035$) and the poles have a value of .07. The peaks are not separated at this scale since the saddle between them has a value of 525.

2.2.5 – Angle Constraint Derivation

Refer to Figure 2-13 for geometry of this derivation. We wish to derive the function mapping $\theta_l \times \theta_r \mapsto \theta \times \varphi$, where

$$0 \leq \theta_l, \theta_r < \pi$$

$$0 \leq \theta < \pi$$

$$0 \leq \varphi \leq \pi.$$

The approach is to convert to rectangular coordinates, do the stereo projections, and convert to spherical coordinates. The stereo projections are given by

$$(x_l, y_l, z_l) = \left(\frac{f}{z}x, \frac{f}{z}y, f\right)$$

$$(x_r, y_r, z_r) = \left(\frac{f}{z}(x - B), \frac{f}{z}y, f\right),$$

and the inverse projections are given by

$$z = \frac{fB}{x_l - x_r}$$

$$y = \frac{z}{f}y_l = \frac{z}{f}y_r$$

$$x = \frac{z}{f}x_l = \frac{z}{f}x_r + B,$$

where:

f is the image length,

B is the base line,

(x, y, z) is a point on the object,

(x_l, y_l, z_l) is a point in the left image, and

(x_r, y_r, z_r) is a point in the right image.

Now consider a unit vector in the left image, centered at (x_l, y_l), at angle θ_l. The tip of the vector has coordinates

$$x'_l = x_l + \cos\theta_l$$

$$y'_l = y_l + \sin\theta_l.$$

From epipolar geometry, we know the points in the right image corresponding to the endpoints of the vector will have the same y-coordinates. Thus, the length of the vector in the right image must be $\sin\theta_l / \sin\theta_r$, and

$$x'_r = x_r + \frac{\sin\theta_l}{\tan\theta_r}$$

$$y'_r = y_r + \sin\theta_l,$$

where θ_l is the angle in the left image plane and θ_r is the angle in the right image plane.

We now inverse project to get the points (x, y, z) and (x', y', z') in object space, the origin and tip of the vector respectively. Note that this vector will not have unit

length, but its orientation will supply the correct value for θ and φ. The values $x' - x$, $y' - y$, and $z' - z$ will be needed:

$$x' - x = \frac{z'}{f}(x_l + \cos\theta_l) - \frac{z}{f}x_l$$

$$= B\left(\frac{x_l + \cos\theta_l}{x_l - x_r + \cos\theta_l - \frac{\sin\theta_l}{\tan\theta_r}} - \frac{x_l}{x_l - x_r}\right)$$

$$y' - y = B\left(\frac{y_l + \sin\theta_l}{x_l - x_r + \cos\theta_l - \frac{\sin\theta_l}{\tan\theta_r}} - \frac{y_l}{x_l - x_r}\right)$$

$$z' - z = fB\left(\frac{1}{x_l - x_r + \cos\theta_l - \frac{\sin\theta_l}{\tan\theta_r}} - \frac{1}{x_l - x_r}\right).$$

To simplify further calculations, we use the following substitutions:

$$Q = \frac{B}{(x_l - x_r)(x_l - x_r + \cos\theta_l - \frac{\sin\theta_l}{\tan\theta_r})}$$

$$U = (x_l - x_r)\cos\theta_l - x_l\left(\cos\theta_l - \frac{\sin\theta_l}{\tan\theta_r}\right)$$

$$V = (x_l - x_r)\sin\theta_l - y_l\left(\cos\theta_l - \frac{\sin\theta_l}{\tan\theta_r}\right)$$

$$W = -f\left(\cos\theta_l - \frac{\sin\theta_l}{\tan\theta_r}\right).$$

Then $x' - x = QU$, $y' - y = QV$, and $z' - z = QW$ and we can easily convert to spherical

coordinates:

$$\theta = \tan^{-1}\left(\frac{y'-y}{x'-x}\right) = \tan^{-1}\left(\frac{V}{U}\right) \qquad (2-8)$$

$$\varphi = \cos^{-1}\left(\frac{z'-z}{\sqrt{(x'-x)^2+(y'-y)^2+(z'-z)^2}}\right)$$

$$= \cos^{-1}\left(\frac{W}{\sqrt{U^2+V^2+W^2}}\right). \qquad (2-9)$$

This completes the derivation of a function mapping image parameters θ_l and θ_r to object space parameters θ and φ. The mapping of probability densities requires the derivative of this function, or more precisely, the determinant of the Jacobian matrix.

The Jacobian matrix is given by:

$$J = \begin{pmatrix} \frac{\partial\theta}{\partial\theta_l} & \frac{\partial\theta}{\partial\theta_r} \\ \frac{\partial\varphi}{\partial\theta_l} & \frac{\partial\varphi}{\partial\theta_r} \end{pmatrix}.$$

To calculate these values, we will need the partial derivatives of U, V, and W:

$$\frac{\partial U}{\partial\theta_l} = -(x_l - x_r)\sin\theta_l + x_l\left(\sin\theta_l + \frac{\cos\theta_l}{\tan\theta_r}\right)$$

$$\frac{\partial V}{\partial\theta_l} = (x_l - x_r)\cos\theta_l + y_l\left(\sin\theta_l + \frac{\cos\theta_l}{\tan\theta_r}\right)$$

$$\frac{\partial W}{\partial\theta_l} = f\left(\sin\theta_l + \frac{\cos\theta_l}{\tan\theta_r}\right)$$

$$\frac{\partial U}{\partial\theta_r} = \frac{-x_l\sin\theta_l}{\sin^2\theta_r}$$

$$\frac{\partial V}{\partial\theta_r} = \frac{-y_l\sin\theta_l}{\sin^2\theta_r}$$

$$\frac{\partial W}{\partial\theta_r} = \frac{-f\sin\theta_l}{\sin^2\theta_r}.$$

Now we have

$$\frac{\partial \theta}{\partial x} = \frac{\partial}{\partial x} \tan^{-1} \frac{V}{U} = \frac{U \frac{\partial V}{\partial x} - V \frac{\partial U}{\partial x}}{U^2 + V^2}.$$

Substituting $d = \sqrt{U^2 + V^2 + W^2}$, we have

$$\frac{\partial d}{\partial x} = \frac{U \frac{\partial U}{\partial x} + V \frac{\partial V}{\partial x} + W \frac{\partial W}{\partial x}}{\sqrt{U^2 + V^2 + W^2}}$$

$$\frac{\partial \varphi}{\partial x} = \frac{\partial}{\partial x} \cos^{-1} \frac{W}{d} = \frac{W \frac{\partial d}{\partial x} - d \frac{\partial W}{\partial x}}{d\sqrt{d^2 - W^2}}$$

$$= \frac{W(U \frac{\partial U}{\partial x} + V \frac{\partial V}{\partial x} + W \frac{\partial W}{\partial x}) - \frac{\partial W}{\partial x}(U^2 + V^2 + W^2)}{(U^2 + V^2 + W^2)\sqrt{U^2 + V^2}}.$$

We can calculate the components of the Jacobian matrix by substituting θ_l or θ_r for x. The determinant is then

$$D(\theta_l, \theta_r) = \det J = \frac{\partial \theta}{\partial \theta_l} \frac{\partial \varphi}{\partial \theta_r} - \frac{\partial \theta}{\partial \theta_r} \frac{\partial \varphi}{\partial \theta_l}. \qquad (2-10)$$

Finally, this probability scale factor must be corrected for the area distortion of the spherical coordinates:

$$\text{scale factor} = D(\theta_l, \theta_r) \sin \varphi. \qquad (2-11)$$

This function is plotted in Figure 2-14 for different camera parameters. The resulting saddle-shaped surfaces have been numerically integrated to give a volume of about 2π, or an average value for the function of $2/\pi$.

2.2.6 – Edge extent

The extent of an image curve can be useful in evaluating matches, subject to certain limitations. We define *extent* as the difference in y-coordinates of the two end points of a curve. This measure, even more than edge angle, involves two-dimensional information, but we believe in using constraints as early in the processing as possible, provided there is a clear way to apply them. Ideally, image features in isolation should have identical extents. This is due to the normal camera model we have chosen, where stereo disparity occurs in the x direction only. Several things can modify this, however. Inaccuracies in the stereo camera model can cause deviations in the projected position of the endpoints. If the images differ by a scale-factor, the difference in extents for a matched edge pair will also depend on the magnitude of the extent.

Edge extent can be measured only where both end points are visible to both cameras. However, for some occlusions, we can derive an inequality condition, which still may be used to discredit a match. We assume occlusions from the presence of a T-junction, which Binford and Lowe [Binford 1981, Lowe 1981] have shown may be considered a necessary condition for occlusion of a curve. Figure 2-16 illustrates four potential matches, two of which satisfy the inequality and two of which don't.

Finally, image curve *segmentation* can cause problems. Segmentation means breaking a long complex curve into simpler pieces, connected end to end. Since corresponding curves in the two views are segmented independently, corresponding pieces may have different extents. This may be compensated or at least recognized by examining the junctions at the ends of an image curve. A 2-junction with similar curve may suggest a segmentation problem, especially if the tangents or curvatures are similar at that point.

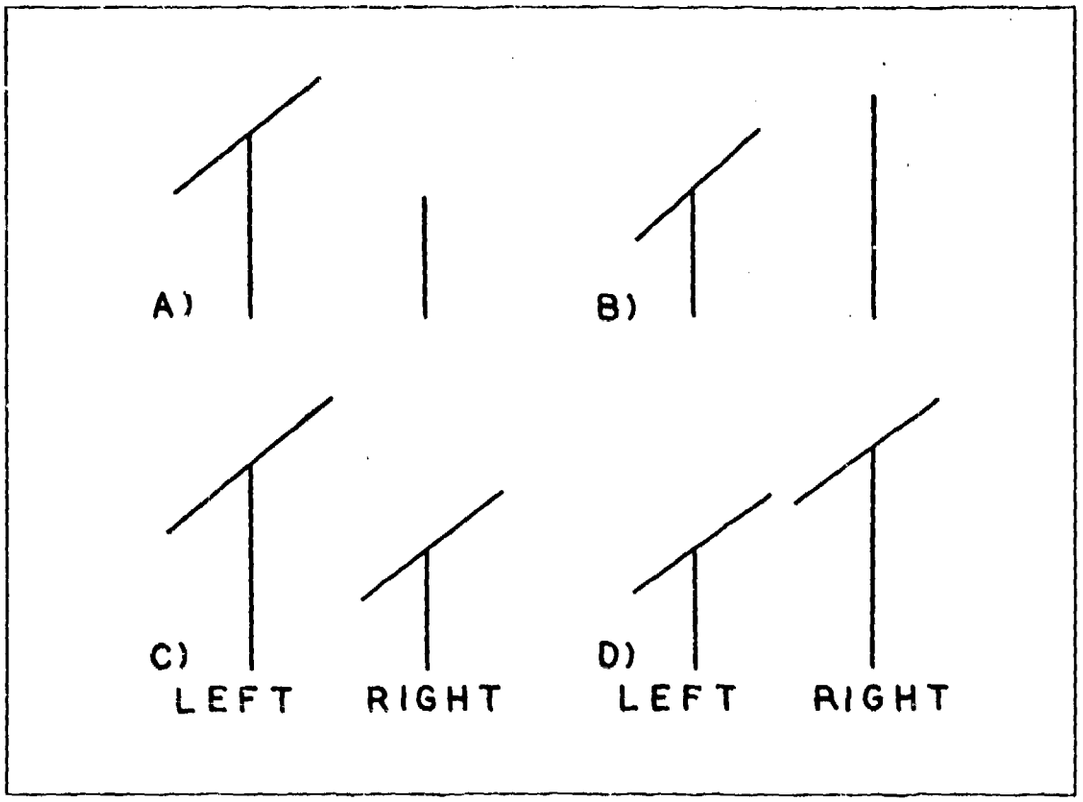


Figure 2-16: *Slanted T-junctions can mean that an edge's extent is known only to an inequality, but this is sufficient to reject some matches. The stereo pairs shown in b and d are possible matches; those in a and c are not.*

2.2.7 – Additional Constraints

Image intensities can be used as a match criterion, although they have many drawbacks, as discussed in the introduction. Still, with a proper allowance for error, they can help to resolve some ambiguities. We have used a simple measure based on average brightness across the interval, but a more sophisticated approach might use area-based correlation with image curves as boundaries.

Other geometric constraints may be taken from the vertices of image curves. For example, if a curve terminates in a 3-vertex in the left image, there are only certain types

of 2- or 3-vertices in the right image that could match it. This type of constraint can be very strong in certain restricted scene domains, like right parallelepipeds, but some constraints apply also to general scenes. We have not yet made use of this type of measure.

2.3 Dynamic Programming

Dynamic programming is a technique useful for matching two sequences. A typical application is in speech recognition, where one of the sequences is the model of a spoken word, and the other is the encoded signal derived from a microphone. Various portions or phonemes may be stretched or compressed, but the continuous flow of time guarantees that no two components will be out of order in one sequence relative to the other. Dynamic programming attempts to map one sequence onto the other, subject to these constraints.

2.3.1 – Introduction to the Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm which finds a "best" match from among all the allowable matches. Figure 2-17 illustrates this algorithm applied to a simple problem. The sequences to be matched, $\{L_i\}$ and $\{R_i\}$, define the two dimensions of a matrix; each entry is determined by a pair of elements, one from each sequence. A function is defined on this matrix such that each entry represents the ***cost*** of matching that pair of elements. This function measures the dissimilarity of the two elements. A ***path*** will consist of a sequence of ***nodes***, each of which corresponds to one entry in the matrix. The goal is to find a path through the cost matrix such that the sum of the costs along the path is a minimum.

To do this we need to define a set of transition rules that specify the allowable successors to a given node on a path. These rules may be derived from constraints on the original problem. For example, assume the following constraints:

- The sequences must be matched monotonically.
- Every element of each sequence must be used at least once.

These constraints are equivalent to assuming that the path must start in the lower left corner and end in the upper right, and that from each node, a transition may only be made one unit vertically, horizontally, or diagonally.

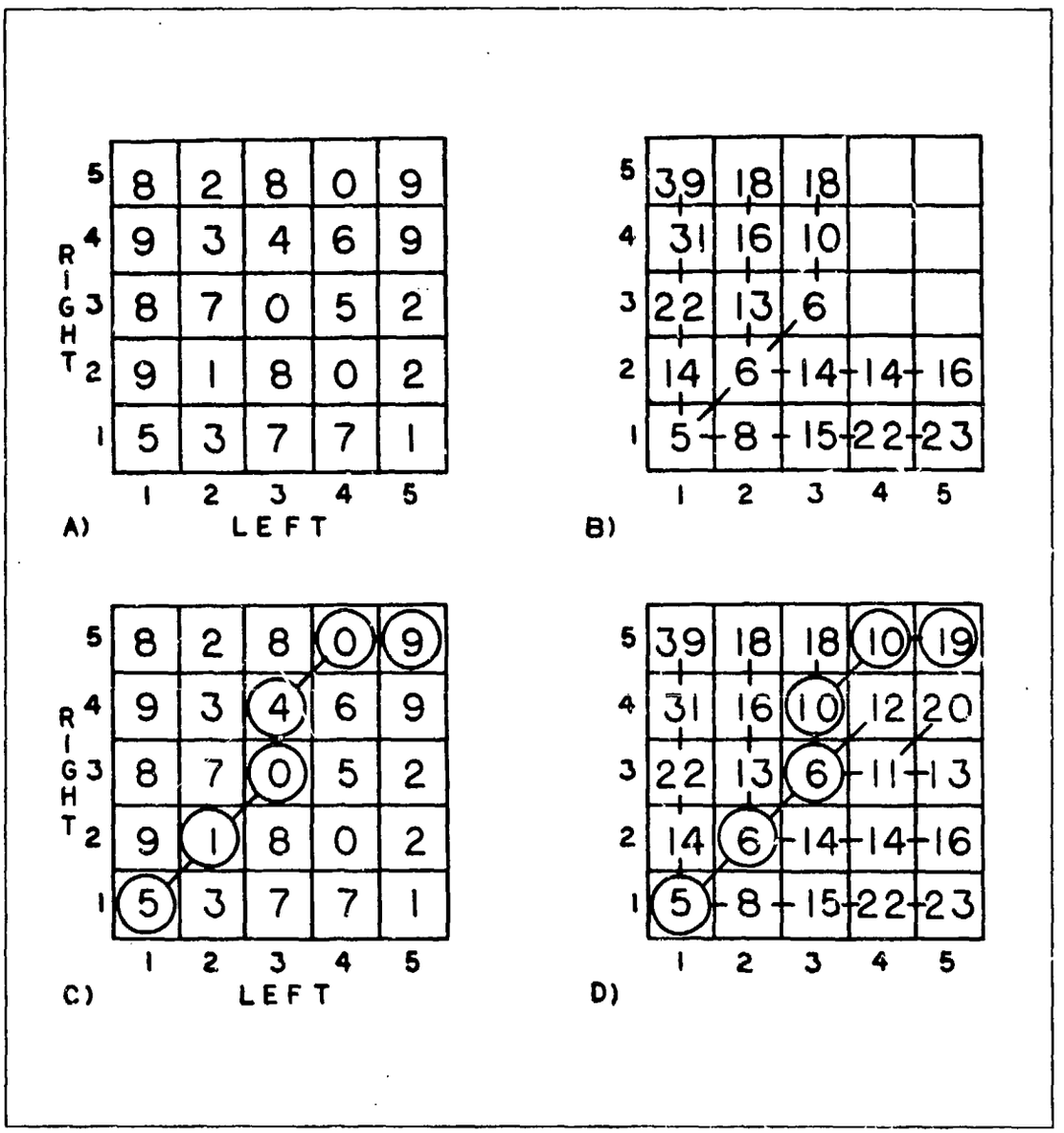


Figure 2-17: *The Viterbi algorithm finds the best path through a cost matrix such as the one illustrated in a. Each transition of the path may be up one unit, right one unit or both. A second matrix (shown partially completed in b) is constructed giving for each element the lowest cost of a path from the start (1,1) to that element. When the second matrix is complete (as in d), the entry in the top right corner gives the minimum cost, and the corresponding path can be traced backward.*

The Viterbi algorithm proceeds by constructing a second matrix, of the same dimensions as the first, each of whose entries is defined as:

> ***The accumulated cost of the lowest cost path from the starting node to the node corresponding to this entry.***

The matrix values are filled in ascending order, left to right and bottom to top, beginning at the lower left. The transition rules guarantee that when it comes time to fill an entry its three predecessors will already have been assigned values. The algorithm simply examines each of these predecessors, adds the cost for the current position (from the cost matrix), and selects the lowest sum. This sum becomes the value for the current entry, and a pointer is stored to indicate which predecessor was selected.

Figure 2-17b shows a partially filled matrix. The filling began with entry $(1,1)$, which, having no predecessors is simply assigned the corresponding cost matrix value of 5. For the rest of row 1 and column 1, two transition types don't apply, so only horizontal or vertical transitions, respectively, are used. In Figure 2-17b, the next entry to be filled is $(3,4)$, whose cost is 5. The algorithm compares $14+5$, $6+5$, and $14+5$, corresponding to vertical, horizontal, and diagonal transitions, and selects the second, filling in value 11 and a pointer back to $(3,3)$. Note that two or more predecessors may produce the same minimal score. If our purpose is only to discover *an* optimal path, we may choose any one of them to store as our pointer. The case of more than one optimal path is best handled by the Viterbi extension described later.

After the last position has been filled, the stored pointers are followed backward to the starting node, tracing out the optimum path from the upper right to the lower left. Figures 2-17c and 2-17d show the final path.

In applications dealing with very long or infinite sequences, it is possible to truncate the best paths to some depth σ [Forney 1973]. This corresponds to choosing a single node to represent the previous history of the sequence, and continuing to explore

all possible paths out from that node. In most cases, σ may be chosen such that the best path for each node under current consideration passes through the same node σ steps back. Thus, the graph may be truncated and no information will be lost. This technique limits the working matrix to a manageable size. However, our application has used only relatively short sequences, and we do not use a truncating Viterbi algorithm.

2.3.2 – Modifications for Stereo

We now discuss some modifications to the Viterbi algorithm to make it more suitable for the stereo matching problem. Because we must allow for occlusion, it is possible that certain sequence elements may have no match in the other sequence. Thus, we will use an algorithm with the following constraints:

- The sequences must be matched monotonically.
- Each element of a sequence is used at most once.

The question arises of how to assign a cost to an unmatched element. It certainly should not be zero, or the optimal path would be one where none of the elements of either sequence were matched. Instead of assigning an arbitrary high cost to unmatched elements, we have redefined the problem slightly. We replace the *cost* matrix with a *similarity* matrix and look for a path of maximum similarity, rather than minimum dissimilarity. Each matrix entry is a measure of how well two elements match, and unmatched elements may be assigned a zero score. A set of transition rules which implements this follows:

- Vertical or horizontal transitions of one unit indicate occlusion of the element whose row or column is being entered.
- Diagonal transitions of one unit indicate a normal match associating the elements belonging to the newly entered row and column.

Note that with these definitions an isolated node no longer represents a match; the type of transition leading to the node is part of the representation.

The algorithm is modified to select the maximum rather than minimum value for filling a new entry, and the cost for the current position is only added for diagonal transitions. We will illustrate this with an example shortly.

Finally, it is desirable to eliminate the restriction that the path always runs from the lower left to the upper right. It is possible that the first or last few elements of a sequence are unmatched. This corresponds to allowing paths to begin at any point in the first row or column and end at any point in the last row or column. We already have a mechanism for skipping unmatched elements (vertical and horizontal transitions), which is equivalent to the ending condition. The simple trick of adding a zeroth row and column allows the same mechanism to provide the beginning condition as well. Any entry in the first row or column may now be the effective start of the path since it may be entered on the diagonal from the zeroth row or column. The algorithm proceeds from the lower left to the upper right as before.

Figure 2-18 illustrates the modified Viterbi algorithm, finding the maximum scoring path subject to the above constraints. Figure 2-18b shows a partially filled matrix, where the next entry to be filled is $(3, 4)$, whose similarity score is 5. The comparisons to be made are $13+0$, $16+0$, and $13+5$, corresponding to vertical, horizontal, and diagonal transitions respectively. The maximum value, 18, is stored with a pointer back to $(2, 3)$. Figure 2-18c and 2-18d show the optimal path, traced back from the pointers. Note that the elements corresponding to row 1 and column 3 are not assigned a match.

