7-1-1988

Rule-Based Approach to Binocular Stereopsis

S. Tanaka
Purdue University

A. C. Kak
Purdue University

A Rule-Based Approach to Binocular Stereopsis

S. Tanaka
A. C. Kak

TR-EE 88-33
July 1988

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

A Rule-Based Approach to Binocular Stereopsis

S. Tanaka

A. C. Kak

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

TR-EE 88-33
July 1988

This work was supported partly by a Bridgestone Corporation fellowship and partly by the Army Night Vision and Electro-Optics Laboratory with contract DAAK20-85-C-0293

TABLE OF CONTENTS

ABSTRACT

This research is motivated by a desire to integrate some of the diverse, yet complimentary, developments that have taken place during the past few years in the area of passive stereo vision. On the one hand, we have approaches based on matching zero-crossings along epipolar lines, and, on the other, people have proposed techniques that match directly higher level percepts, such as line elements and other geometrical forms. Our rule-based program is a modest attempt at integrating these different approaches into a single program. Such integration was made necessary by the fact that no single method by itself appears capable of generating usable range maps of a scene.

CHAPTER I INTRODUCTION

Before the famous random-dot stereogram experiments by Julesz [1] , it was generally believed that a necessary precursor to binocular fusion was a recognition of monocular cues in each of the two images. The experiments by Julesz caused a paradigm shift of sorts in the psychophysics of the human visual system; suddenly, the preponderance of the research effort shifted toward explaining practically all aspects of human stereopsis in terms of low-level processes, as opposed to high-level cognitive phenomena. One of the high points of this post-Julesz period was the development of the Marr-Poggio paradigm [2]. Marr and Poggio presented a computational theory, later implemented by Grimson, that provided successful explanation of the Julesz experiments in terms of the matchings of zero-crossings at different scales, the zero-crossings at each scale corresponding to the filtering of the images with a Laplacian-of-a-Gaussian (LoG) filter [3].

During the last few years it has been recognized that while the Marr-Poggio paradigm may be an elegant model of the low level mechanisms for generating depth information, not to be ignored are the higher level cognitive phenomena that are also capable of producing the same type of information. In a majority of these higher level phenomena, there is first an explicit recognition of monocular cues in each image; depth perception is then generated either directly from the monocular cues, or through their fusion in a stereo pair.

In this report, we will present a rule-based procedure that makes a modest attempt at combining the Marr-Poggio type of low-level processing with higher level knowledge-based processing that takes into account a priori knowledge of the geometric nature of object surfaces involved in the binocular fusion process. In our current effort, the higher level knowledge based processing is limited to scene with planar surfaces. One could say that at this time we have limited ourselves to a polyhedral world. However, it is not hard to conceive of ways in which the present method could be extended to worlds with different types of curved surfaces simultaneously present.

Basically, our approach represents an integration -- in an opportunistic framework -- of the following methods for stereo matching: 1) dominant feature matching, 2) matching under geometric constraints, 3) zero-crossing contour matching, and 4) the full MPG matching. A rationale behind the integration is that while none of these methods by themselves is capable of yielding an adequately dense range map, their synergistic combination can be expected to provide more useful input for scene interpretation. Another rationale is that the first three methods provide us with mechanisms to inject higher level constraints into the MPG process. The only dominant features currently implemented are the straight line features; our implementation of the method for such features was inspired by the work of Medioni and Nevatia [5] and Ayache et al. [4]. For the geometric constraint based matching, we have been much inspired by the work of Eastman and Waxman [6] and Hoff and Ahuja [7]. The only geometric constraints currently incorporated in our system are those that are given rise to by planar surfaces. For zero-crossing contour matching -- also called matching under figural continuity constraints -- we have learned much from the work of Mayhew and Frisby [8] and Grimson [9]. Our implementation of figural continuity is different but retains the spirit espoused by the earlier efforts.

The framework for the integration of the methods is opportunistic in the sense that given all the available matching strategies, at each stage of processing the system invokes the one whose applicability conditions are best satisfied by evidence extracted so far from the image region. For example, if at the beginning of processing a pair of matchable dominant features is available in the two images, the system will go ahead and utilize them for the determination of the local depth information. However, in the absence of dominant features, the system will try to invoke other matching strategies depending upon the zero-crossing features extracted from the image region.

The opportunistic framework for integration, implemented as a rule-based program, is intended to capture human intuitions about how higher level constraints should be injected into a low-level matching procedure such as the MPG algorithm. Our intuition, which is in concurrence with the BRPS conjecture made by Mayhew and Frisby [8], says that if a strong high-level feature can be extracted from an image, the human visual system will go ahead and do so; the stereoptic fusion will be subsequently driven by the constraints generated by the high-level feature.