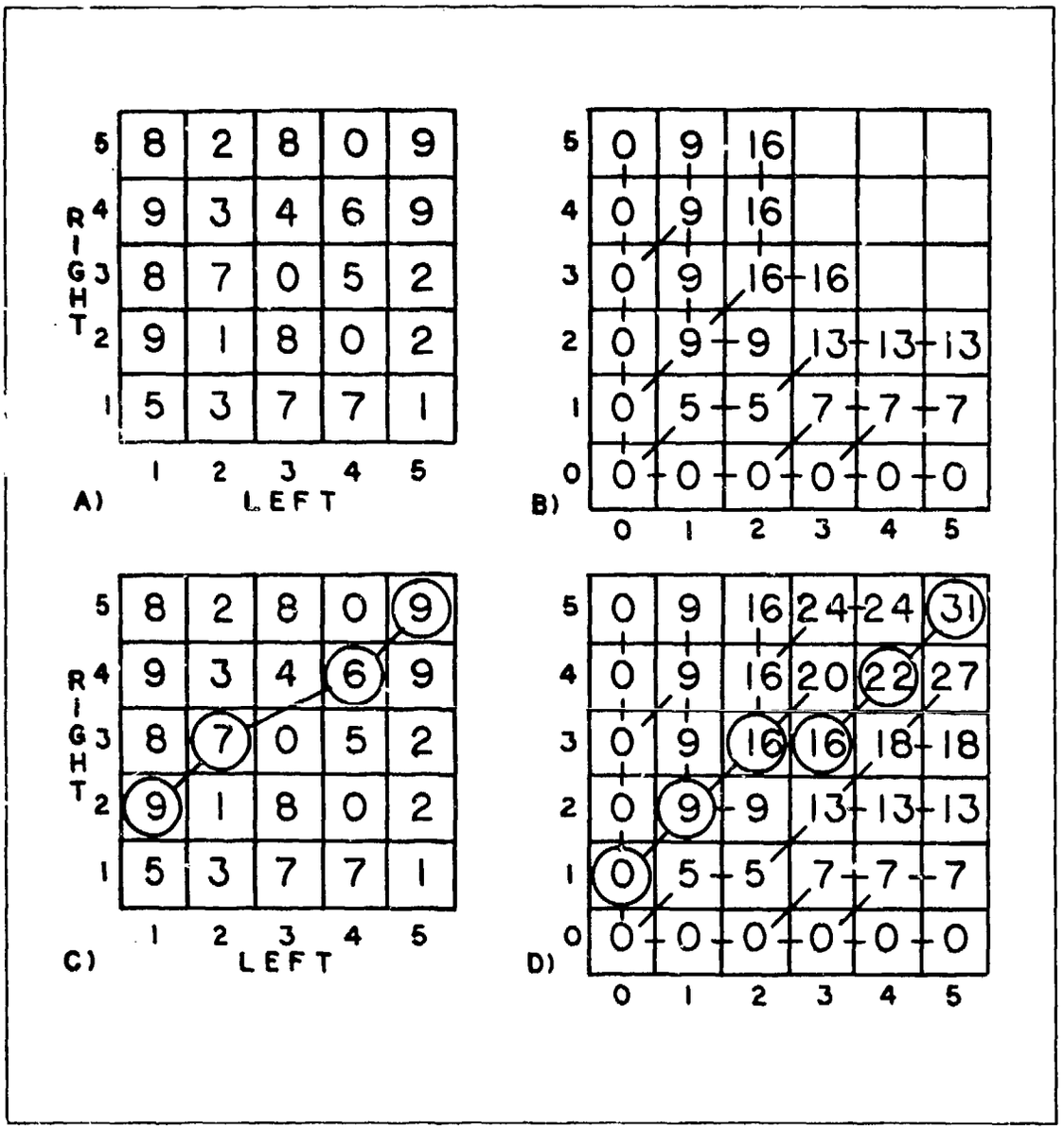


Figure 2-18: *The stereo Viterbi algorithm differs from that in Figure 2-17 in three ways. The cost matrix is replaced by a similarity matrix and the path of highest similarity measure is found. The same three transition types are allowed, but only diagonal transitions accumulate a score. A special row and column are added so that (1,1) need not be on the path. The partially and fully complete second matrices are shown in b and d. The matrix at c shows the four elements that were assigned stereo matches.*

2.3.3 – Extension to the Viterbi Algorithm

In the dynamic programming literature there are several algorithms for determining the "*k*-best" paths through an arbitrary directed graph. Hoffman and Pauley [Hoffman 1959] first published an algorithm whose application was finding the *k* shortest routes through the streets of Detroit. A conventional shortest path algorithm was run first to determine the best path to the terminal node from all other nodes in the graph. Alternate paths were calculated as ***deviations*** from this path. In other words, an optimal path was followed up to some node *A*, at which point a non-optimal branch (***deviation***) to *B* was taken. From B, the best path to the terminal node was followed. This process was repeated, as the third best path must be some deviation of the best or second best.

An improvement to this algorithm was published as part of a survey by Dreyfus [Dreyfus 1969], and this algorithm was itself subsequently improved by Fox [Fox 1973]. All of these algorithms produce one new path per iteration, each iteration requiring computation proportional to the number of nodes in the graph.

Any of these algorithms could be applied to the modified Viterbi algorithm just presented, since the Viterbi operates on what may be considered a directed graph, where each node has no more than three branches leading in and three leading out (vertical, horizontal, and diagonal). However, we have developed a more efficient algorithm that allows determination of all paths scoring within ϵ of the optimum, where ϵ is a threshold that may be chosen after the optimum is known. As discussed, the principal idea is to explore the alternate paths in addition to following the back pointers of the optimal path.

To permit this, the choices at every decision point in the algorithm are stored. The choices are represented by the partial path ***similarity*** scores for each of the possible predecessors at a node. These sums may be stored explicitly or they may be recalculated during the search. In the examples presented above, recalculation is easy from the matrix of partial paths. Similarly, the back pointer that selects the maximum choice at each

node may be stored or recomputed, since all the original information is present. The tradeoff is simply one of storage against time, since the recalculation may have to be done several times for each node.

Figure 2-19 illustrates the search algorithm. We assume that the modified Viterbi algorithm of the previous section has run on the data of Figure 2-18, and filled in the matrix of partial path scores. We require a stack with enough storage to hold all of the paths that score within ϵ of the optimal. This amount of storage will be the sum of the lengths of such paths, where length is in nodes, and a node is represented by an ordered pair, (row, column). One problem not addressed here is estimating the number of paths expected and hence the storage requirement.

The example has an optimal path score of 31, and we choose ϵ equal to 5; we want to find all paths meeting our constraints that score 26 or more. The stack will use three pointers: one, **TS**, is the usual top of stack, used for adding paths to the stack; the other two, **SB** and **SP**, are search pointers which will gradually work from the bottom of the stack to the top. Initially, the three pointers are at the bottom of the empty stack.

We initialize the stack by storing the optimal path, in reverse order. This path is determined in the usual manner by following the matrix back pointers. Along with the path are stored some additional data (actually stored in a separate index):

- The relative score of this path.
- A marker at the end of the path.
- A marker at the first node yet to be explored.

In Figure 2-19, these are represented respectively by a number next to the first node, a bracket, and an asterisk next to the appropriate node. For the first path in the example (the optimum), the relative score is 0, the path is seven nodes long, and the first node to be explored is (5, 5). After storing this initial path, pointers **SB** and **SP** are at node (5, 5), and pointer **TS** is one location beyond node (0, 0).

The main loop of the algorithm is:

Examine the choices at the node indicated by **SP**.

Increment **SP**.

If **SP** *encounters the end of path marker, then*

if this is top of stack, then done.

else move **SB** *to first node of next path and move* **SP** *to marked node of next path.*

Continue.

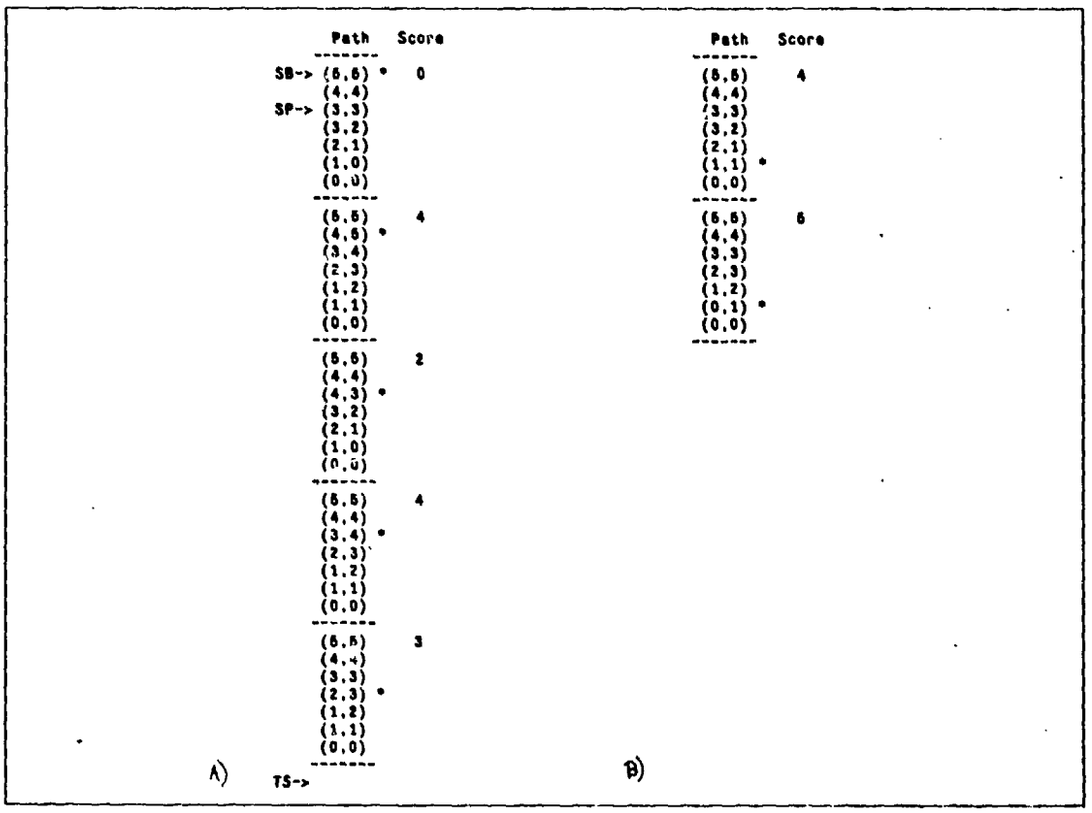


Figure 2-19: *The extension to the stereo Viterbi algorithm finds suboptimal paths. Shown here is the data structure used for the data in Figure 2-18. In a the stack is shown at a point part way through execution. The remaining entries are shown in b.*

Choices are examined in a loop, checking each predecessor, ignoring the one marked with a back pointer as the maximum. The procedure followed for each predecessor is:

Recalculate the threshold by which this predecessor was rejected (the difference between this partial path score and the score chosen as maximum).

Add that amount to the relative score of the path pointed to by **SB**.

If that sum exceeds ϵ, *then done.*

Else store this suboptimal path.

The procedure to store a suboptimal path is:

Copy the path from **SB** *to and including* **SP** *onto the top of the stack.*

Store the predecessor currently being considered onto the stack.

Mark this node with an asterisk.

Continue storing nodes on the stack by following the back pointers of the matrix until node $(0, 0)$ *has been stored.*

Store an end of path marker.

Record the relative path score as the sum determined above (the one $\leq \epsilon$*).*

In Figure 2-19, the full stack is shown, i.e., after all 7 paths are found, but the pointers are shown for the state where the 5th path has just been pushed on the stack. Figure 2-20 shows the 7 paths and their scores. After the search is complete, the suboptimal paths on the stack may easily be sorted by relative score.

Note that each path pushed onto the stack consists of three parts: the ***back end*** (higher coordinates) which is always identical to some previous subpath; the alternate choice transition, or ***deviation***; and the ***front end***, which is given by the matrix pointers. The ***back end*** carries a penalty given by the relative score stored with the path from which it was copied. The alternate choice transition carries its own penalty, just calculated. The ***front end*** is an unexplored subpath, but its score is optimal because it is determined

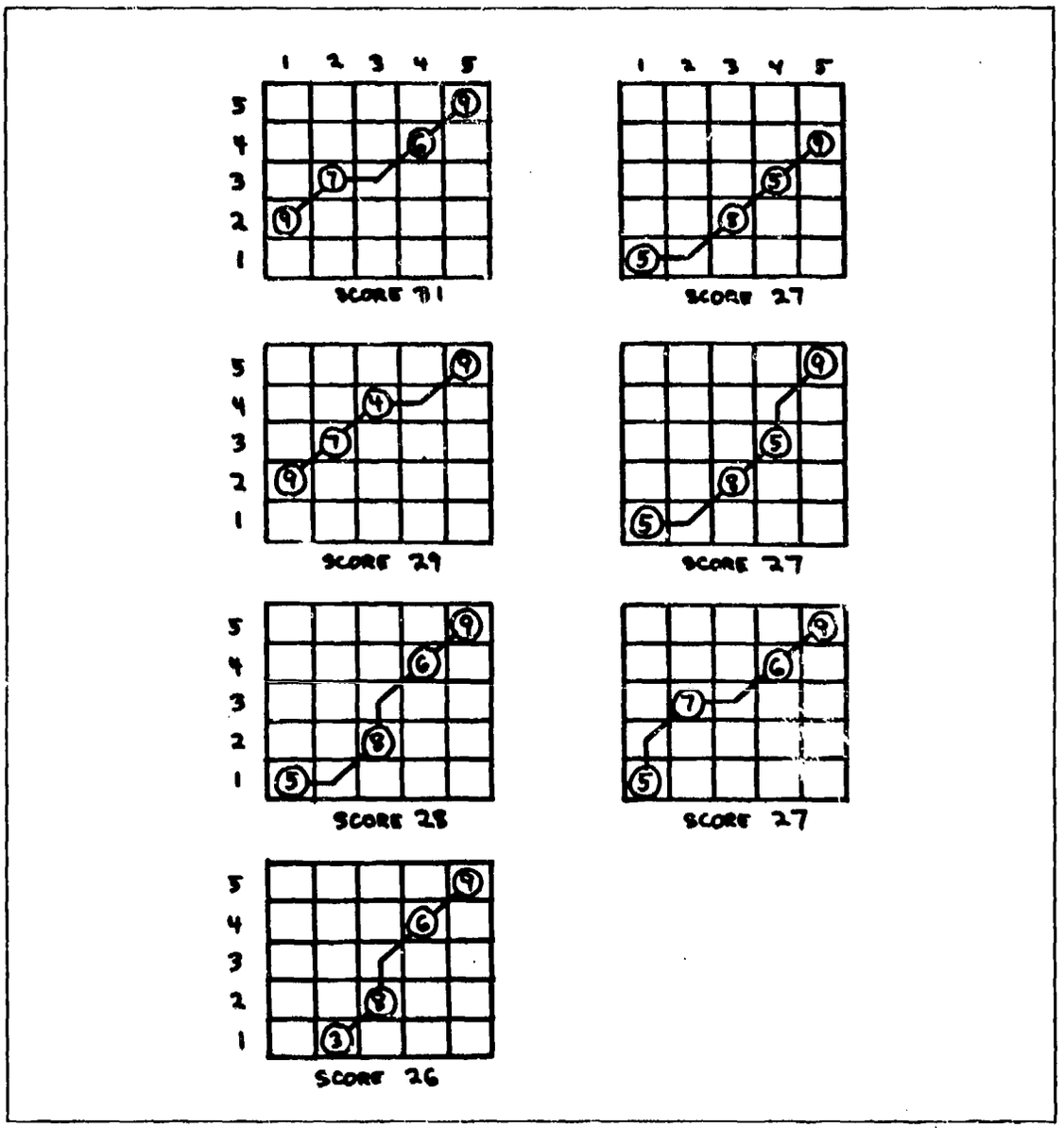


Figure 2-20: *From the data in Figure 2-18, seven paths can be found with scores of 26 or greater.*

by the back pointers in the partial path matrix. Thus, the sum of the ***back end*** score and the current alternate choice transition gives the relative score of the new path. The mark represented by the asterisk ensures that only the ***front end*** of the new path will be subsequently explored.

The algorithm finds all paths whose score differences are less than or equal to ϵ, and because it examines only those paths, it is efficient. All copying of paths is done with a destination pointer of **TS**, which is incremented after each node is copied. Thus, the total storage required is equal to the total length of all the paths found. Also, the examination of each node requires a constant number of comparisons, and for each node examined an entry is made on the stack, so the computation time will be proportional to the total length of all paths found. This is of order kN, where k is the number of paths found and N is the shortest path length.

Although our algorithm has only been applied to the results of a Viterbi algorithm, it could be extended to work on a generalized directed graph. The principal difference between our algorithm and published "k-best" algorithms is that ours finds ***all*** paths within ϵ of the best. There is no way of predicting how many paths will be found when ϵ is chosen; there is also no guarantee as to the order in which paths will be found. However, if a given ϵ results in k paths, computation proportional to kN will have been done, rather than kN^2 as in the other algorithms. Note that we use N here to represent the length of an input sequence, rather than the number of nodes in the graph, which is N^2 by our definition.

Our algorithm produces suboptimal paths only between the terminal node and an initial node, whereas k-best algorithms generally produce paths between the terminal node and all other nodes. Both algorithms require ***per node*** storage proportional to the maximum number of branches into any node. For our Viterbi this is only 3, but in general, it would be equal to the number of nodes, N^2. Finally, the k-best algorithms all have problems dealing with ties, *i.e.* disjoint paths having the same score. This is usually solved by *perturbing* each branch value with a small random number. Our algorithm has no such difficulty since it is not attempting to order the paths found.

2.3.4 – Application to Stereo

To discuss the application of the extended Viterbi algorithm to stereo matching, we need to introduce the data structure, the evaluation function that computes the similarity measure, and the transition rules for stereo.

There are two choices for data structure: surface-based and edge-based. Since surfaces (intervals) and edges occur alternately within a sequence, they are essentially equivalent for one-dimensional matching. For our implementation, we have chosen to represent the nodes as intervals; surface descriptions are the ultimate goal, and intervals are closer to that than edges. As we will explain later, we have not yet been able to produce a good surface-based data structure for two-dimensional matching, so the choice of intervals in the short term may not be best.

Each row of the matrix will be assigned to each interval in the right image sequence and each column will be assigned to an interval from the left image. As discussed in a previous section, intervals may be classified into six groups according to visibility conditions. Each entry in the dynamic programming matrix, will be broken down into six *subnodes*, each carrying a different interpretation for that portion of the path. Subnodes are identified by an ordered triple: (row, column, subnode type).

The transitions between subnodes are limited to those allowed by the occlusion constraints defined previously. These, together with the coordinate system based on edge rays, define a space of *allowable paths*. Subject to the original assumptions, only physically realizable profiles are allowed. That is, for every allowable path, there is a continuous, connected profile of straight line segments that will result in the observed left and right projections.

Figure 2-21 illustrates this space of allowable paths. In the diagram, each hexagon represents a node; the numbers within the circle represent the allowable subnodes at that

position. Some subnodes are disallowed due to windowing effects. The basic transitions proceed up, down, and horizontally to the right, corresponding to vertically, horizontally and diagonally in previous rectangular grids. Given a subnode, a transition out of that node is allowed only across a hex side labeled on the left with the current subnode number. A transition across a hex side must terminate in a subnode whose number appears in a corresponding place on the right side of the hex side. For example, (1, 2, 4) may precede (1, 3, 3) or (2, 3, 5) but not (1, 3, 4) or (2, 3, 1).

Thus, occlusion constraints serve to reduce the search space from what it would be if transitions were allowed between all subnode types. The rest of the constraints

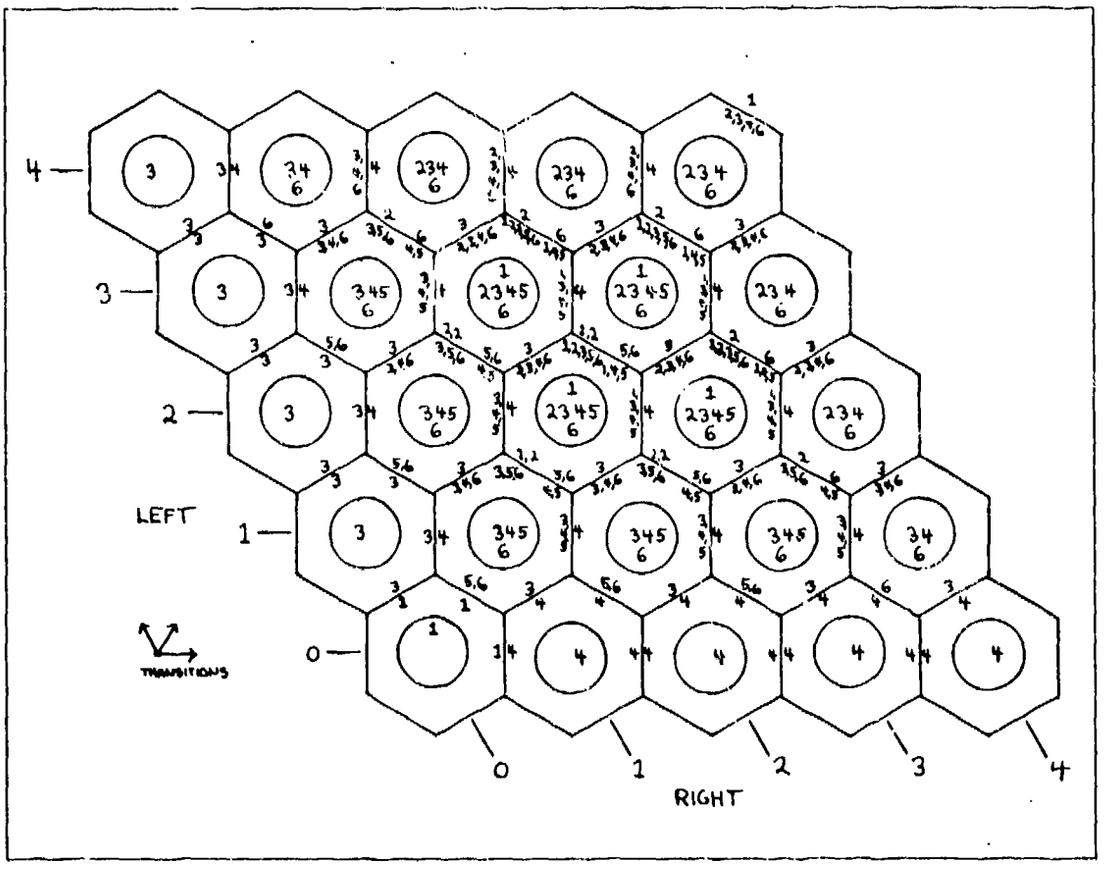


Figure 2-21: *This diagram combines the transition rules for the stereo Viterbi algorithm with the stereo zone and lattice and the occlusion constraints discussed earlier.*

are incorporated into the ***evaluation function***, which serves as the similarity function in the previous examples. We do not understand exactly how to construct this function, but from experiments, the performance of the dynamic programming algorithm on real stereo data seems to be fairly insensitive to minor changes in the function. The principal components of the function are the edge measures, angle and length, and the surface measures, brightness and interval ratio, discussed in a previous section. Each component is normalized to a range of 0 to 1, weighted and combined to give a score for each node. The strongest constraint get the highest weight. The dynamic programming algorithm maximizes the *sum* of the individual node scores.