We do not wish the reader to consider our approach a mere smorgasbord of the already well-known approaches to stereo matching. The different component approaches in the rule-based system are not a complete duplicate of their earlier implementations by other investigators. As a case in point, when matching under geometrical constraints is invoked, only the output of fine zero-crossing channel is constrained by the expected geometric form of the object surface. This is contrary to the implementations proposed by Waxman and Eastman [6] and Hoff and Ahuja [7] where geometrical constraints are also applied to coarse-channel outputs. Our rationale is that the analytical properties of the coarse channel output may bear little resemblance to the analytical properties of the object surface. For example, for a planar object like a cube against a uniform background, the fine channel output, if free of mismatch errors will correspond fairly closely to the three dimensional shapes of the visible surfaces of the cube; however, the coarse channel output will, in most cases, look like a blobular bell shaped function.

The above discussion leads us to the following dilemma: Ideally, any available geometrical constraints generated by hypothesized knowledge of object surfaces should be applied to zero-crossings at all scales; however, due to the large smoothing operators that come into play for coarse channels, the geometrical properties of coarse-channel disparities may not correspond to those of object surfaces. And, of course, applying the geometrical constraints to just the fine channel output is hazardous because of the mismatches in the coarse channel output that could not be eliminated by not applying any geometrical constraints in these channels.

To get around these difficulties, we have taken the following approach: Geometrical constraints are hypothesized and applied only if certain "robust" features are detected at the outputs of the finest channel. For example, we consider strong straight edges as robust features. If, at the output of the finest channel, the system detects zero-crossing contour segments that are straight and represent large changes in image brightness levels, the system then fuses these segments by a contour matching scheme. This fusion generates a straight line in the 3-D space of the scene. Geometrical constraints are then enforced by insisting that in the vicinity of this line in 3-D space, the disparities associated with the other zero-crossings be accepted only if the disparity values are close to that of the line.

There is a further mechanism built into our computational procedure for discarding a selected geometrical hypothesis : If in the vicinity of the zero-crossings generated by straight edges, a majority of the other zero-crossings cannot be fused under the disparity constraint just mentioned, the hypothesized geometrical constraint is discarded, and a search conducted for an alternative hypothesis.

Obviously, binocular fusion under geometrical constraints generated by straight edges would not be applicable everywhere in a scene. In other regions, where applicable, we invoke the constraint generated by figural continuity. And if that also is impossible, our rule-based system used as a last resort the straightforward MPG matching scheme.

Evidently, in any passive stereo scheme, the nature of illumination has a considerable influence on the final results. We will report on experiments conducted with both natural lighting and under artificial "unstructured" light illumination. The latter type of illumination was pioneered by Nishihara [10, 11] in the PRISM system.

Since what we have done can, in a sense, be considered to be a modification of the Marr-Poggio-Grimson (MPG) procedure, in what follows we will begin with a brief review of this procedure in Chapter 2, where we also discuss some of the shortcomings associated with the MPG method. Hopefully, the discussion in Chapter 2 will convince the reader that there is a need to also invoke higher level constraints during the matching process. In Chapter 3, we will review the procedures that are currently available for enforcing high-level constraints. In Chapter 4, we present in detail the matching algorithms included in the rule-based procedure. Chapter 5 then proceeds to present the organization of the rules in the system; this chapter also discusses the interaction of the rules with a control matrix which is used by the system to figure out which matching strategies have already been tried for a given region of the image and which strategy should be tried next. In Chapter 6, we then discuss some aspects of the implementation of the system, particularly those dealing with the interaction of OPS83 for the rule based part and the C language for the rest of the image processing routines. In Chapter 7, we show some experimental results. Finally, concluding remarks are made in Chapter 8.

CHAPTER II THE MPG APPROACH TO BINOCULAR FUSION

2.1. Brief Review of the Coarse-to-Fine Matching Strategy

Figure 1 illustrates what we usually refer to as the MPG process. Each retinal image is first convolved with a set of LoG functions, the function being of the form :

$$\nabla^2 G(x, y) = \left[\frac{x^2 + y^2}{\sigma^2} - 2 \right] \exp\left[-\frac{x^2 + y^2}{2\sigma^2} \right] \tag{1}$$

where ∇^2 stands for the Laplacian operator

$$\nabla^2 = \frac{\delta^2}{\delta x^2} + \frac{\delta^2}{\delta y^2} \tag{2}$$

and G(x,y) represents the smoothing kernel

$$G(x,y) = \sigma^2 \exp\left[-\frac{x^2 + y^2}{2\sigma^2} \right] \tag{3}$$

Although the "width" of the LoG function, as described above, is characterized by the parameter σ, it is more frequently referred to by a parameter usually denoted by w_{2D}, the relationship between the two being

$$\sigma = \frac{w_{2D}}{2\sqrt{2}} \tag{4}$$

The convolving functions in Figure 1 correspond to w_{2D} values of 63, 32, 16, and 8. As Grimson has pointed out, in the HVS there is physiological evidence for the presence of a fifth channel with w_{2D} equal to 4 [12].