We have used both additive and multiplicative combinations of constraint measures at each node, and have had success with both types. The addition of linear measures gives a low score only if all the components are low, while multiplication gives a low score if any component is low. We currently multiply related measures (e.g., edge angle and extent) and add independent groups (e.g., edges and intervals). The evaluation function is discussed in a later section.

The evaluation function depends also on the transition and subnode types. For subnodes corresponding to occluded surfaces (visible only to one camera), a default measure must be used, since there are not two intervals to calculate a ratio or brightness comparison. The default value is currently the approximate probability of a surface being self-occluded, which is a function of the camera model. Similarly, a default is used for edges visible to only one camera.

Some *ad hoc* measures have been used experimentally to favor profiles that are globally simpler. Long intervals of types 3 or 4 correspond to drastic altitude changes in the profile. Two different methods have been tried to penalize profiles containing these types of intervals. One method is to simply subtract an amount proportional to the

interval length for these types. Another is to calculate ***excess length.*** This last has the advantage of being applicable to matched intervals as well. It is defined as follows:

Sum the interval lengths, both if a match (types 1,2,5,6), otherwise just the one (type 3,4).

Calculate the minimal length surface in a profile whose projected lengths add to the above sum.

Subtract this minimal surface length from the profile length calculated from the actual projections.

Thus we try to minimize the length of the profile, compared to its projected length. This favors smooth scenes over jagged ones (see Figure 2-22).

A second *ad hoc* measure is a penalty for surface breaks. Whenever a node is chosen for a path, we are constraining the slope of the underlying profile surface in some

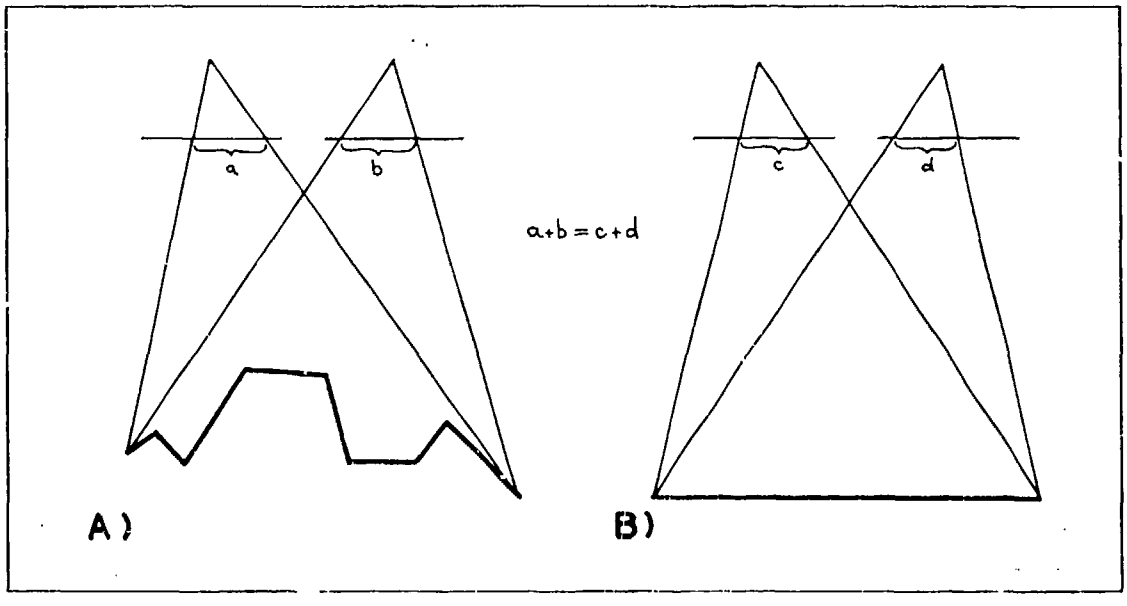


Figure 2-22: *A typical profile is jagged as in a. A minimal flat surface, b, can be found whose total projected length in the images is equal to that of profile a. The difference between the total length of all the surface intervals in a and the length of the surface in b is excess length.*

way. Some nodes constrain it fully (type 1), others allow some freedom. Each time a node is evaluated, the slope constraints are checked. If the new constraints require a slope discontinuity in the profile, a penalty is added. If the constraints allow the underlying profile segment to have a continuation of the previous segment's slope, no penalty occurs. This favors surface markings over surface discontinuities, and profiles with fewer surfaces over profiles with many.

2.3.5 – Conclusion

The principal advantage of the dynamic programming stereo matching is its ability to combine most of the geometric constraints we have investigated with a strong global consistency – at least global in the sense of the one-dimensional problem. The resulting profiles are guaranteed to make geometric sense over the entire epipolar line. That is, they can be constructed from a connected sequence of line segments and an edge is present in an image if and only if a corresponding junction of two segments is not occluded from that camera. We rely on the the evaluation function to select only the best matches from among the many possible profiles.

The modified Viterbi algorithm is also efficient. If n is the average number of elements in the sequence, the average path length is of order n. Since there are a constant number of choices at each node of a path, the total number of possible paths will be exponential in n. The Viterbi algorithm, however, evaluates these in time and space proportional to n^2. As noted, the time and space complexity of the search for suboptimal paths is linear in the total length of output paths.

The algorithm is required to "explain" every element in each sequence; an element either matches another, or it is occluded. However, this can be a disadvantage when the input data have missed or extraneous features. These may result from edges near threshold, movement in the scene between successive views, or inaccurate camera models.

This algorithm does not account for such imperfect data. For example, instead of ignoring an extraneous edge, it tries to distort the profile to occlude it from the other view (see Figure 3-11). Similarly, distortions are introduced to explain missing edges by providing an occluding surface.

We have made some attempts to develop an algorithm which could automatically edit out obvious errors. The number of subnode types could be increased to represent erroneous data points. This would allow the dynamic programming algorithm to additionally assign paths that interpret features as missing or extraneous. However, this would require a more complex evaluation function and would increase the number of transition types between subnodes. The storage required to retain all the decision points then increases as the square of the number of subnodes. This added complexity would have made it impractical to retain the feature of recovering suboptimal paths.

We note that the most common source of errors in an epipolar line match has been alignment failures near the terminations of extended edges. The epipolar line in one view may just miss a corner that intersects in the other. In such cases, the error disappears in adjacent epipolar lines. Also, experiments show that the effect of errors tends to be localized. Rather severe profile distortions may be required to "occlude" an extra edge, but one or two elements farther along in the sequence, the profile is undisturbed. This is because any radical distortions caused by the error tend to be the same in all paths, so all paths are equally penalized and their relative ranking is unchanged. For these reasons we have decided to postpone the problem of missing or extraneous edges to the two-dimensional matching stage, and to try to filter it out there.

2.4 Continuity and Consistency

Except for accidental alignments and occlusions, continuous edges in a scene will project to continuous edge curves in an image. We define edge curves A and B in an image to be ***continuous*** if there is a sequence of edge curves beginning with A and ending with B where each adjacent pair of edge curves meet at a vertex which is not a "T-junction" (see Figure 2-23). The continuity constraint, then, is that edge curves which are continuous in one image cannot match discontinuous edge curves in another image. This constraint can be used to resolve matches that are ambiguous in a small context (see Figure 2-24) and has been used in earlier stereo systems.

In 1978 we reported [Arnold 1978] results of a stereo system using what we then termed *local context* to resolve ambiguities. This system worked from the unlinked *edgel* output of the Hueckel edge operator and used constraints based on edge angle and brightness measures from the operator. The search space was limited by measuring the

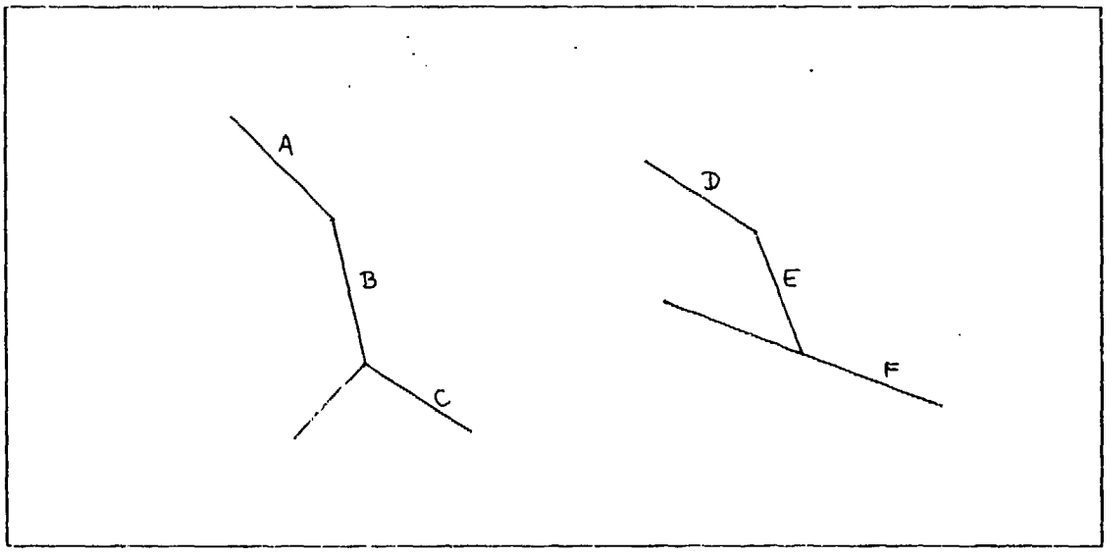


Figure 2-23: *Continuity in the image implies continuity in the scene. "T-junctions", however, usually imply a discontinuity in the scene. Thus, A and C are continuous while D and F are not.*

camera model as described earlier, and using epipolar geometry. For each *edgel*, a list of possible matching edgels was produced (occlusions were not considered). This list was filtered by a continuity constraint. Continuity was calculated in each image by linking edgels that were approximately collinear; the constraint required the stereo disparities of two linked edgels to agree.

This early system suffered from some serious problems, many of which resulted from the quality of data produced by the edge operator. However, continuity turned out to be a surprisingly strong constraint, and the system produced some stereo maps that clearly separated scene objects from the ground and showed structure within the objects. A more detailed summary of this work has been included as an appendix to this thesis.

While continuity is a strong constraint, it does not always apply in its simple form. For example, Figure 2-25 shows a case where edge curves on two epipolar slices are continuous in one view, but do not have a corresponding pair of continuous curves in the

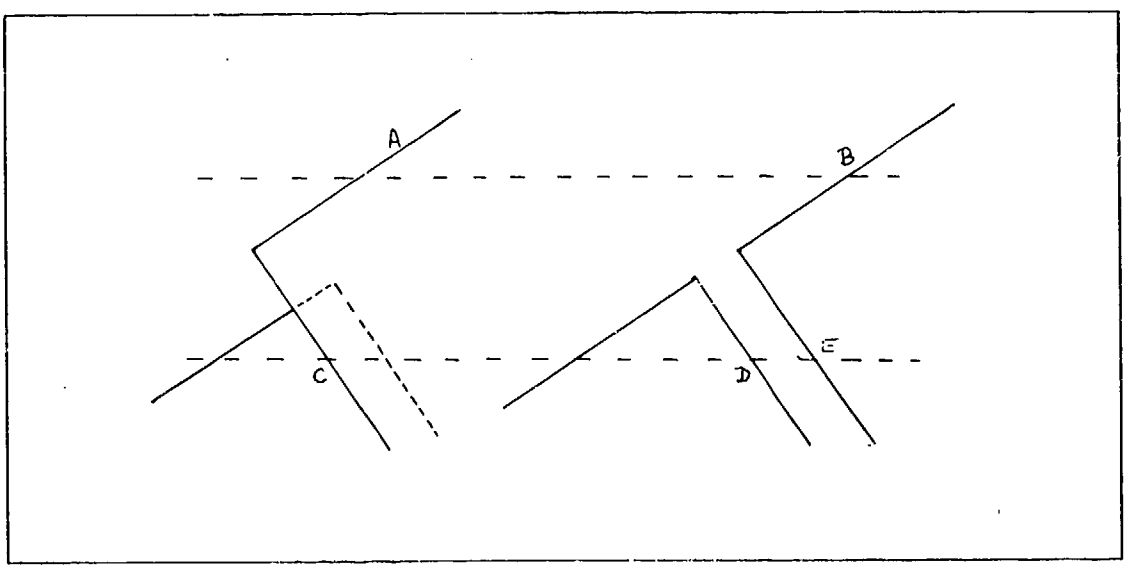


Figure 2-24: *Scene edges will not appear continuous in one stereo image and discontinuous in another. In this example, edge curves A and B on the first epipolar line match unambiguously. On the second epipolar line, C may match with either D or E. The continuity constraint resolves this ambiguity, since A – C and B – E are continuous while B – D is not.*

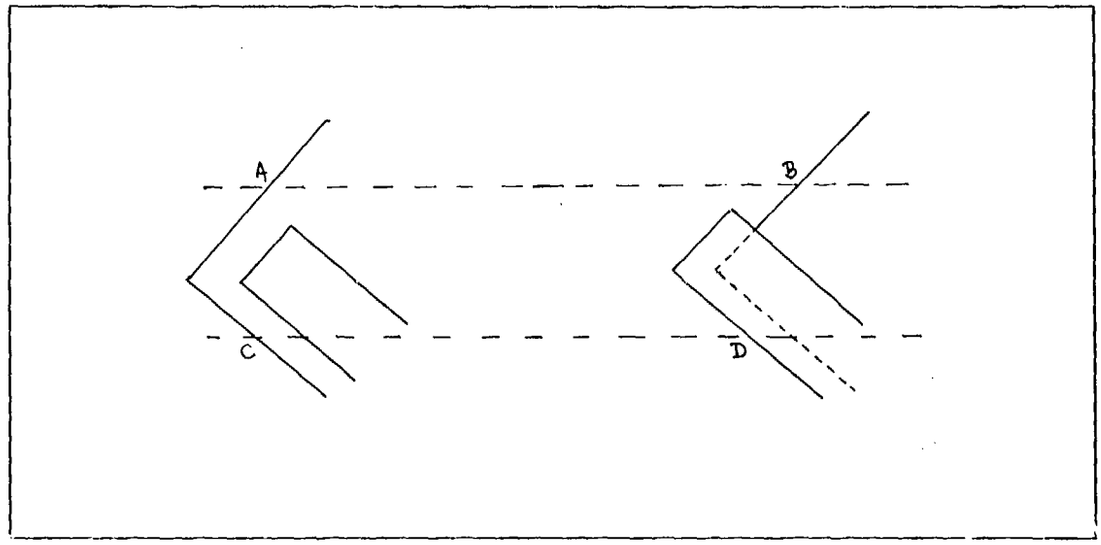


Figure 2-25: *The continuity constraint provides negative evidence for the match of C with D. To express this positively, we say that the stereo interpretation of C occluded by D is consistent with a match of A with B.*

other view. The failure to find a match for edge C should not reduce our certainty for the match of A with B. On the other hand, an attempted match of C with D may make sense locally (i.e., on the lower slice), but should be rejected by the continuity constraint, since B and D are not continuous. Thus, the interpretation of C as occluded by D is *consistent* with the interpretation of a match for $A-B$.

The problem comes in recognizing *consistency* conditions. *Continuity* is easily checked, but more analysis is needed to characterize *consistency*. We make use of some simple cases in our implementation, but leave a complete analysis to future work.

IMPLEMENTATION

3.1 Producing the Data

The stereo system we describe here operates on an input file consisting of edge and intensity data. The intensity data are taken from digitized photographs while the edges are produced interactively with a computer drawing program. It is anticipated that advances in edge-finding and segmentation techniques will allow this process to be fully automated soon.

3.1.1 – The Images

In each of the examples, we begin with a black and white stereo image pair: two digitized images of the same scene from different viewpoints. The images are from 128 to 512 pixels on a side, and from 6 to 8 bits per pixel. Typically, the overlap permits 60% or more of each image to be viewed in stereo. The camera model is known imprecisely or not at all and must be calculated from the images. Part of this calculation is done by hand and part with computer aid.

The digitized images we use include actual aerial photographs with subjects such as aircraft at a terminal, and artificial data, where the subject is a simple block model of a city (see Figure 3-1). In aerial photographs, the camera is typically mounted in an aircraft to look straight down and the two photographs are taken at different points in time; the flight path of the aircraft determines the stereo baseline.

Except for the artificial data, the two images are usually not in perfect registration, and must be adjusted before processing. Furthermore, professional aerial photographic film is very large (nine inches on a side) and only a small portion can be digitized for our experiments. The selection of a digitization window in each image is done by hand, usually with the goal of maximizing overlap in an interesting portion of the scene. This process introduces further uncertainties in image registration.

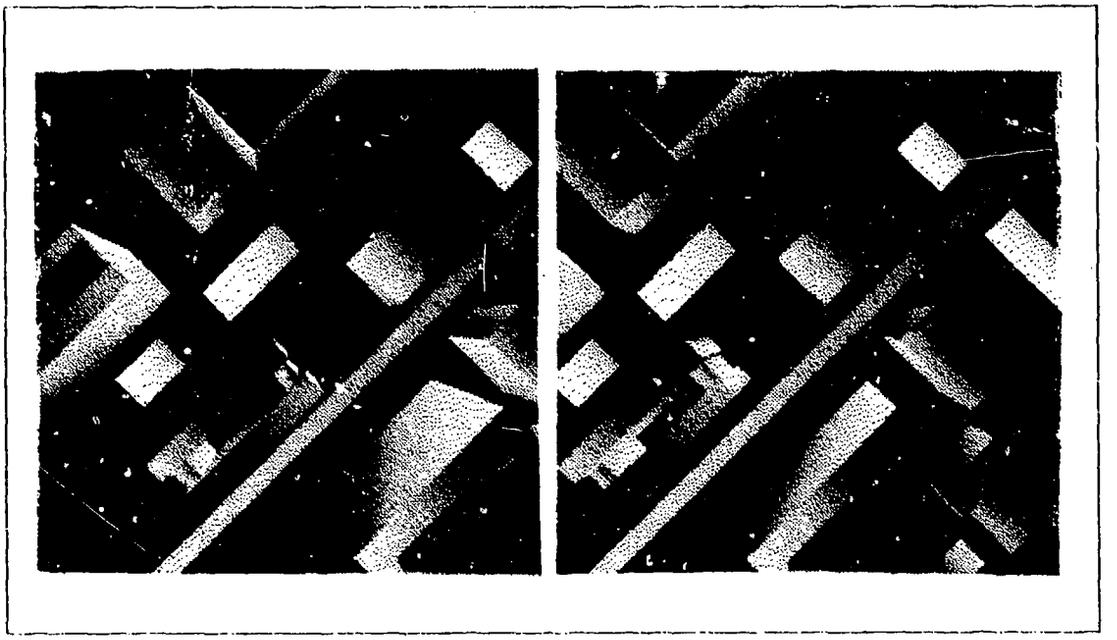


Figure 3-1: *Artificial images of a block model city were provided by Control Data Corporation.*

3.1.2 – Determining the Camera Model

In the aircraft images, the registration of the two images was only approximate. We used a technique described in an earlier paper [Arnold 1978] (and in the appendix) to calculate the parameters required for more precise registration:

- orientation of the stereo axis
- relative rotation
- relative scale factor
- relative translation perpendicular to the stereo axis

The choice of these four image-based parameters is more suitable than camera-based parameters (e.g., pan and tilt) for aerial photographs, where the depth range of the subject is very much smaller than the altitude of the camera.

The parameters are calculated by establishing stereo correspondence on a non-coplanar set of four or more points, and executing a least squares algorithm. Programs by Moravec [Moravec 1980] and Gennery [Gennery 1980] are used to choose the set of points automatically, to do the stereo correlation and to solve for the parameters. Once the images are registered, the remaining camera parameters are calculated (see Figure 3-2). X_L, Y_L, X_R and Y_R are calculated from the digitization window location, and B and f are calculated as explained below.

For example, consider the images of the blocks scene, which were obtained from Control Data Corporation. These data are artificially produced, so the images are already registered. There is no relative rotation, translation or scale factor between the two images

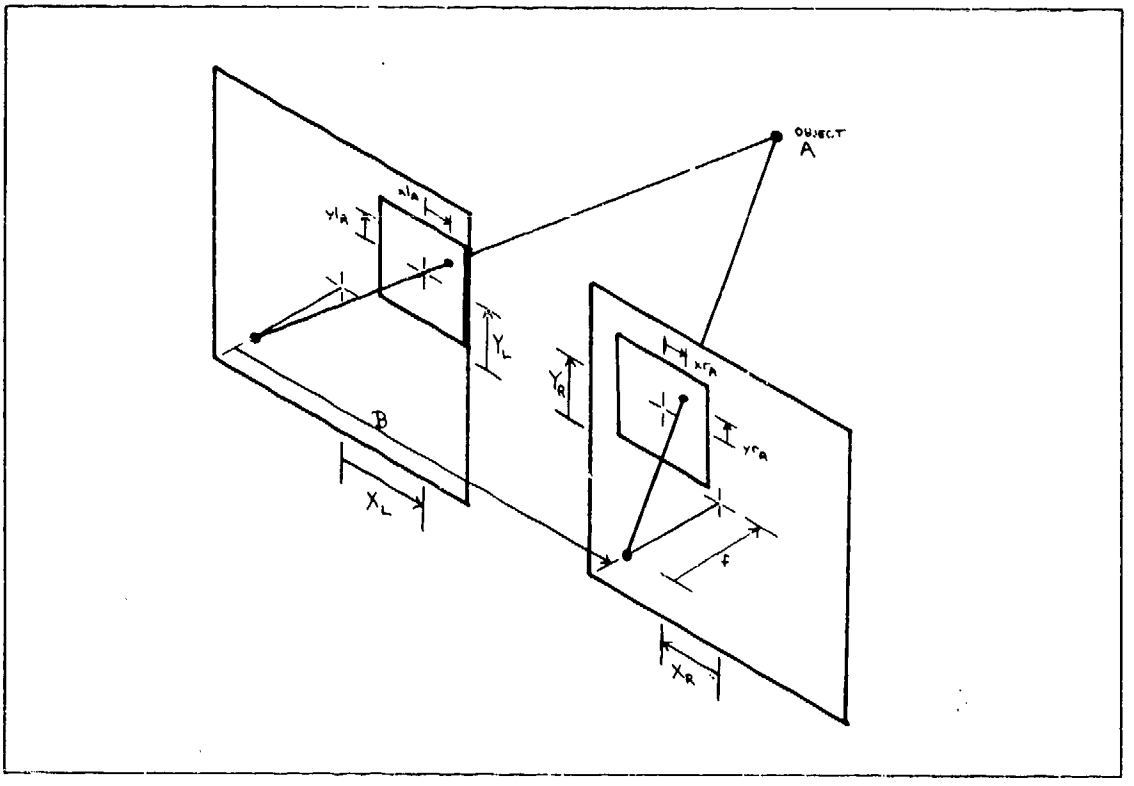


Figure 3-2: *The normal camera model for our stereo calculations is based on data taken from aerial photographs. We assume the cameras are aligned as shown and that only part of each image is digitized for processing.*

and the stereo axis is parallel to the x-axis in the image plane. The optical axes of the cameras pass through the center of each image, so $X_L = Y_L = X_R = Y_R = 0$. However, the image distance, f, and stereo baseline, B, are not known in advance.

First, we loosely define a ground plane as a plane in the scene parallel to the film plane, in or before which many features lie but beyond which few if any features are visible. In the case of aerial photographs over flat terrain, this corresponds with the actual ground surface. If A and B are ground plane features in the scene with actual x-coordinates X_A and X_B, then their corresponding image coordinates are xl_A and xl_B in the left image and xr_A and xr_B in the right image (see Figure 3-3). Since the ground plane is parallel to the image plane, $xl_A - xl_B = xr_A - xr_B$. If from "ground truth" information we know the actual distance between A and B in meters, then we can determine a mapping scale factor, m:

$$m = \frac{X_A - X_B}{xl_A - xl_B} \qquad (3-1)$$

where:

m is in meters per pixel.

This mapping scale factor may be calculated from either the left or the right image. It will convert any distance in the chosen ground plane to a distance in the image plane.

From the camera geometry and similar triangles we can see that $m = Z/f$, so we can solve for the baseline B:

$$\frac{B}{Z} = \frac{d_A}{f}$$

$$B = \frac{Z}{f} d_A = m d_A \qquad (3-2)$$

where:

$d_A = (X_L + xl_A) - (X_R + xr_A)$ is the ***stereo disparity*** in pixels, and

B is the baseline in meters.

Note that knowledge of both xl_A and xr_A requires a correspondence between a left image point and a right image point representing a feature in the ground plane. This, together

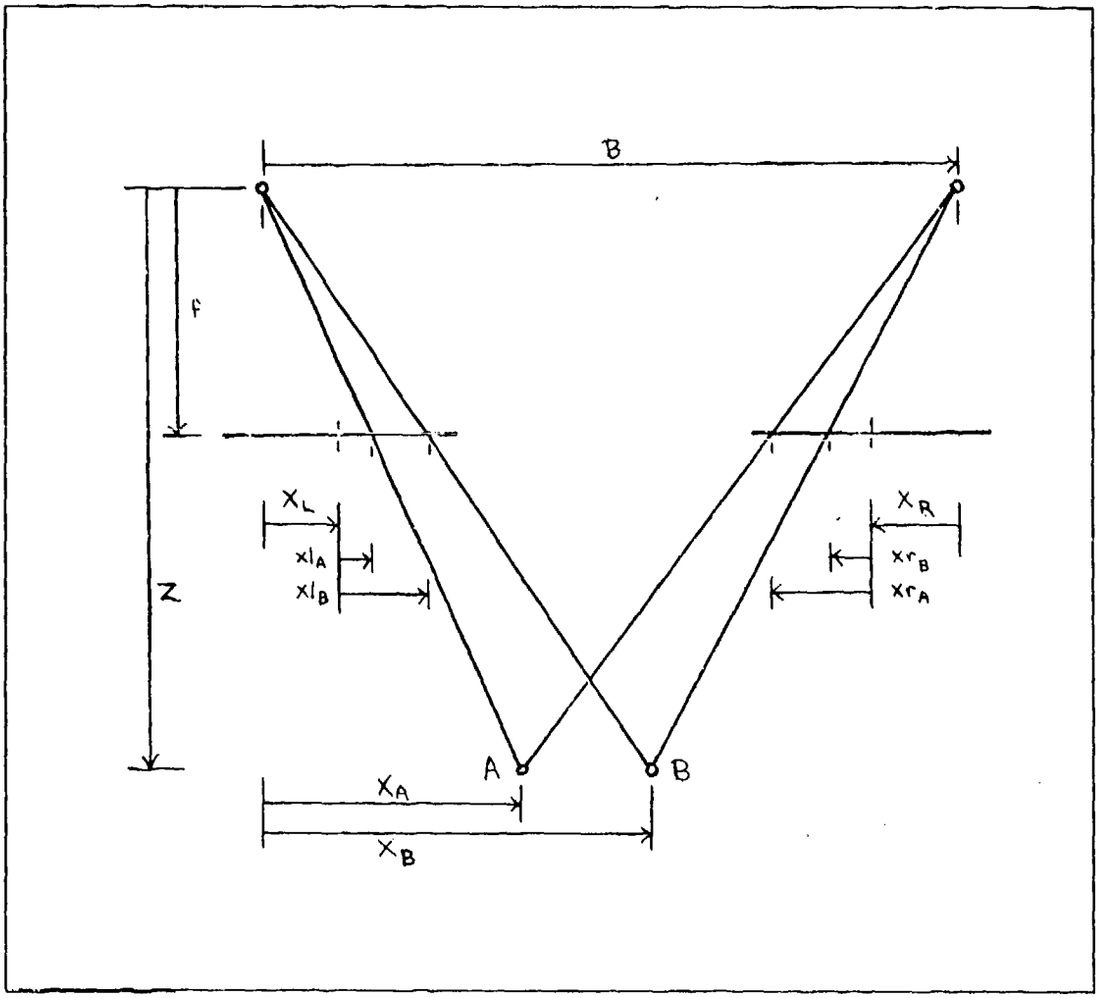


Figure 3-3: *These projections can be used to determine stereo baseline (B) from ground truth information (X_A and X_B) and image distance (f) from camera altitude (Z).*

with the mapping scale factor, allows calculation of the stereo baseline. (If m were not known, the baseline could be expressed in pixels.)

The image distance, f, will often be known, since it is a simple function of the camera lens and film size, but it is interesting to note the conditions required to calculate it. Just as horizontal ground truth is required to calculate the baseline, vertical information is necessary to calculate f. If the distance from the camera to the ground plane, Z, is known then:

$$f = \frac{Z}{m} \tag{3-3}$$

where:

f is in pixels, and

m is the mapping scale factor defined above.

Thus f can be determined from the "altitude" of the camera, Z, plus horizontal ground truth. (More commonly, Z and f are known and are used to determine m.)

If Z is not known, then the height of a known object in the scene can be used (see Figure 3-4). If two points A and B in the scene differ in their distance from the image plane by $h = z_A - z_B$ meters, and stereo correspondence can be established for both points, then:

$$f = \frac{h}{B}\left(\frac{d_A d_B}{d_B - d_A}\right) \tag{3-4}$$

where:

B is the stereo baseline,

$d_A = (X_L + xl_A) - (X_R + xr_A)$ is the stereo disparity for object A, and

$d_B = (X_L + xl_B) - (X_R + xr_B)$ is the stereo disparity for object B.

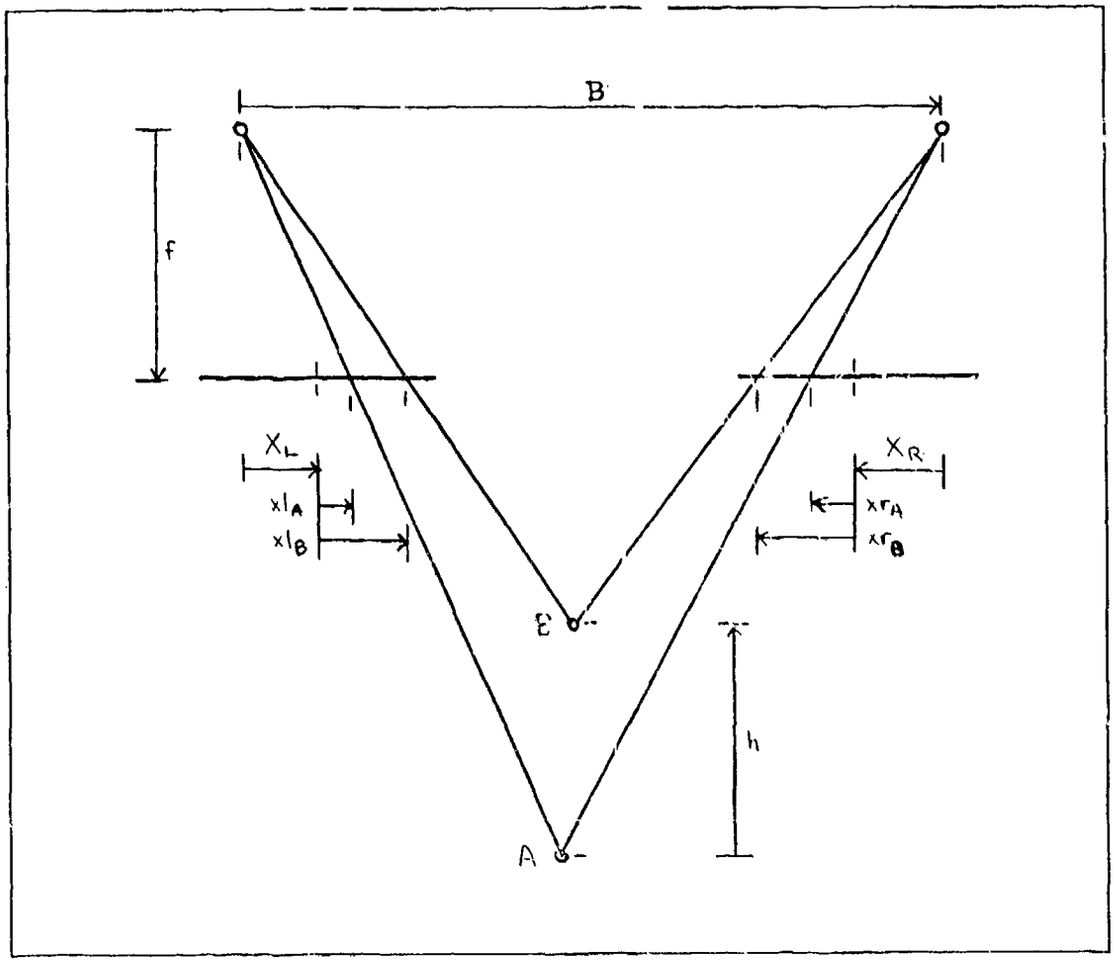


Figure 3-4: *These projections can be used to determine image distance (f) from vertical ground truth (h).*

3.1.3 – Edge Detection

Research on edge detection, linking and segmentation is proceeding at Stanford [Marimont 1982] and elsewhere and promises to supply fairly clean line drawings from real scenes in the future. In the meantime, we have chosen to derive our edge information by hand with a computer's aid. The technique is to superimpose straight line drawings from the DESIGN [Lowe 1982] program on a grey-scale display of the image data. The drawing is adjusted by hand until the superposition of all prominent edges looks accurate.

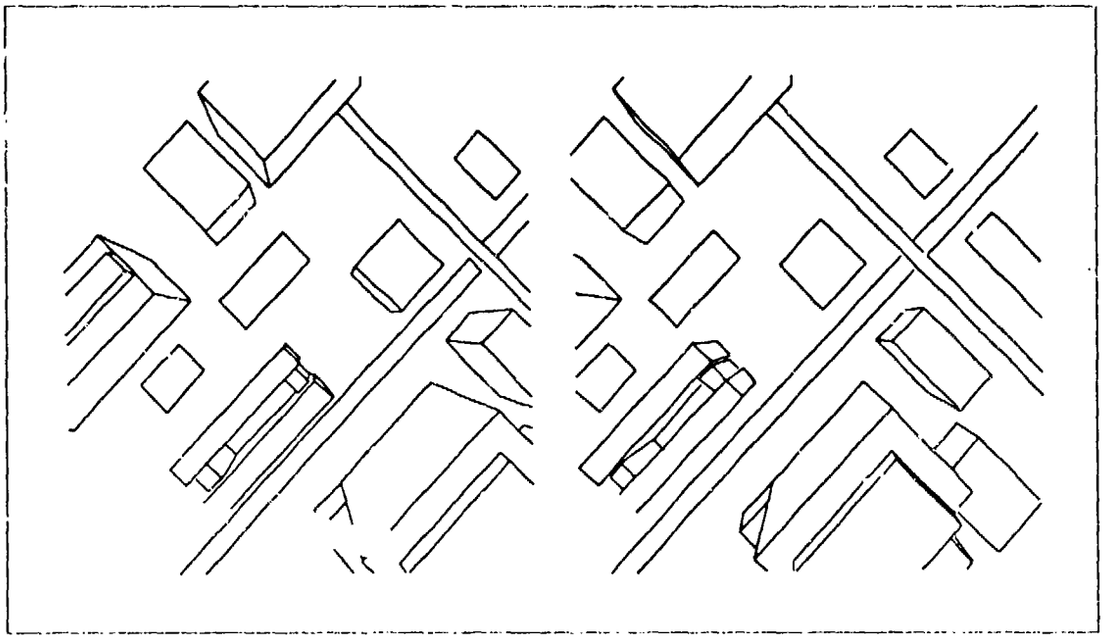


Figure 3-5: *Edge data is derived by hand from digitized images with the help of a line drawing program. These edges are intentionally imperfect.*

Since these data are intended to reflect the expected performance of future edge segmentation programs, care is taken to avoid using higher level human visual functions. Edges are not extended into ambiguous or low contrast areas. Left and right images are derived independently, so some edges are "detected" in one view, but not in the other. The information in the vicinity of corners or intersections is often omitted, so surface boundaries need not be closed. Even edge data from the blocks scene, while derived from noiseless artificial images, is not a perfect line drawing (see Figure 3-5).

3.1.4 – Preprocessing

The edge data are read by a program that takes the edge information together with the original images and writes a file containing structured input data for the stereo system. Figure 3-6 illustrates this data structure.

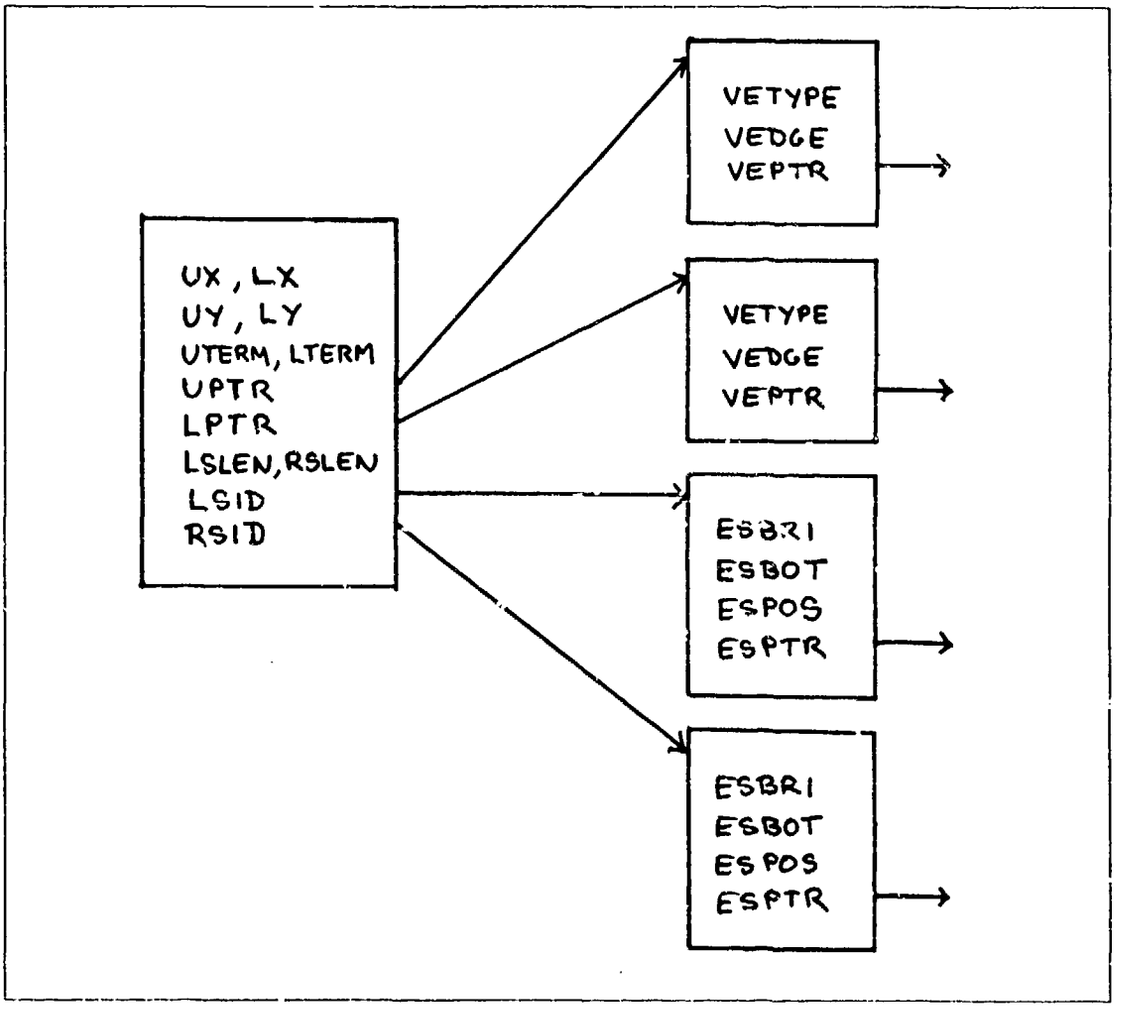


Figure 3-6: *Line drawing data are read into a structure that relates each edge to its neighbors. Information on the edge endpoints includes their coordinates, the type of vertex and a list of other edges belonging to that vertex. Information on the sides of the edge includes a list of T-junctions that segment the side, their positions, and the average image brightness adjacent to each segment.*

Edge data consist of pairs of endpoints, each with an x- and y-coordinate in the transformed stereo coordinate system. This system has units of pixels, a right handed coordinate system with its origin at the image center and a stereo axis running left to right along the x-axis. A record is created in the data structure for each edge segment. This is done separately for left and right images.

An *edge record* comprises the following:

UX,LX	x-coordinate of the upper and lower endpoints
UY,LY	y-coordinate of the upper and lower endpoints
UTERM,LTERM	number of edges in upper and lower vertices
UPTR,LPTR	pointers to upper and lower vertex record lists
LSLEN,RSLEN	length of left and right side record lists
LSID,RSID	pointers to left and right side lists

Each edge record is compared with every other edge record to determine the vertices in the image. A vertex is the intersection of two or more edge segments in the image. When checking for intersections, each line is extended by a given amount in order to compensate for data lost near corners. Thus, edge segments that approach within a threshold but don't touch in the input data will be analyzed in subsequent steps as if they intersected.

As each vertex is examined, it is classified as a termination if it is within a threshold of the end point of both lines, or as a "T" if it is near the end point of only one line. "X" intersections, that are not near any end points, are rare and are ignored at present. (They may be handled by breaking one or both edge segments into two pieces.) The information from each vertex is stored in the data structure as a ***vertex record***, linked to either the upper or lower end point of the edge segment. A ***vertex record*** comprises the following:

VETYPE	type of termination

VEDGE	pointer to edge record
VEPTR	pointer to next vertex record

In addition, "T" junctions generate *side descriptor records* linked to either the left or right side of the edge segment (see below). It is important to note that the classification of *up* and *down* or *left* and *right* depends on knowing an accurate camera model. This can be a disadvantage if the camera model is subsequently refined. For example, if an edge is nearly horizontal, a small rotation of the coordinate axes could change the up-down sense of its endpoints and require a restructuring of the data.

Two steps are now taken to "clean up" the data. For a vertex that involves only two lines, the coordinates of the intersection are used to replace the coordinates of the endpoint of the edge(s) involved. This has the effect of lengthening edges that "almost" touch and shortening edges that cross "slightly". These judgments are determined by a distance threshold that is governed by the accuracy of the original edge finder. If a vertex involves the endpoints of more than two lines, there is a good chance that not all pairwise intersections will occur at the same point. In such cases, an average position is taken, and the endpoints are adjusted to agree with it.

Each line that serves as the top of a "T" junction will have its corresponding side (left or right) divided into two or more parts depending on the number of "T" junctions involved. These parts are stored as *side descriptors* in the data structure and are classified as either *left* or *right* depending on their relative positions. A *side descriptor record* comprises the following:

ESBRI	average brightness
ESBOT	edge whose "T" junction forms bottom of this side
ESPOS	position of bottom "T"
ESPTR	pointer to next side descriptor

After the side descriptors have all been found, they are sorted from top to bottom on each side of each edge. Each edge record has at least one left and one right side descriptor.

The side descriptor of an edge record corresponds to a surface which is adjacent to that edge. While much of the data structure is oriented toward representing the geometric relations on edge segments, side descriptors provide a place to store surface properties. Brightness is stored as a single value in each side descriptor. Thus it must represent less information that the original image, since there are fewer side descriptors than pixels. A region in the image corresponding to a surface will be represented by n side descriptors, where n is the number of edge segments in the boundary of the region.

To calculate these brightness values, we generate for each side descriptor an epipolar line along which brightness values are sampled and averaged. The line intersects the edge segment midway between the two vertices that define this side descriptor. Intensity values are sampled at 1/4 pixel intervals along this line either to the left or right (depending on which side descriptor) until another edge or the edge of the image is encountered. Each sample comprises a bilinear interpolation of the four pixel intensities nearest the sample point. The samples are averaged to produce a single value representing the brightness of the surface. This very simple measurement, repeated for each side descriptor, is the only form in which the original intensity information is retained for subsequent processing.

3.2 One Dimensional Processing

The stereo problem is divided into a series of one-dimensional problems along epipolar lines and the dynamic programming algorithm discussed in the last chapter is applied to each one independently. This section describes the data structures and procedures of the implementation.

3.2.1 – Slices

The dynamic programming match is applied to a selected set of *slices* through the images. A slice consists of an epipolar line pair together with information about each edge curve in the image that crosses the epipolar line. We define a slice to be two lists of *intersection records*, one for the left image, one for the right image. Each list is sorted on the value in **LOC**. An *intersection record* comprises the following:

EDG	pointer to edge record whose edge curve intersects this epipolar line
LOC	x-coordinate of the intersection
ANG	angle of the edge at the intersection
BRI	average brightness of the interval to the left of this edge
TOP	y-coordinate of top end point of this edge
BOT	y-coordinate of bottom end point of this edge
TV	type of vertex at top
BV	type of vertex at bottom

This record supplies all the information for computing the constraints used by the Viterbi matching.

The procedure for generating a slice is straightforward. Given the equation of an epipolar line in each image, search through all edge records and make a list of intersection

records, one for each edge that intersects the epipolar line. Sort the lists by the value in **LOC**, producing a left to right ordering. Calculate the average brightness, **BRI**, by inspecting the side descriptor records for each edge in the list. For each interval, there will be typically two side descriptors from which we simply take the average of their brightness values.

It is useful if each slice can be processed independently, making use of no information from adjacent slices. This property allows the computation over the whole image to be easily programmed for parallel computation. Thus, in our single processor implementation, the order of choosing slices does not matter. However, the particular set of slices chosen does matter.

One technique is to generate slices every 8 pixels for the first iteration over images of about 256 x 256 pixels. The second iteration uses another set of slices at an 8 pixel spacing, but phased to lie half way between those of the first set. This interlacing covers the image with a resolution of 4 pixels. A second method is to double the number of slices at each iteration until the final resolution is reached. For example, on a 256 pixel image, slice at 128, then at 64 and 192, then at 32, 96, 160, and 224, etc.

We have also experimented with data-dependent choices, usually for the final iteration. For example, the 4 pixel interlace provides no direct data for some edge curves with an extent of less than 4, but the number of missed edges is small. Thus, it is practical to choose a final set of slices that pass through the centers of each missed edge.

For each slice chosen, we apply the modified Viterbi algorithm to match the list of left intersection records with the list of right intersection records. (The implementation is actually organized around the intervals between intersections, rather than the intersections themselves.) The dynamic programming array is initialized and the best path calculated using the evaluation function described below. Then a threshold is set and all paths whose scores are within that threshold of the best path are identified.

At this point, there is more information available about each of the paths than is needed in the next step. The data structures are simplified by preserving only a limited amount of data for each slice: a list for each edge (intersection) of all the ***match interpretations*** given it by any of the collection of suboptimal paths.

An ***edge match interpretation*** consists of a match type and a pointer to an edge in the other image. The match type may be ***visible to both***, in which case the pointer is to a corresponding edge in the other image, or ***occluded***, in which case the pointer is to an edge of the occluding surface in the other image. This list is considered as a list of possible interpretations, where an edge interpretation is possible if and only if it occurs within a high scoring path. No attempt is made to assign weights to the edge match interpretations based on path scores; all paths selected by the threshold are considered equally likely.

It should also be noted that while a context spanning the image was used in selecting each edge match interpretation, this context is not passed back with the match. Although this represents a loss of some information, it serves to make the output of the Viterbi algorithm more manageable.

3.2.2 – Evaluation Function

We have earlier described the modified Viterbi algorithm for determining the optimal and sub-optimal paths. This section describes the evaluation function used in that algorithm. The function consists of four terms, each with an ***ad hoc*** weight, combined linearly:

$$\sum_{i \in path} (K_1 I_i + K_2 E_i + K_3 B_i + K_4 X_i) \qquad (3-5)$$

where:

K_1, K_2, K_3 and K_4 are the *ad hoc* weights,

and for each *path element, i,*:

I_i is the composite *interval* measure,

E_i is the composite *edge* measure,

B_i is the *surface breaks* penalty, and

X_i is the *excess length* penalty.

This sum gives the score for each of the paths which satisfies the geometric occlusion constraints outlined earlier. The paths for which the sum is a minimum is the optimal or "best" path referred to above.

The summing of interval and edge measures is a simplification. These measures are not in fact independent, since they are related to one another by the geometry of the scene. However, we do not yet know the proper function for combining them. Experimentally, adding them with a 60:40 weight favoring intervals has worked best. The last two measures, *surface breaks* and *excess length*, are *ad hoc* measures that are given low weights. Their primary purpose is to distinguish paths where there are many occlusions, and hence little information from the *interval* and *edge* measures.

Interval Measure

The *interval* measure consists of two components, *brightness* and *interval length ratio*. These measures apply to a common object, the surface represented by the matched intervals, so these two components are treated as probabilities and are multiplied to produce a composite measure:

$$I_i = \mathrm{BRI}_i \, \mathrm{RATIO}_i \qquad (3-6).$$

In the case where the interval is occluded (visible in one image only) the value for I_i is set to zero.

The ***brightness*** measure approximates the probability that the brightnesses from the left and right images represent two measurements of the same physical property with gaussian noise added. Note that reflectivity of a surface is not, in general, independent of angle; thus the two cameras will not in fact measure the same physical property. However, the effect of this simplification should be small in most scenes. Thus we use the following:

$$\mathrm{BRI}_i = \exp\left(-\left(\frac{\mathrm{BL}_i - \mathrm{BR}_i}{2\mathrm{NS}}\right)^2\right) \qquad (3-7)$$

where:

BL_i and BR_i are the average brightnesses of the left and right intervals, respectively, and

NS is an estimate of noise in the brightness value.

Equal left and right brightness values always produce $\mathrm{BRI}_i = 1$, while values that differ by NS produce $\mathrm{BRI}_i = \exp(-1/4)$ and so on.

The ***interval length ratio*** measure has been described earlier, in the section on constraints. It is normalized to lie between 0 and 1, by dividing by MAXRATIO, the maximum value of the ratio function. Thus the ratio of the interval lengths from the two images is mapped to a value between zero and one which we treat as a likelihood of match.

Edge Measure

The ***edge*** measure consists of three components, ***edge angle***, ***edge extent*** and ***endpoint position***. As in the interval measure, these components represent measurements of a common object, i.e., the edge crossing the slice. Thus the components are normalized to lie between zero and one and are multiplied to give the ***edge*** measure:

$$E'_i = \mathrm{EANG}_i\, \mathrm{ELEN}_i\, \mathrm{EPOS}_i \qquad (3-8).$$

In the case where the edge is occluded (visible in one image only) the value for E'_i is set to zero. This measure will later be adjusted according to ***previous information*** (see below).

The ***edge angle*** measure is based on the calculation described earlier. The saddle-shaped probability density surface takes on values between zero and positive infinity. We normalize it to have an average value of one (volume under the surface $= \pi^2$) and then apply the following function to map its values to a range of zero to one:

$$f(x) = 1 - \exp(-x).$$

Thus the two angles at which the edge curves cross the slices in the left and right images are mapped to a single value between zero and one which represents the likelihood that the edge curves match. This ***edge angle*** measure is the strongest component contributing to the overall edge measure.

The use of ***edge extent*** as a constraint was discussed earlier. In this function, the difference in extent lengths is normalized by the average of the two extents, and then treated as two measurements subject to gaussian noise. Expected differences are due to differences in y-axis scale factors between the left and right images. The normalization removes dependency on absolute differences, and this is also the reason for using the

derivative of camera model position error. Errors from image misregistration are to be handled by the ***endpoint position*** measure.

The ***extent*** of each edge segment is calculated from the following:

$$\mathrm{LEXT} = \mathrm{LTOP}_i - \mathrm{LBOT}_i$$

$$\mathrm{REXT} = \mathrm{RTOP}_i - \mathrm{RBOT}_i$$

From the left and right extents, we calculate:

$$\mathrm{ELEN}_i = w \exp\left(-\left(\frac{\mathrm{LEXT} - \mathrm{REXT}}{(\mathrm{LEXT} + \mathrm{REXT})\mathrm{DYNS}}\right)^2\right) + 1 - w \qquad (3-9)$$

where:

w is a weighting constant, currently 0.5, and

DYNS is an estimate of the derivative of camera model and segmentation accuracy, in pixels/pixel.

The weighting factor allows the contribution of this measure to be adjusted in overall ***edge*** measure. A weight of 1 results in values ranging from zero to one, a weight of 0.5 results in values between 0.5 and 1.0 and so on. Equal left and right extents always produce a value of 1.

In calculating this measure, a very simple form of monocular shape cue is used. If the shorter of the two extents has a "T" junction at either end, we assume scene geometry has occluded part of that edge curve and return a value of 1 for this measure.

The edge ***endpoint position*** measure is similar to the ***extent*** measure, but is performed separately for top and bottom, and depends on absolute y-coordinate positioning rather than y-scale factor. The measure is calculated as follows:

$$\mathrm{TVAL}_i = exp\left(-\left(\frac{\mathrm{LTOP}_i - \mathrm{RTOP}_i}{\mathrm{YNS}}\right)^2\right)$$

$$\mathrm{BVAL}_i = exp\left(-\left(\frac{\mathrm{LBOT}_i - \mathrm{RBOT}_i}{\mathrm{YNS}}\right)^2\right)$$

$$w' = 1 - \sqrt{1 - w}$$

$$\mathrm{EPOS}_i = (w'\mathrm{TVAL}_i + 1 - w')(w'\mathrm{BVAL}_i + 1 - w') \qquad (3-10)$$

where:

w is a weighting constant, currently 0.5, and

YNS is an estimate of noise in y-position, in pixels.

In this function, the difference in y-position is compared directly to an expected error in y-position, YNS, assumed to be gaussian distributed. The values for top and bottom are multiplied and weighted to give a final value. A weight of 1 results in values ranging from 0 to 1 and a weight of 0.5 results in values ranging from 0.5 to 1. Equal top and bottom positions will produce the maximum value of 1.

As in extents, the end points are examined for "T" junctions. In this case, however, the slope of the crossing edge is also considered. Four cases are considered explainable by occlusion, and receive a maximum value (see Figure 3-7).

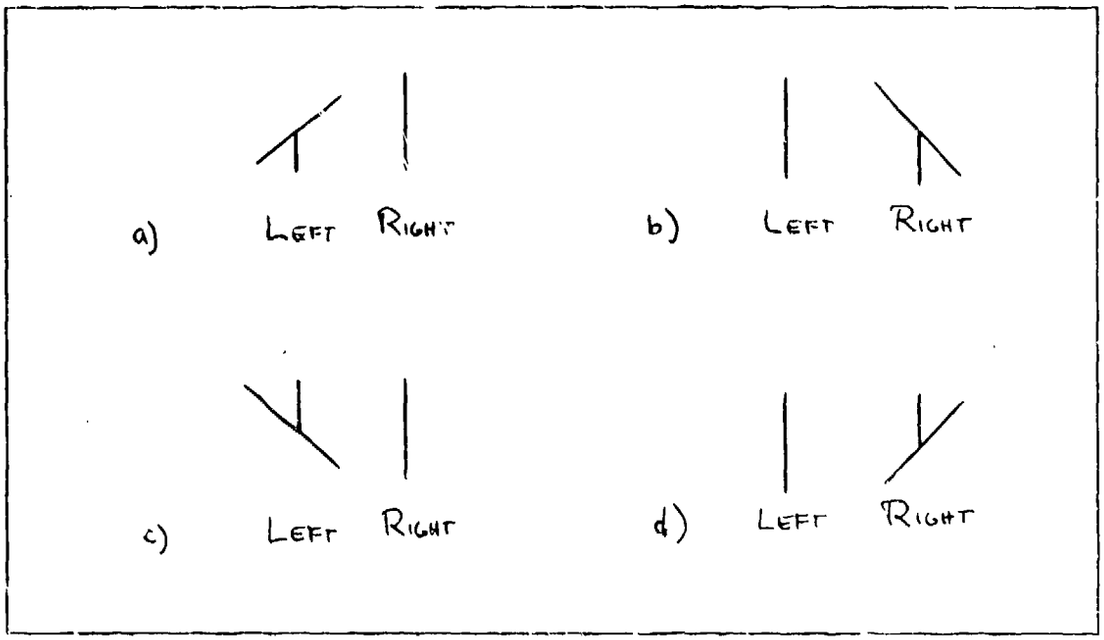


Figure 3-7. *These are four types of T-junctions that may shorten an edge in one view.*

Previous Information

Finally, there is a mechanism for incorporating external knowledge of an edge match into the measure. This is used to feed information from previous iterations into the current evaluation, as will be described in a section below. The function used is a simple one:

$$E_i = \textbf{if } \mathrm{PRE}_i > 0.5 \textbf{ then } 1 - 2(1 - \mathrm{PRE}_i)(1 - E'_i) \textbf{ else } 2\,\mathrm{PRE}_i E'_i \qquad (3-11)$$

where:

PRE_i is the bias of previous information between 0 and 1, and

E'_i is the current measure between 0 and 1.

If there is no previous information about this particular edge match, PRE_i is set to 0.5 and this function is the identity. If previous information is strong, PRE_i is > 0.5 and the result is scaled to lie between $(2PRE_i - 1)$ and 1. If $PRE_i < 0.5$ then the result lies between 0 and $2PRE_i$. For $PRE_i = 0$ or 1, the result is constrained to be 0 or 1 respectively.

Thus the measure for edges is taken as a combination of edge angle, extent and endpoint position, modified by previous information. The result is a value between 0 and 1 which we treat as a likelihood of match.

Surface Breaks

The *surface breaks* measure is an *ad hoc* measure of surface continuity and smoothness. Its value is 1 for an edge which represents a discontinuity and 0 for a smooth continuous surface where the edge is a surface mark. The surface breaks measure is given a small negative weight (-0.05) in the evaluation function, and serves to bias the results toward smooth surfaces when there is no other strong information.

The measure is determined as follows (see Figure 2-10 for terminology):

- 0 for edges which are interpreted to be out of the field of view of one camera, since nothing can be deduced about such edges.
- 0 for edges on a left or right face, since these could lie on a smooth surface.
- 0.5 for edges on left or right tops or bases since these represent a transition from a visible surface to an occluded one or vice versa. There must be at least a slope discontinuity at these points.
- 0.5 for peaks and valleys, since these must be slope discontinuities.
- 1 for left or right cliffs, which can be caused either by discontinuities or two or more slope changes involving unseen edges.

If there is no previous information about this particular edge match, PRE_i is set to 0.5 and this function is the identity. If previous information is strong, PRE_i is > 0.5 and the result is scaled to lie between $(2PRE_i - 1)$ and 1. If $PRE_i < 0.5$ then the result lies between 0 and $2PRE_i$. For $PRE_i = 0$ or 1, the result is constrained to be 0 or 1 respectively.

Thus the measure for edges is taken as a combination of edge angle, extent and endpoint position, modified by previous information. The result is a value between 0 and 1 which we treat as a likelihood of match.

Surface Breaks

The *surface breaks* measure is an *ad hoc* measure of surface continuity and smoothness. Its value is 1 for an edge which represents a discontinuity and 0 for a smooth continuous surface where the edge is a surface mark. The surface breaks measure is given a small negative weight (-0.05) in the evaluation function, and serves to bias the results toward smooth surfaces when there is no other strong information.

The measure is determined as follows (see Figure 2-10 for terminology):

- 0 for edges which are interpreted to be out of the field of view of one camera, since nothing can be deduced about such edges.
- 0 for edges on a left or right face, since these could lie on a smooth surface.
- 0.5 for edges on left or right tops or bases since these represent a transition from a visible surface to an occluded one or vice versa. There must be at least a slope discontinuity at these points.
- 0.5 for peaks and valleys, since these must be slope discontinuities.
- 1 for left or right cliffs, which can be caused either by discontinuities or two or more slope changes involving unseen edges.

- 0 or 0.5 for flat edges. For each of these intervals to left and right both represent visible surfaces. From the detailed profile information, the slopes of these surfaces can be calculated, either exactly or to an inequality. For the exact calculations, slopes whose ratio is between 0.9 and 1.1 are considered part of a smooth surface, and score 0. For the inequalities, if slope values can be chosen which satisfy the inequalities and produce a ratio between 0.9 and 1.1, a score of 0 is returned. For all other cases, a slope discontinuity is indicated and the value returned is 0.5.

Excess Length

Excess length is a measure of profile irregularity. It is calculated by taking the length of a profile segment, measured in the epipolar plane of the slice, and subtracting the minimum segment length that could have produced the same total projected intervals in the left and right views. This minimum length segment is generally one whose normal intersects the baseline of the two cameras. Thus, a flat surface of this orientation would have the minimum profile length, while still filling the field of view. (Refer to Figure 2-22.) On the other hand, an irregular surface in which every part was visible to only one camera would have the maximum value according to this measure. By giving this measure a small negative weight (-0.1) in the evaluation function, we bias the results toward nearly planar profiles in the absence of other strong information.

3.2.3 – Results

The results of one-dimensional processing are a set of profiles for each epipolar slice; each set includes the optimum and those suboptimal profiles that met the score threshold ϵ. The paths produced by the Viterbi algorithm together with the camera model parameters allow profiles to be reconstructed in the original scene geometry. We

present several such profiles here, using the notation developed in a previous section, to illustrate the results of one-dimensional processing.

The primary advantage of the extended algorithm is that it produces alternative interpretations in addition to the local optimum. It is possible for one of these suboptimal profiles to be preferred when a wider context is examined.

In Figure 3-8 we have selected a slice from an image of an L-1011 aircraft. The slice passes near a corner in the wing, and the optimum profile found by the Viterbi algorithm misinterpreted the profile at that point. In Figure 3-9a the notation (2,2,2) indicates that the surface corresponding to that interval has been interpreted as visible to both cameras. This is an erroneous match of the wing shadow in the left view with the aft portion of the wing in the right view. Interval (2,3,2) indicates that the aft portion of the wing in the left view is interpreted as occluded. The forward portion of the wing in both views is correctly matched, as are all other intervals in the profile.

The evaluation function for the profile in Figure 3-9a equalled 4.4283. The profile in Figure 3-9b scored 4.1964, but correctly interpreted the wing shadow, (1,2,3), as occluded and the aft portion of the wing, (2,3,1) as matched. These match interpretations, while locally suboptimal, were ultimately selected by the two-dimensional processing.

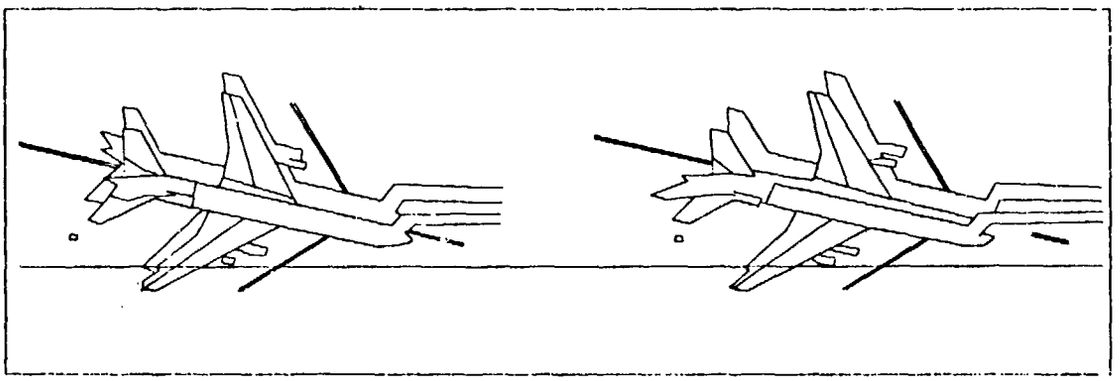


Figure 3-8: *A slice taken through the aircraft scene at row --37 is used to demonstrate a "correct" profile that is locally suboptimal.*

In addition to the coarse distinction between matched and occluded surfaces, the Viterbi algorithm also distinguishes surface discontinuities from slope changes or surface markings. Thus, we see that Figure 3-9b has incorrectly interpreted the ground surface, (4,5,1), as being continuous with the wing surface, (3,4,1). The profile illustrated in Figure 3-9c correctly shows the discontinuity. This last profile scored 4.0964, and was 72nd in the list of suboptimal profiles. The main reason for this large number of paths is that surface discontinuities and surface markings are only weakly distinguished in the evaluation function and the number of combinations over seven surfaces is large.

Most errors in computing profiles are caused by imperfect data. Naturally, a system based on real images cannot expect to have perfect data, so it is important that the effects of extra or missing edges on the Viterbi algorithm are localized. Figure 3-10

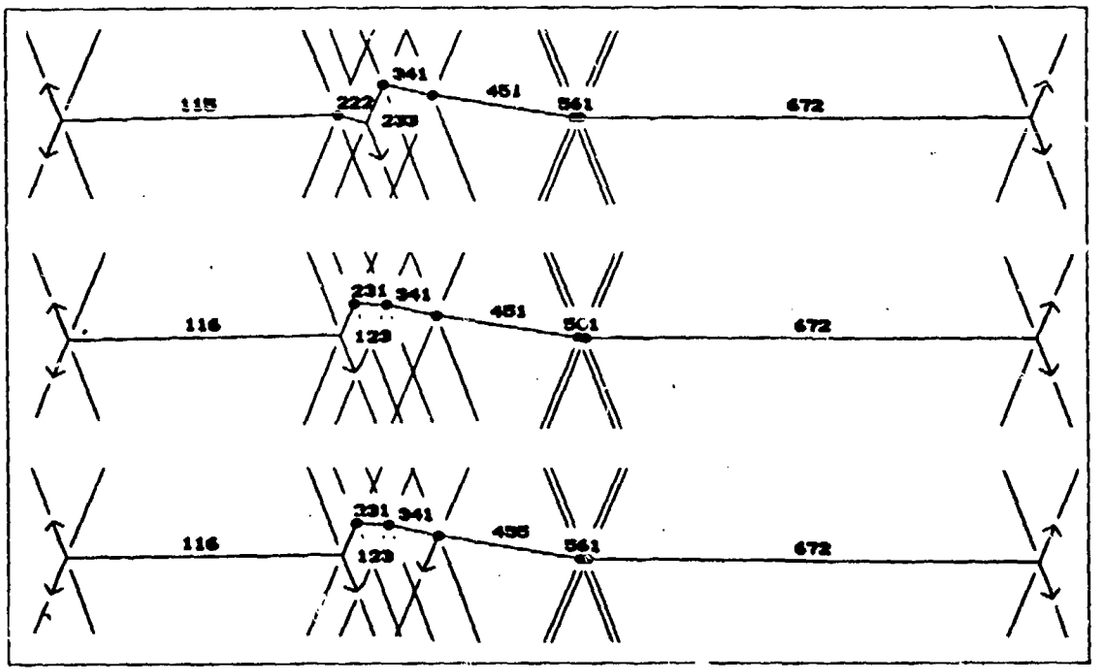


Figure 3-9: *Three selected profiles computed from the slice in Figure 3-8 are shown in a, b and c. These paths scored 1st, 34th and 72nd respectively, in the list of paths within 0.35 of the optimum score. The "correct" interpretation is the last.*

shows the same aircraft images with slices selected at rows −16 and −17, one pixel apart. In row −16, the slice misses the small box near the tail in the right image and also misses one boundary of the taxiway mark near the nose. The same slice misses an edge of the boarding ramp near the nose in the left image.

Figure 3-11a shows the locally optimal profile generated from the slice at row −16. The most obvious problem is caused by the box, which has been depressed below the ground level to make it occluded by the wing. This severe distortion is due to the fact that no other intervals were nearby to serve as an occluding surface. Near the nose, missing edges from the right and left cancelled; there was no occlusion introduced, only an incorrect and distorted match. In each case, the effects of the missing edges did not extend beyond the adjacent intervals. From the wing shadow, (1,3,3), to the fuselage, (5,8,1), all matches are correctly interpreted.

Only one pixel away at row −17, the optimum profile correctly interprets all surfaces (see Figure 3-11b). There are no missing edges and the occlusions are all legitimate. Errors isolated to a single slice is a common effect of misregistration of images and illustrates the value of using adjacent slices and continuity constraints to overcome local errors.

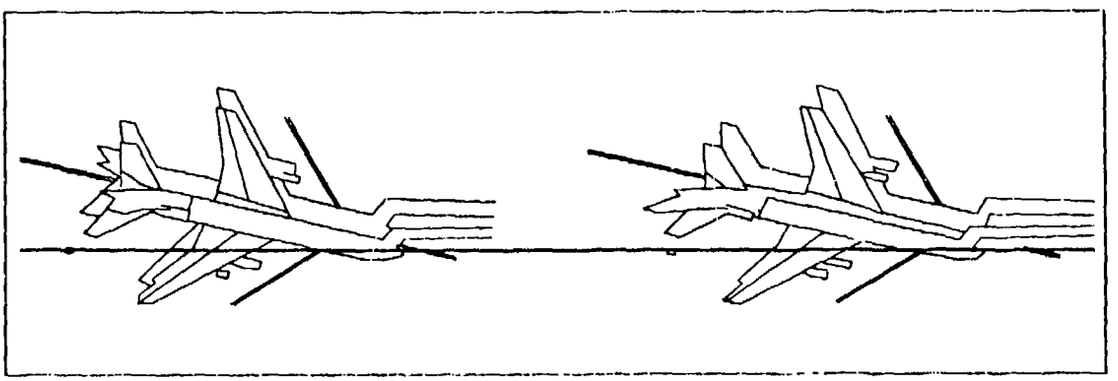


Figure 3-10: *Two slices taken through the aircraft scene at rows −16 and −17 are used to demonstrate the effect of missing edges.*

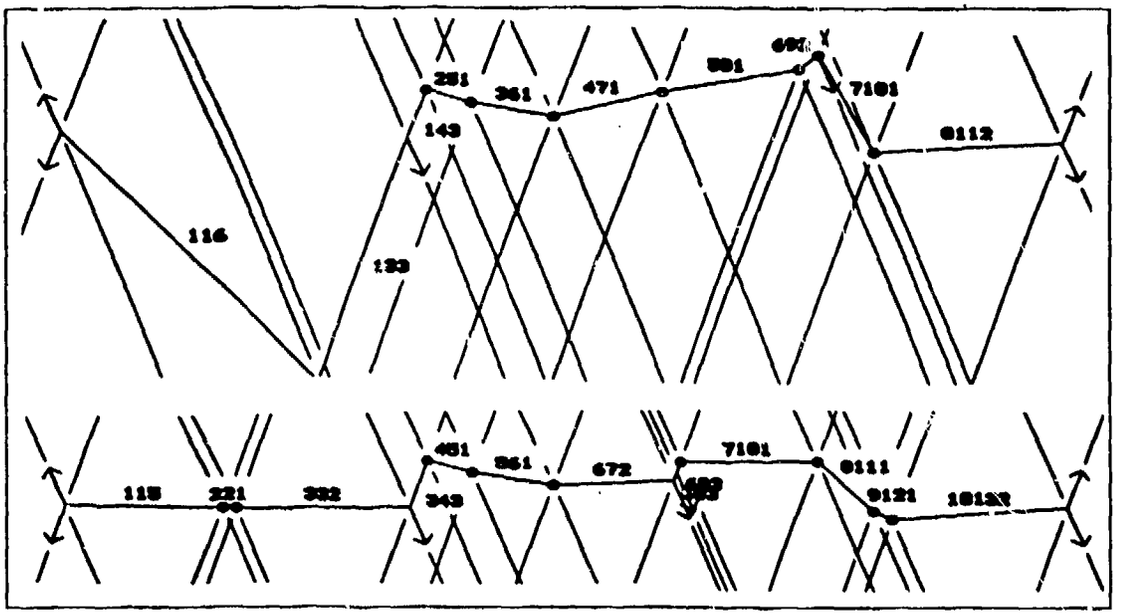


Figure 3-11: *Profiles computed from the slices −16 and −17 in Figure 3-10 are shown in a and b, respectively. In a, the profile is distorted to explain missing edges, but the effect is limited to intervals adjacent to the missing edges. In b, there are no missing edges, and the profile is essentially correct.*

3.3 Two Dimensional Processing

The processing in two dimensions consists primarily of computing a value to be assigned to each *edge match interpretation* and feeding this value back into subsequent iterations of the Viterbi algorithm. This computation involves the final constraints, *continuity* and *consistency*.

3.3.1 – Match Lists

The results of the Viterbi algorithm acting on each slice must be incorporated into the data structure in such a way that consistency between slices can be computed. Four additional fields are added to the *edge record* that was described above. They are:

EMAT pointer to edge match list

ESAMP	number of samples supporting edge match list
M2MAT	pointer to secondary edge match list
M2SAMP	number of samples supporting M2MAT

An edge match list is a list of *edge match* records, each of which comprises the following fields:

EMCLS	pointer to match class list
EMCNF	confidence measure for this match class
EMNUM	number of supporting samples
EMPTR	pointer to next *edge match* record

A match class list is a list of *match class* records, each of which comprises the following fields:

ECLRB	match type (visible to left, right or both)
ECEDG	pointer to match or occluding edge
ECPTR	pointer to next *match class* record

The secondary structure (**M2MAT**) uses two similar record types with fields named **M2CLS, M2CNF, M2NUM, M2PTR** and **C2LRB, C2EDG, C2PTR**. Figure 3-12 shows the relationship of these structures to the *edge record*.

The Viterbi algorithm processes a slice consisting of left and right parts. For each edge in the left, **EMSAMP** is incremented and the list of edge match records is searched for each interpretation found in the Viterbi algorithm. For exact matches (down to the *match class* record), **EMNUM** is incremented in the corresponding *edge match* record. If the interpretation is *equivalent* (see definition below) to an interpretation already present, then a new *match class* record is added to the existing match class list and **EMNUM** is incremented. If the interpretation is a totally new one, it is added to the data structure as a new *edge match* record with a single *match class* record, and **EMNUM** is set to 1.

The same procedure is then followed for edges in the right part of the slice. Figure 3-13 illustrates this data structure for a sample image.

A refinement to this procedure is to ignore the interpretations for slices that are too close to an edge's endpoint. "Too close" is defined here as within YNS/2 pixels in the y-direction, where YNS is the estimated uncertainty in the camera model (see Equation 3-10). This improves performance by preventing misregistration errors from propagating.

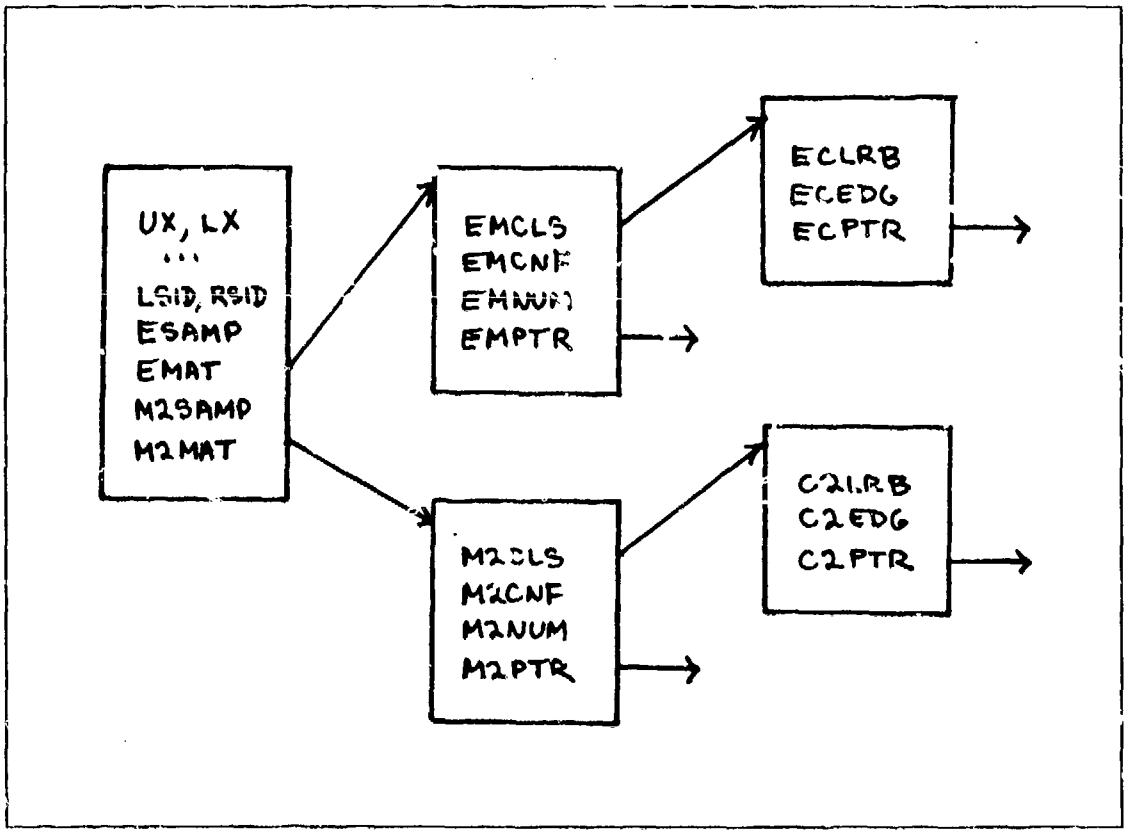


Figure 3-12: *The data structure used in the two dimensional processing includes a list of potential matches for each edge. Each potential match may comprise a list of consistent match interpretations. This primary match list is rebuilt on each iteration and is used to update the secondary structure, which holds the final results.*

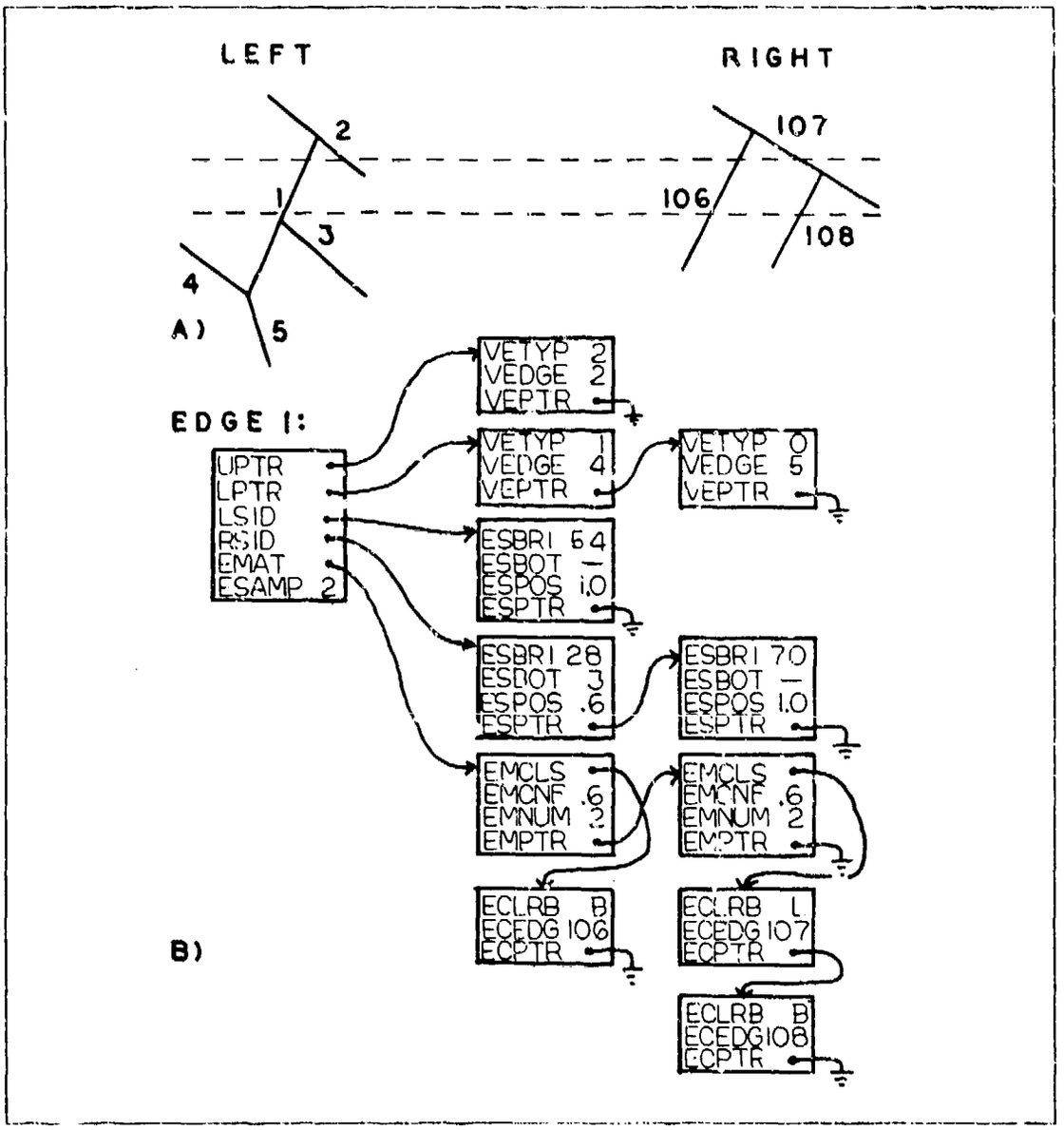


Figure 3-13: *A sample left and right image pair is shown in a with the corresponding stereo analysis illustrated in the data structure in b. The primary match structure is shown based on two samples, and indicates that edge 1 has two potential match classes: a match with edge 106 or a match with edge 108. Note that edge 1 may be occluded by edge 107 on one of the epipolar lines but that this match interpretation is consistent with the match to edge 108, so they appear in the same match class. In this example, the stereo interpretation for edge 1 remains ambiguous.*

In this procedure a simple notion of edge match *equivalence* is used. The intent is to make the matching procedure independent of edge segmentation. Therefore, two edge matches are considered *equivalent* if:

- the matches are of the same type (i.e., both visible to both cameras, or both occluded in the same view), and
- the edges pointed to are *connected* in the sense that the top endpoint of one meets the bottom endpoint of the other at a vertex, and no other edges meet at that vertex.

The result of the procedure just described is to ensure that every *edge match interpretation*, whether a match in both views or an occlusion, is incorporated into the data structure. The frequency of occurrence of each interpretation is also recorded (i.e., how many samples or slices support a given interpretation). *Equivalence classes* are formed for matches which are not identical in our representation, but which are potentially identical in the scene.

3.3.2 – Consistency

The results from applying the Viterbi algorithm independently to each slice are recorded in the data structure as described above. The matches listed are those which were calculated to be possible based only on the information in the particular slice taken. The next step in the program is to filter these potential matches by looking for *consistency* across slices.

It is quite possible for a given match interpretation to be in error because of noise or imperfect data on a particular slice. Such interpretations will generally not be supported by adjacent slices. Interpretations which are consistent with a context including multiple slices are copied into a parallel data structure in preparation for subsequent iteration.

For each edge, new potential matches are ***hypothesized*** from adjacent connected edges. The intent of this procedure is shown in Figure 3-14. If edge A is connected at top or bottom to a second edge B, and the second edge continues in the same direction (up or down respectively) then the edge match list for this neighboring edge is examined. The connectivity of each edge in this match list (e.g., edge C in the figure) is examined for connectivity similar to that of the second edge, B. Edges that have a position analogous to the original edge (e.g., D in the figure) are hypothesized as matches, and are added to the primary match list without incrementing **EMSAMP**. This procedure allows match information to be propagated along segmented scene edges from one segment to the next, without the overhead of selecting additional slices and executing the Viterbi algorithm. The continuity of edges across epipolar lines is a sufficiently strong constraint to justify this.

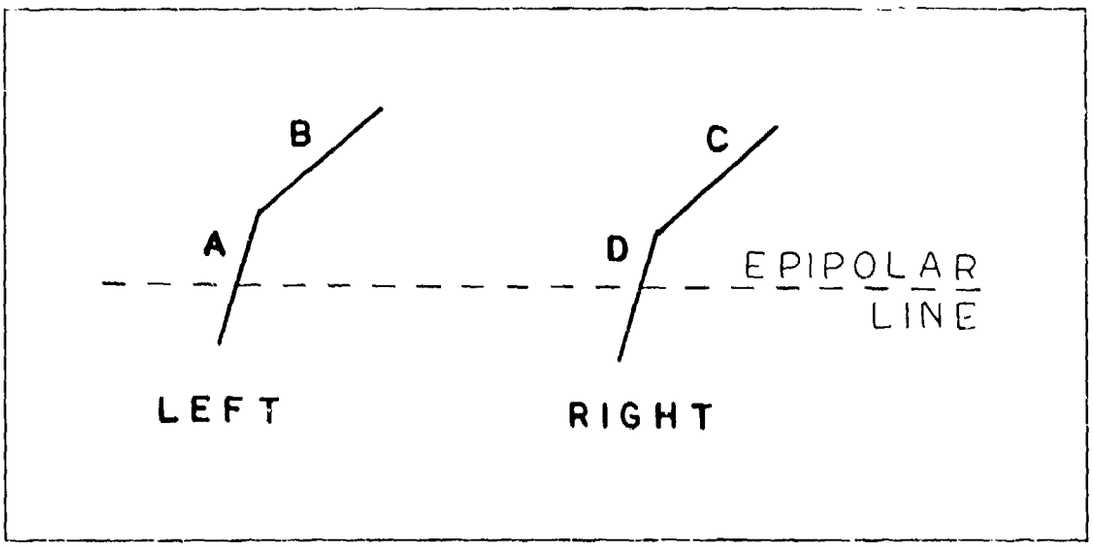


Figure 3-14: *Some matches may be hypothesized without actually running the stereo Viterbi algorithm. Similar connectivity allow us to assume a match between A and D based on a known match between B and C. Such a match is added to the data structure with zero evidence so it will not initially affect other matches.*

The remainder of the consistency checking procedure builds the secondary data structure. While the primary structure, (**EMAT**), is cumulative, this secondary match list, (**M2MAT**), is initialized on each iteration by copying for each edge the values of the primary match list. After this initialization, the continuity constraint is used to extend the secondary match lists as described below.

Information on potential matches is propagated to adjacent edges based on *continuity*. Here we define two edges as *connected* if they meet in a vertex that includes no other edges, and if one edge extends upward from the vertex and the other downward. Two edges that are joined by a sequence of *connected* edges are *continuous*. For each edge, a search is made along such *connected* edges for matches which are *equivalent* to any match in the current match list. Any such *equivalent* matches are tallied in the **M2NUM** field of the secondary match list.

Thus the evidence in the secondary match list includes information from two sources:

- Evidence derived from slices passing through the edge.
- Evidence gathered from *connected* edges.

This evidence is evaluated based on the number of samples that support a given match interpretation, and the total number of samples contributing to any interpretation for the edge. For the primary match list, the total number of samples is just the number of slices intersecting the edge. The secondary match list adds to this the number of samples from connected edges. Note that an edge match that was *hypothesized* in the previous step may now accumulate evidence from connected edges. (This constitutes a very crude use of inference rules of the type discussed by Binford [Binford 1981].)

The function used to produce a confidence measure for each edge match interpretation is:

$$CNF = 0.5 + \frac{NUM - \frac{SAMP}{2}}{SAMP + 2} \qquad (3-12)$$

where:

CNF is the confidence measure,

NUM is the number of supporting samples, and

SAMP is the total number of samples.

This function has the property of yielding a value of 0.5 if no samples are available. If every sample supports the match interpretation (NUM = SAMP), the function approaches 1.0 as the number of samples increases $(1/2, 2/3, 3/4, \ldots)$. If none of the samples supports the interpretation (NUM = 0), the value of the function approaches zero as the number of samples increases $(1/2, 1/3, 1/4, \ldots)$. This function is designed to yield a number which can feed directly into the Viterbi evaluation function as "previous information" (see ***edge measure*** and Equation 3-11.)

3.3.3 – Results

This section reports some of the results of applying the stereo system to test data. A system has been written in the SAIL language and has been run on a Digital Equipment Corporation PDP-10 computer with a model KL-10 processor. The examples in this section were computed by doubling the number of slices per iteration until slices had been obtained at uniform 4 pixel intervals. This required six iterations, beginning with one slice through a 256 by 256 pixel image, and ending with 32 slices. For the jet aircraft scene, total computation was 287 seconds, comprising about 4.5 seconds per slice for the one-dimensional processing, and 1 second per iteration for the two-dimensional consistency checking.

It is difficult to show the total output of the stereo system; some edges have been matched in stereo, some have been classified as occluded and some have not been successfully classified. Perhaps the simplest and most direct way to display results is to

display just those edges which have been given an unambiguous stereo correspondence. Therefore, before an edge can be displayed, it must satisfy the following conditions:

- There is a unique *match class* that had the highest number of supporting samples; i.e., it was consistent with the most slices intersecting it.
- That *match class* did not interpret the edge as totally occluded; i.e., some part of it was visible to both cameras.
- The edge and its matching edge are each longer than four pixels and overlap, when projected normal to the y-axis, at least 50%.
- The edge and its match have angles that are greater than 0.2 radian from the stereo axis.

Thus, ambigous matches and known occlusions are not graphed. The position of an occluded edge is bounded but not known exactly. The diagrams show only edges whose position in 3-space has been completely determined by the system.

The edges are mapped to 3-dimensions and are scaled to fit in a convenient volume of space. This results in a cluster of edges which are then reprojected onto the image planes of two cameras that can be positioned interactively. The resulting images show the edge curves from viewpoints other than those of the original cameras. The stereograms may also be viewed in stereo by the practiced reader.

Stereo Results

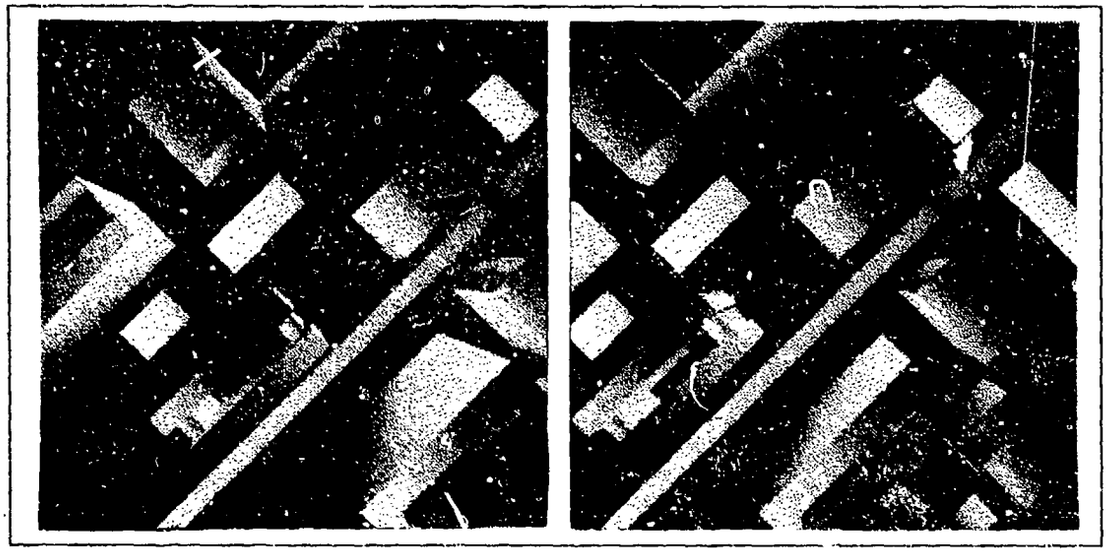


Figure 3-15: *Images from CDC show an artificial city scene.*

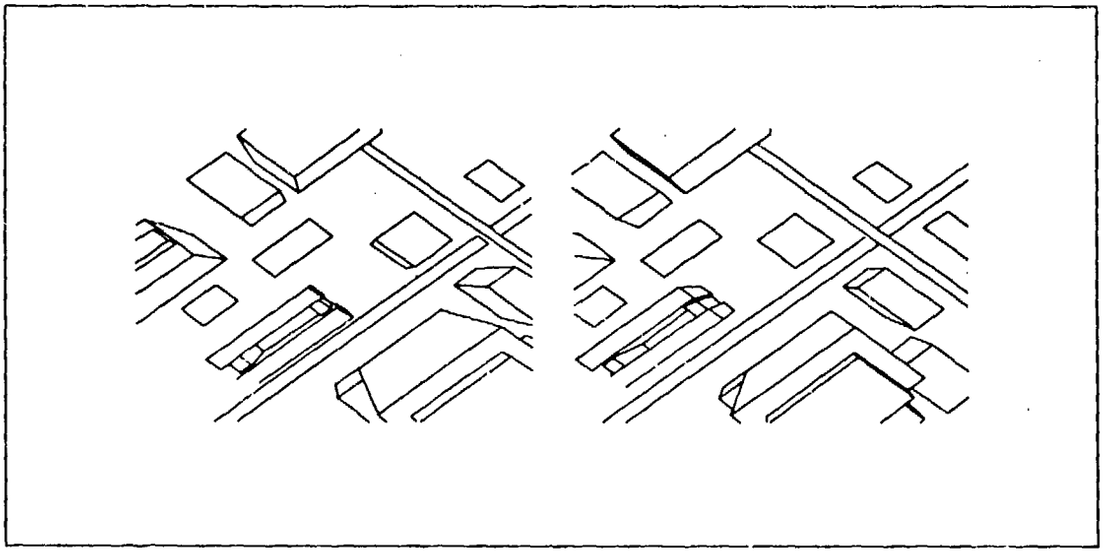


Figure 3-16: *Edges are produced and the images are registered for stereo processing.*

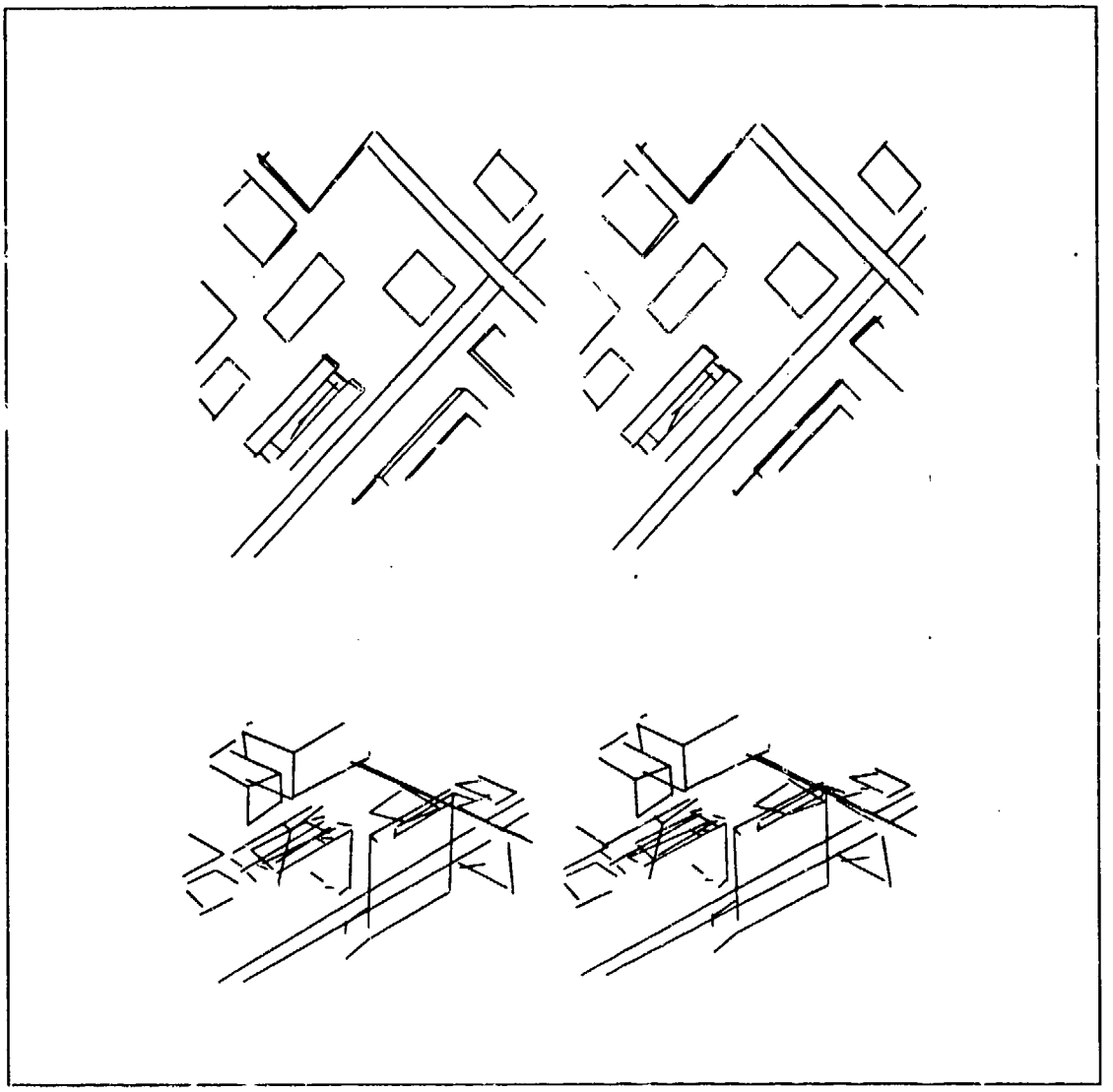


Figure 3-17: *These stereograms show overhead (90 degree) and 30 degree views of edges whose 3-dimensional positions have been determined.*

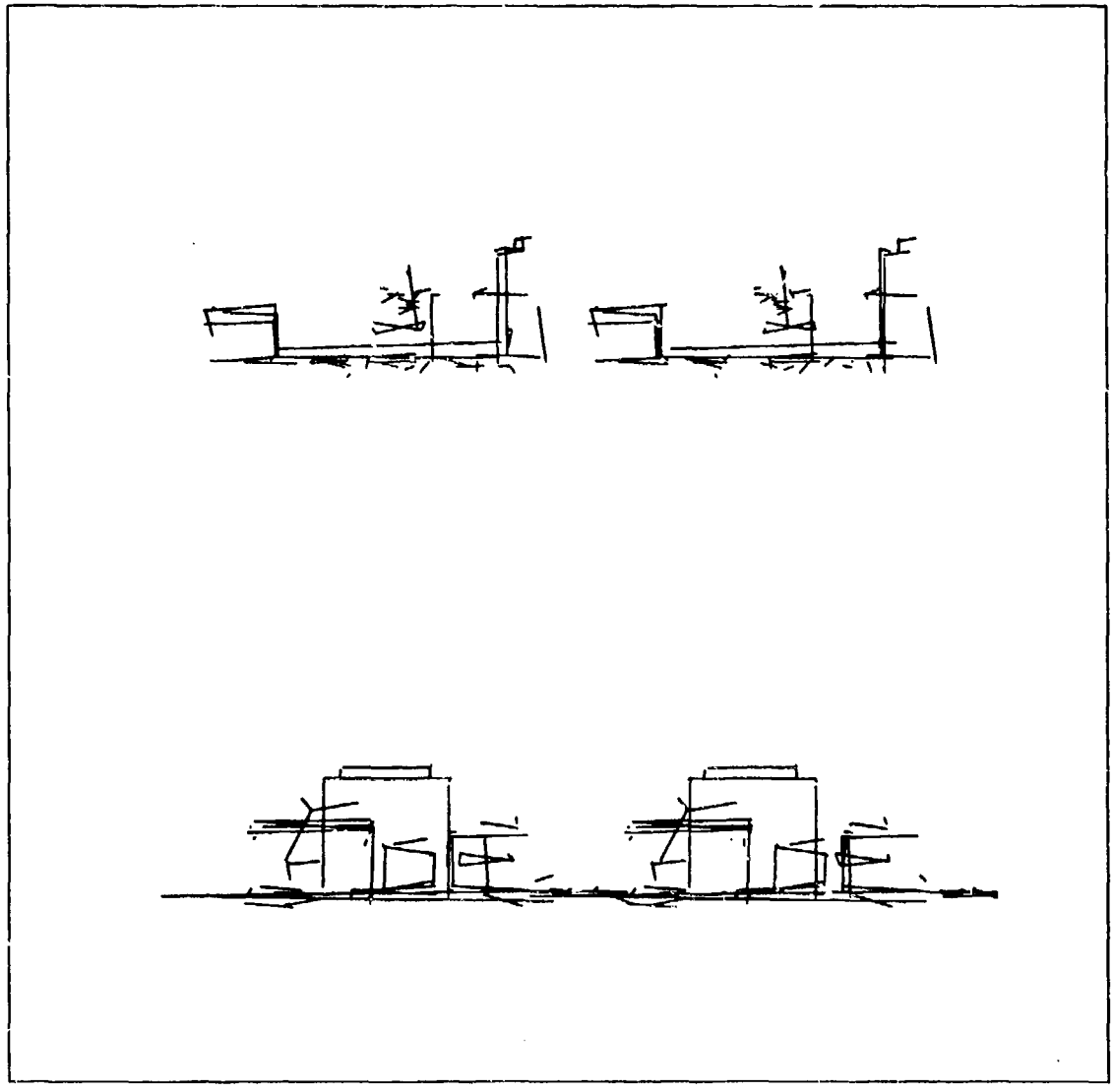


Figure 3-18: *These stereograms show the same edges as Figure 3-17, but from ground level (0 degrees).*

Figure 3-19: *Images from San Francisco Airport show an L-1011 at a boarding ramp.*

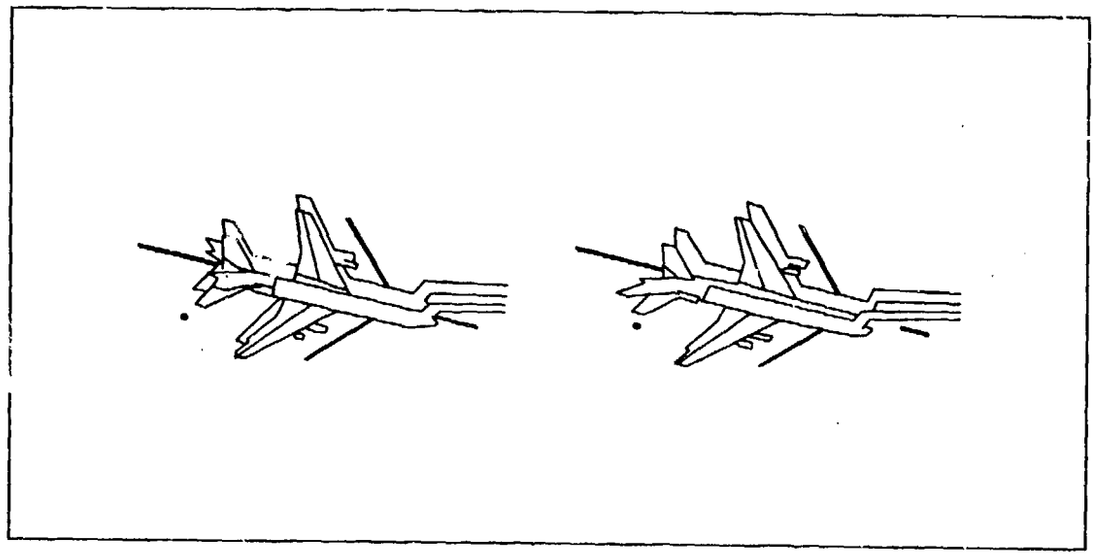


Figure 3-20: *Edges are produced and the images are registered for stereo processing.*

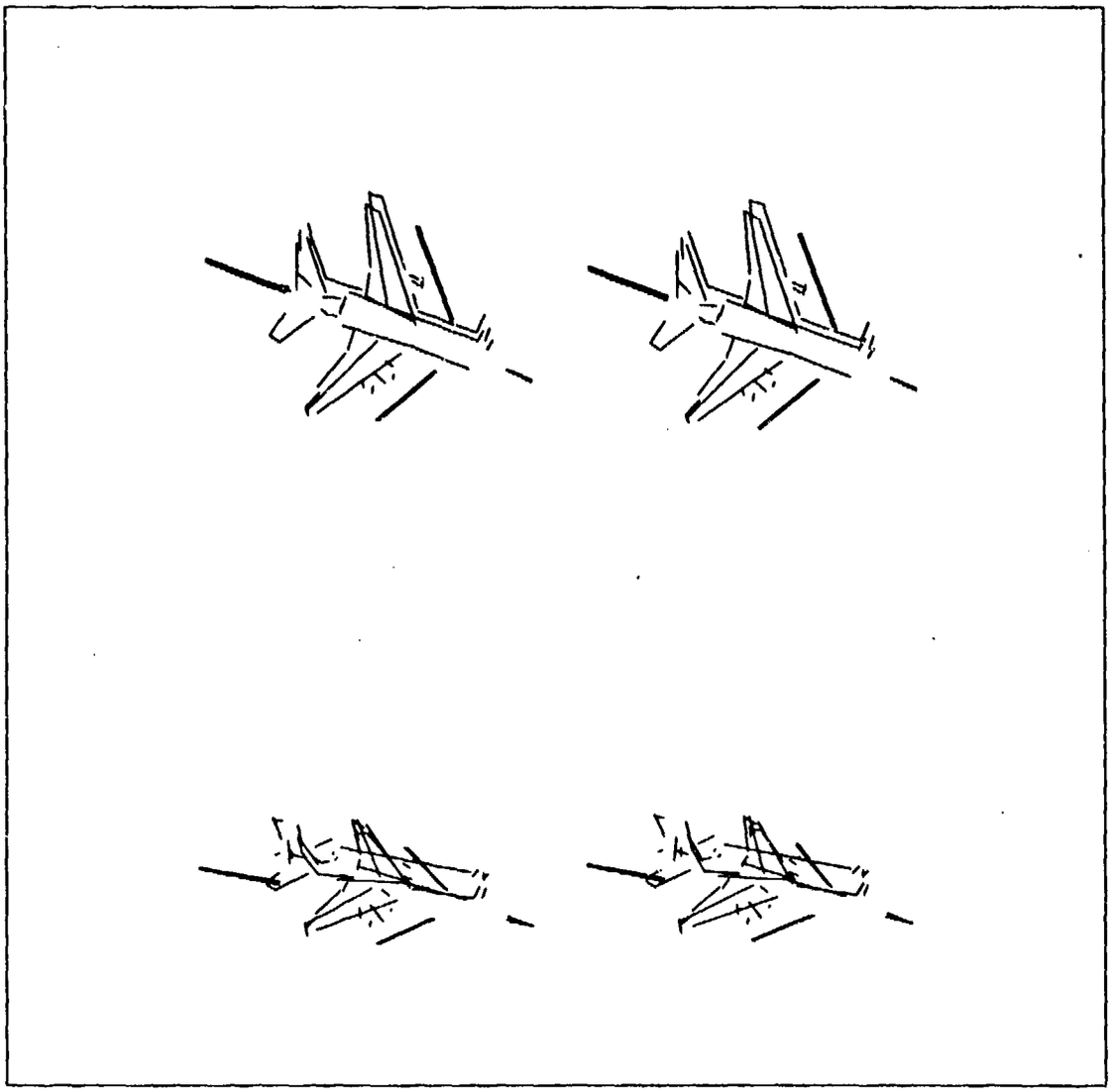


Figure 3-21: *These stereograms show overhead (90 degree) and 30 degree views of edges whose 3-dimensional positions have been determined.*

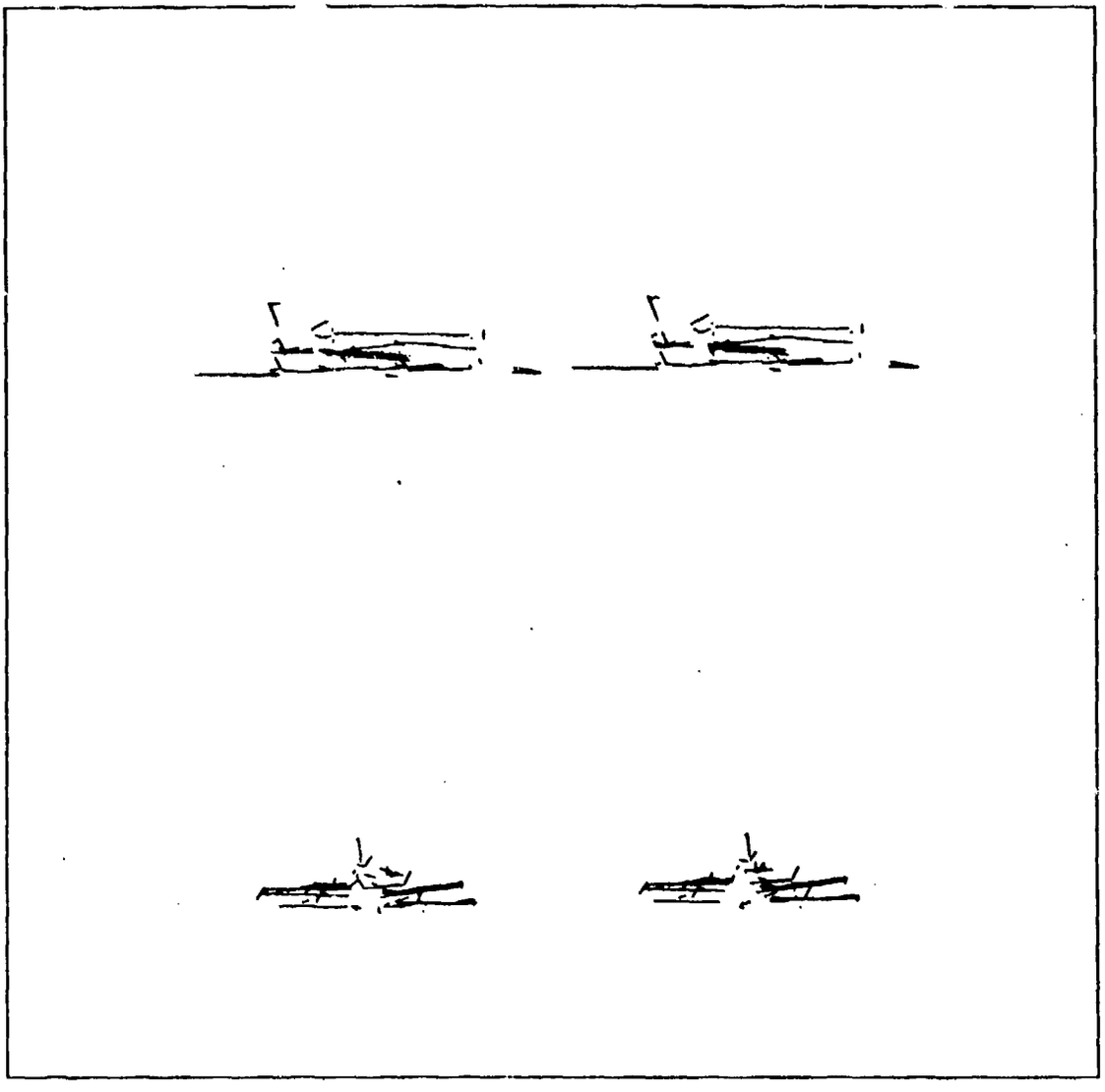


Figure 3-22: *These stereograms show the same edges as Figure 3-21, but from ground level (0 degrees).*

Errors

The most common type of errors occur from slight misregistration of the images, or errors in edge detection. Thus we require edges to overlap significantly before accepting their interpretation. Another common form of error is the detection of short edges in one view but not in the other. Filtering out short edges avoids this "noise" without deleting large objects. With the current scheme of applying slices at uniform intervals, these two conditions are approximately equivalent to requiring a minimum number of samples or slices on both edges.

Another major source of error is due to positional inaccuracy on edges that are nearly parallel to the stereo axis. The error in stereo disparity, e_d, is approximately:

$$e_d = \frac{e_p}{\sin\theta} \qquad (3-13)$$

where:

e_p is the error in edge position (perpendicular to the edge), and

θ is the angle the edge makes with the epipolar line.

Matched edges whose angles are close to zero tend to have wild disparities, so these are omitted from the display.

Finally, due to alignment errors, the endpoints of the edge curves will generally not have identical y-coordinates. One or both edges are shortened to make this condition true, i.e., to give 100% overlap.

FUTURE DIRECTIONS

4.1 Extensions and Unfinished Work

It is often true that research raises more questions than it answers. There are a number of directions future work could follow from the state reported in this thesis. Some of these extensions fit in easily to the framework developed; some require restructuring.

4.1.1 – Surfaces

The data structure used by our Viterbi algorithm allows for the relating of edges and surfaces. For example, an edge that lies "on" a surface is given a different representation than an edge that is separated from a surface by a spatial discontinuity. In the current implementation, these states are distinguished only by weak constraints (surface breaks and excess length), and none of this information is preserved in the main data structure or checked for consistency across slices.

To make use of this surface-edge information, more work needs to be done on the constraints that affect it. For example, "T" junctions usually imply spatial discontinuities, with the surface along the top of the T "in front of" the two surfaces along the stem of the T. Such information can be incorporated into the stereo system as it is developed, and the data structures reorganized to preserve and use it. Some of this work in the area of "shape from shape" is being done by Binford and Lowe [Lowe 1981] and Liebes [Liebes 1981].

4.1.2 – Equivalent Match Relations

Another problem that needs work is the classification of match interpretations across epipolar slices. We have defined only simple equivalence relations, and the consistency checking will find no information if the adjacent epipolar slices generate match pairs that are not in one of the simple equivalence relations we have defined. A more complete analysis of line drawings in stereo would yield a larger and more complex set of relations.

4.1.3 – Viterbi Extensions

More work is possible on the Viterbi algorithm itself. In particular, its greatest shortcoming is the requirement that every edge crossing an epipolar line be explained geometrically. However, extraneous or missing edges due to noise or misregistration cannot be explained this way. It would be useful if the Viterbi algorithm could be extended to edit such edges out. All of our efforts to accomplish this have resulted in an unmanageable increase in complexity of both time and space.

4.1.4 – Evaluation Function

There should also be more theoretical work on the evaluation function. While the most important components have resulted from analytical work, others are *ad hoc*, and there is no unifying theory for combining the various components.

The two most important numeric constraints, edge intervals and edge angles, have been derived to map between distributions in object space and distributions in image space. However, the implementation has assumed uniform distributions in both cases. It should be possible to use *a priori* knowledge of the scene to estimate a more accurate feature distribution, e.g., many vertical and horizontal surfaces and edges. This would translate into even stronger constraints on the image parameters.

4.1.5 – Area-based Stereo

One component in particular that has been greatly oversimplified is the use of brightness information. We have intentionally limited our use of this source in order to better study the problems of edge-based stereo. In the man-made scenes we were concerned with, edges were dominant, and intensities could often be misleading. In any stereo system that hopes to be general, however, intensity-based (area-based) techniques will be required. An obvious compromise is to use both, since there are places, often in a single scene, where each is superior. Certainly, the use of brightness in our system could be extended beyond a single value per surface.

4.1.6 – Edge Curves

Our implementation has concentrated on edges, and to simplify the problem we have assumed edge curves comprise straight line segments. This assumption is not essential, and could be relaxed to include curved segments or splines. All of the essential inputs to the constraint calculations – edge length, end point position, vertex types, angle with respect to a given epipolar line – are also available with curves.

4.1.7 – Camera Model

Much of the preprocessing effort goes to determine camera model parameters and to register the images. As we noted, it is necessary in these steps to solve the stereo correspondence problem for a selected number of points before all the parameters can be determined. This leads to a circular sort of problem which is resolved only by the fact that the stereo correspondence required for the camera model solution is much more limited than the full correspondence in that most parameters are known *a priori* to some approximation..

However, we have noted that many of the errors seen in our examples result from small errors in these parameters. Ideally, there should be a feedback process where a byproduct of the matching is a refinement of the camera parameters, which leads to a better match, and so forth. The use of vertex information is well suited for this feedback, for once an edge is matched, a correspondence is set up for any vertices to which it belongs. If two vertices correspond, any difference in their y-coordinates is one error measure for the camera model parameters at that location in the image. This can lead to a correction matrix capable of accounting for and correcting many types of geometric distortion.

APPENDIX

5.1 Local Context System

This section summarizes an earlier stereo system that was reported in 1978 [Arnold 1978]. This system represents the first use of edge continuity as a constraint in feature-based stereo. The initial processing steps, through the ground plane finder, are used in the current system to determine camera model parameters.

Stereo images were digitized from small regions of 9x9 inch black and white aerial photograph negatives. To reduce processing and memory requirements, these were normally reduced to 128x128 pixels. Subjects included commercial aircraft at a terminal in San Francisco Airport, cars in a parking lot, and an apartment building complex.

A camera model and ground plane were calculated from the data in the images in a process which was entirely automated. An Interest Operator [Moravec 1977] was applied

Figure 5-1: *A 128x128x8 bit image pair was used, showing an L-1011 at San Francisco Airport.*

to the left view to select approximately 50 "interesting" points. A point was "interesting" if it promised to be easily locatable in two dimensions (i.e., corners and intersections). A fast binary search correlator [Moravec 1977] produced an initial match for each point, searching the entire right image each time.

These matches were refined with a high resolution area correlator [Gennery 1977] and passed to a camera model solver [Gennery 1977]. This camera model program solved for four parameters:

1) direction of the stereo axis
2) relative rotation between left and right views
3) scale factor between left and right views
4) translation perpendicular to the stereo axis

The usual camera solver determines 5 parameters. The special form we used is useful in the degenerate case in which scene heights are small with respect to distance from the film plane.

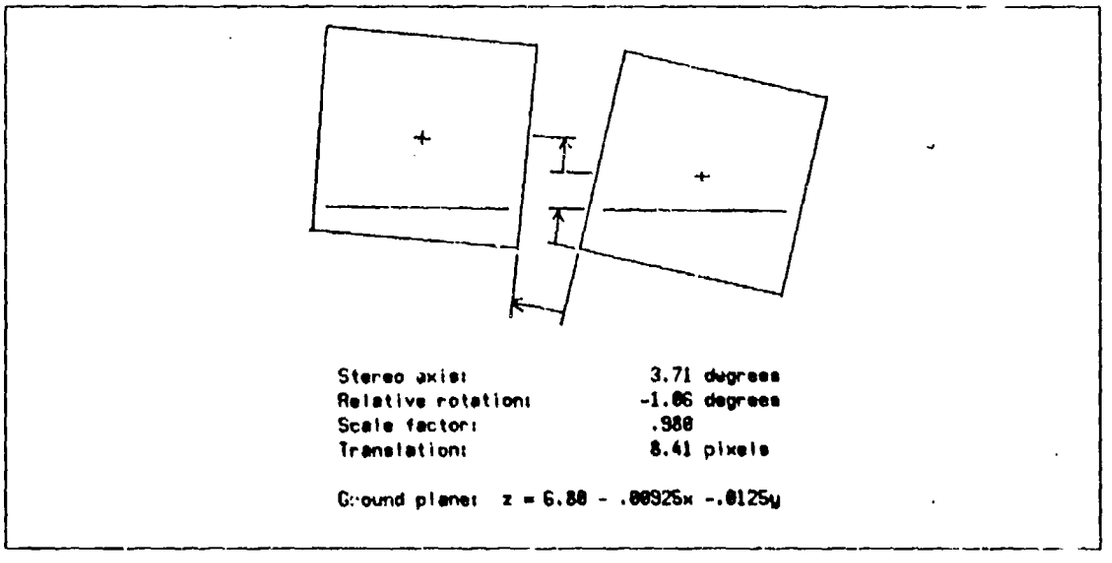


Figure 5-2: *The camera model and ground plane solvers produced four image parameters and the equation for a plane.*

The relative positions (disparities) of each matched pair along the stereo axis provided information on heights relative to the film plane. At this stage, about half the original 50 points had been automatically rejected for various reasons, and we relied on the remainder to be evenly distributed in the scene. The points and their heights were given to a ground plane finder [Gennery 1977] which attempted to fit a plane to a subset of them such that a few points were assigned heights above the plane, fewer below the plane, and as many as possible on the plane. The total processing for the camera model and the ground plane was about 8 seconds on a PDP-10.

The next step was to raster-scan an edge operator over the two pictures to extract all usable edges. We used the Hueckel operator [Hueckel 1973], with an operator radius of 3.19 (32 pixels area). The Hueckel operator produces several accurate measurements which can be useful in discriminating matches, including a measurement of angle that is more accurate than other operators. Of this information, we retained for each edgel the x-y position, angle of edge, and brightness of minus and plus sides. About 1200 edgels were produced from a 128x128 pixel picture in about 18 seconds.

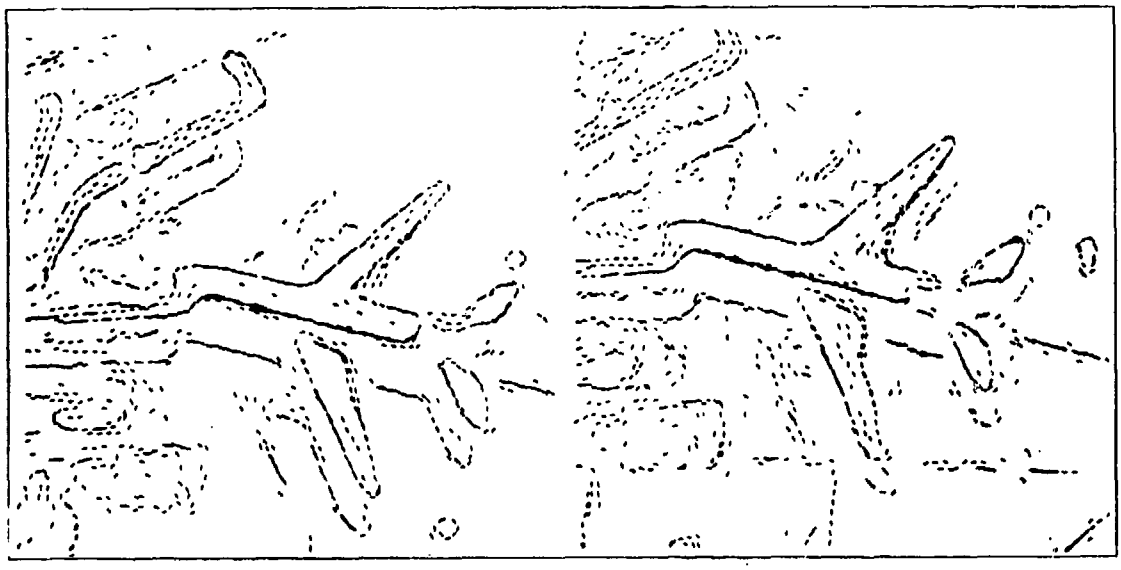


Figure 5-3: *The Hueckel edge operator produced about 1200 edgels in each view.*

At this point, all information was contained in the edge files, and the original images were set aside. The edges from the left and right pictures were then adjusted with the camera model and ground plane parameters to a standard coordinate system with the stereo axis in the x direction and disparity shifts due to the tilt of the ground plane cancelled. Thus all points lying on the ground plane had identical x and y coordinates in the two views.

We then proceeded to match edges in the left (master) image with those in the right, and extract a local context for each edge in the left. A grid of 8x8 pixel cells was set up for the left and right pictures, each cell being the head of a linked list. Edge records were read in and linked to an appropriate cell based on the x and y coordinates of the edgel. For these pictures, the linked lists had an average length of about 4.

For each edgel in the left picture, we wanted to find a list of possible matching edgels in the right picture. The search was constrained to those edgels within a narrow band, about 6 pixels wide in the y direction. The band started at the x coordinate of the left edgel (zero disparity) and extended to an *a priori* disparity limit in the x direction.

For edgel pairs within the band, differences in brightness and angle were thresholded to determine whether to accept or reject a potential match. If the match was accepted, a disparity was calculated by extending the right edgel to intercept the y coordinate of the left edgel. On the average, this search produced 8 ambiguous matches for each edgel, that is, 8 edgels that agree in position, angle and brightness. Most of these ambiguous matches were actually multiple edgels from the same scene edge, with slight deviations in disparity due to noise. From this point on, no further information was obtained from the right edge file.

For local context, we wanted a list of edgels in the left picture that probably lay on the same physical edge of the object. Again, a scan ran through all edgels on the left,

and a search was made for each one, this time in the left grid. Two edgels were linked if certain loose conditions were met:

1) x and y coordinates matched within 3 pixels,
2) their angles matched within 90 degrees,
3) the angle of a line connecting edgel centers lay between the individual edgel angles,
4) brightnesses were consistent on at least one side of the edgels.

Typically, this produced 3 or 4 links per edgel, and linked edgels tended to follow edges of low to moderate curvature (see Figure 5-4.) The time for the matching and linking was 33 seconds.

We then had for each edgel in the left picture a list of possible disparities and a list of neighboring edgels which were linked to it. The problem was to choose a disparity for each edgel in such a way that disparities were consistent along linked edges. We implemented an *ad hoc* "voting" scheme whereby each disparity on the edgel's list was a candidate, and only those neighbors which were linked could vote (see Figure 5-5).

The voting proceeded as follows: Let E be an edgel and L an edgel linked to E. Let d_L be a disparity on L's disparity list and d_E a disparity on E's disparity list. If d_L and d_E were equal or nearly equal (within .125 pixel disparity) then d_E got two votes. If d_L and d_E were close (within .375 pixel disparity) then d_E got 1 vote. Otherwise, there were no votes.

This loose condition for voting compensated for quantization error in the recording of disparities and allowed multiple edgels from a single edge to reinforce. After all the voting, a bell-shaped distribution usually resulted about the best disparity, with wild or inconsistent matches out on the tails of the curve. The maximum of the distribution

was taken as the disparity for E. This processing took 8 seconds. We could then output a file of edgels with their three dimensional locations.

The system outlined above suffered from some serious problems. It relied heavily on the edge operator, which was deficient in several respects. It was susceptible to slow gradients, at which it found a multitude of parallel edges that tended to match at every possible disparity. Because it was a least squares process, it was easily led astray, for example, near corners. This system also made very weak use of constraints other than continuity (e.g., brightness and edge angle).

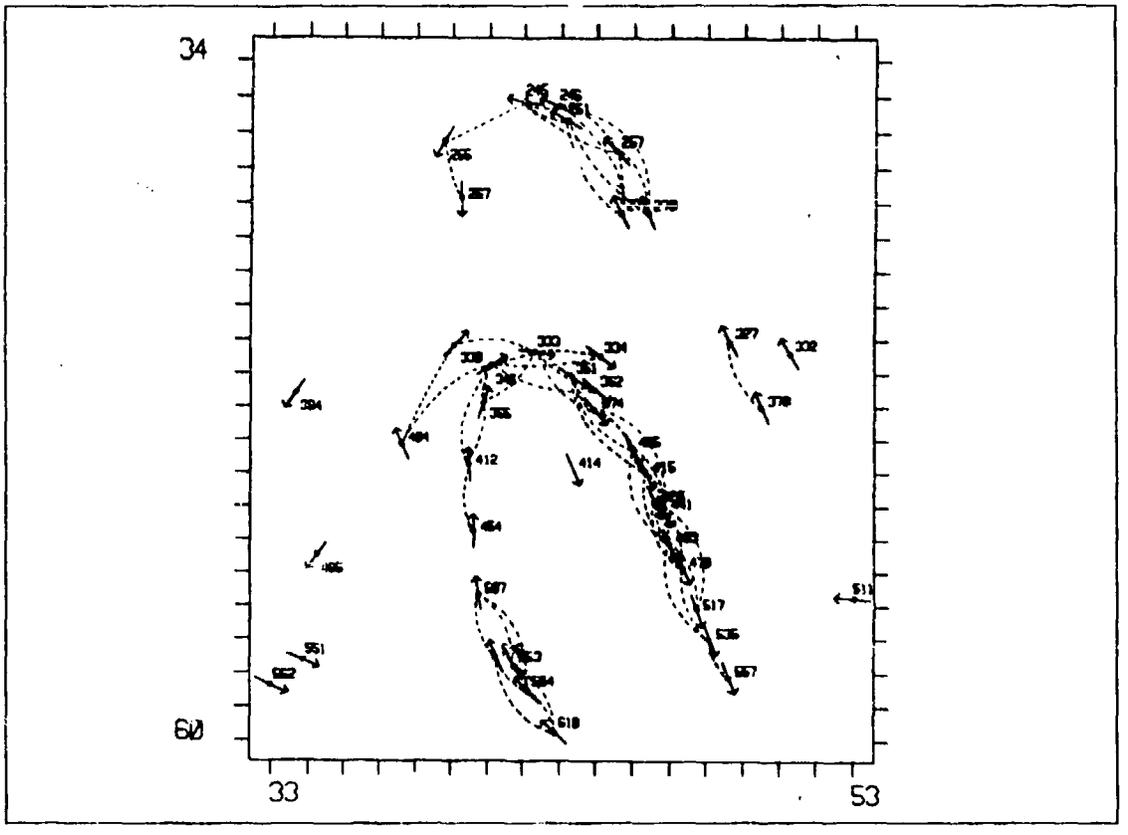


Figure 5-4: *This plot of edgels is from the left view of the aircraft images, near the left stabilizer and its shadow. X and Y axes are in units of pixels (octal), and dotted lines represent the links between edgels used for local context.*

Nevertheless, the system produced some useful depth maps and the results were encouraging in many respects. Although there were many edgels, over 90% of them were correctly matched. The depth map for the aircraft images provided clear separation of the ground from the plane, and resolved different parts of the plane according to their height above the ground: wings, fuselage, stabilizer and boarding ramp. Even the dihedral angle of the main wings was apparent; edgels at the wing tips had greater disparity than edgels near the fuselage.

```
Edge: 345;  Disparities: 34,40,54,60;
                 Links: 333,365,404,412,334,362;

Edge: 365;  Disparities: 40,44,46,55,76;
                 Links: 333,345,412;

Edge: 412;  Disparities: 41,41,42,45,75;
                 Links: 345,365,454;

Edge: 454;  Disparities: 42,42,42,46,50,64,112;
                 Links: 412;

                Voting tally for 412:

     Disp.    345     365     454     Total

     41       ||      |||     ||||||  11*
     42       |       ||      ||||||  9
     45               ||||    |||||;
     75               ||                *
```

Figure 5-5: *A portion of the data structure produced by the matching program shows a sample voting. The edgels are selected from those in figure 5-4. (All numbers are in octal).*

REFERENCES

[Arnold 1978] Arnold, R. David, "Local Context in Matching Edges for Stereo Vision", *Proceedings of the ARPA Image Understanding Workshop*, Boston, May 1978, 65-72. (*cited on pp.* 4,64,68,113)

[Arnold 1980] Arnold, R. D., and T. O. Binford, "Geometric Constraints in Stereo Vision", *Soc. Photo-Optical Instr. Engineers*, Vol. 238, Image Processing for Missile Guidance, 1980, 281-292. (*pp.* 3)

[Baker 1981] Baker, H. Harlyn, "Depth from Edge and Intensity Based Stereo", University of Illinois, Ph.D. thesis, September 1981. (*pp.* 6)

[Ballard 1982] Ballard, Dana H., and Christopher M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, N.J., 1982. (*pp.* 9)

[Bellman 1957] Bellman, Richard, *Dynamic Programming*, Princeton University Press, 1957. (*pp.* 8)

[Binford 1981] Binford, Thomas O., "Inferring Surfaces from Images", *Artificial Intelligence*, Vol. 17, August 1981, 205-244. (*pp.* 43,99)

[Burnside 1979] Burnside, C. D., *Mapping from Aerial Photographs*, John Wiley and Sons, New York, 1979. (*pp.* 8)

[Denardo 1982] Denardo, Eric V., *Dynamic Programming Models and Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1982. (*pp.* 8)

[Dreyfus 1969] Dreyfus, Stuart E., "An Appraisal of Some Shortest-Path Algorithms", *Oper. Res.*, 17, 3, pp. 395-412, May-June 1969. (*pp.* 52)

[Forney 1973] Forney Jr., G. David, "The Viterbi Algorithm", *Proc. IEEE*, Vol. 61 No. 3, March 1973. (*pp.* 8,48)

[Fox 1973] Fox, B. I., "Calculating *k*th Shortest Paths", *INFOR*, 11, pp. 66-70, 1973. (*pp.* 52)

[Gennery 1977] Gennery, Donald B., "A Stereo Vision System for an Autonomous Vehicle", *Fifth Int. Joint Conf. on Artificial Intelligence*, MIT, Cambridge, Mass., August 1977, 576-582. (*pp* 114,115)

[Gennery 1980] Gennery, Donald B., "Modelling the Environment of an Exploring Vehicle by Means of Stereo Vision", Ph.D. thesis, Stanford Artificial Intelligence Laboratory, AIM-339, June 1980. (*pp.* 2,5,11,69)

[Grimson 1980] Grimson, William Eric Leifur, "Computing Shape Using a Theory of Human Stereo Vision", Department of Mathematics, MIT, Ph.D. thesis, June 1980. (*pp.* 6)

[Henderson 1979] Henderson, Robert L., Walter J. Miller, C. B. Grosch, "Geometric Reference Preparation Interim Report Two: The Broken Segment Matcher", Henderson, R. L., Rome Air Development Centre, Rome, New York, RADC-TR-79-80, April 1979. (*pp.* 5)

[Hoffman 1959] Hoffman, W. and R. Pauley, "A method for the Solution of the Nth Best Path Problem", *JACM*, 6, pp. 506-514, 1959. (*pp.* 52)

[Hueckel 1973] Hueckel, M., "A Local Visual Operator which Recognizes Edges and Lines", *JACM*, 20, 634 (1973). (*pp.* 115)

[Julesz 1971] Julesz, B., *Foundations of Cyclopean Perception* University of Chicago Press, 1971. (*pp.* 8)

[Julesz 1975] Julesz, B., "Experiments in the Visual Perception of Texture", *Sci. Am.*, 232, pp. 34-43, 1975. (*pp.* 8)

[Liebes 1977] Liebes Jr., S. and A. A. Schwartz, "Viking 1975 Mars Lander Interactive Computerized Video Stereophotogrammetry", *Journal of Geophysics Research*, 82, No. 28, 4421, Sept. 30, 1977. (*pp.* 11)

[Liebes 1981] Liebes Jr., S., "Geometric Constraints for Interpreting Images of Common Structural Elements: Orthogonal Trihedral Vertices", *Proceedings of the ARPA Image Understanding Workshop*, 1981. (*pp.* 109)

[Lowe 1981] Lowe, David G., and Thomas O. Binford, "The Interpretation of Geometric Structure from Image Boundaries", *Proceedings of the ARPA Image Understanding Workshop*, April 1981, 39-46. (*pp.* 43,109)

[Lowe 1982] Lowe, David G., Documentation for DESIGN program, online in DESIGN.DOC[TER,DLO] at SAIL, Stanford. (*pp.* 73)

[Marimont 1982] Marimont, David H., "Segmentation in ACRONYM", *Proceedings of the ARPA Image Understanding Workshop*, Stanford University, September 1982. (*pp.* 73)

[Marr 1982] Marr, David, *Vision*, W. H. Freeman and Co., San Francisco, 1982. (*pp.* 9)

[Marr 1977] Marr, D. and T. Poggio, "A Theory of Human Stereo Vision", MIT Artificial Intelligence Memo No. 451, November 1977. (*pp.* 6)

[Moravec 1977] Moravec, H. P., "Towards Automatic Visual Obstacle Avoidance", *Proc 5th IJCAI*, 1977. (*pp.* 113,114)

[Moravec 1980] Moravec, Hans P., "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover", Ph.D. Thesis, Stanford Artificial Intelligence Laboratory, AIM-340, September 1980. (*pp.* 2,5,11,69)

[Nudd 1977] Nudd,G. R., P. A. Nygaard and J. L. Erickson, "Image-Processing Techniques Using Charge-Transfer Devices", *Proceedings of the ARPA Image Understanding Workshop*, Palo Alto, October 1977. (*pp.* 1)

[Panton 1981] Panton,D. L., C. B. Grosch, D. G. DeGryse, J. Ozils, A. E. LaBonte, S. B. Kaufmann, L. Kirvida, "Geometric Reference Studies", RADC-TR-81-182, Final Technical Report, July 1981. Vol. 44, No. 12, December 1978, 1521-1536. (*pp.* 6)

[Slama 1980] Slama, C. C., editor-in-chief, *Manual of Photogrammetry*, American Society of Photogrammetry, Falls Church, Va., 1980. (*pp.* 8)

[Viterbi 1967] Viterbi, A. J., "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Trans. Inform. Theory*, Vol. IT-13, pp. 260-269, Apr. 1967. (*pp.* 8)

[Viterbi 1979] Viterbi, Andrew J., and James K. Omura, *Principles of Digital Communications and Coding*, McGraw-Hill, New York, 1979. (*pp.* 8)

[White 1969] White, D. J., *Dynamic Programming*, Holden-Day, San Francisco, 1969. (*pp.* 8)

[White 1978] White, D. J., *Finite Dynamic Programming*, John Wiley and Sons, 1978. (*pp.* 8)

[Wylie 1970] Wylie Jr., C. R., *Introduction to Projective Geometry*, McGraw-Hill, New York, 1970. (*pp.* 8)