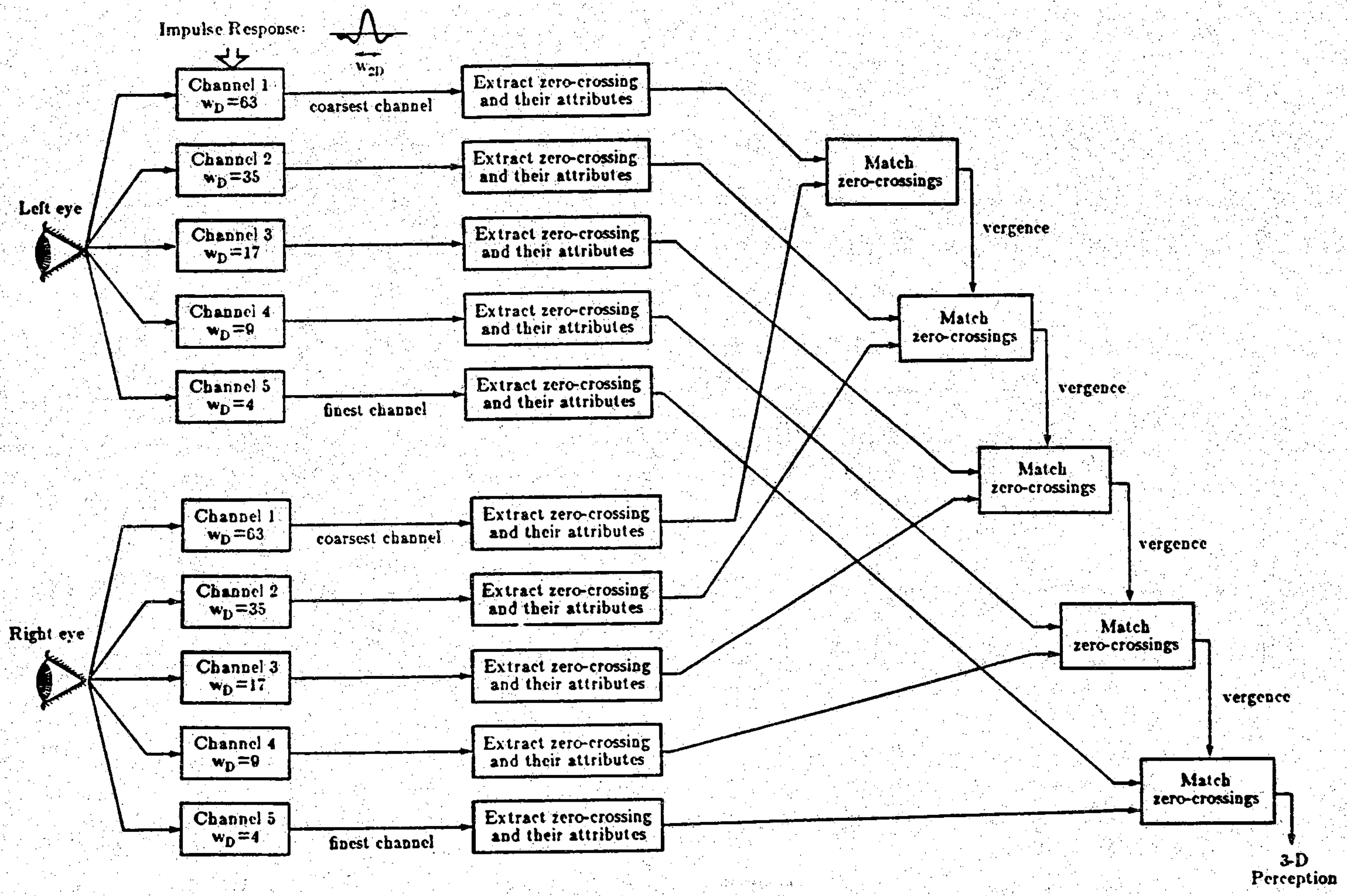


Figure 1 Marr - Poggio algorithm.

For each image, the convolved outputs form a coarse-to-fine representation. Large scale variations manifest themselves via zero-crossings only in the coarser (large w_{2D}) channels, while the small scale variations generated by surface textures "show" mostly in the outputs of the finer channels. In Figure 1, there is an order to the matching of zero-crossings, which may or may not be representative of the human visual system but forms a good strategy for engineering implementations. As depicted in the figure, we first match the zero-crossings, taking into account the local orientations of the zero-crossing contours, in the coarsest channel. [Because of the nature of our rule-based superstructure, it is not necessary for us to directly enforce figural continuity constraints in the matching process, as was, for example, done by Grimson [9]. In other words, the figural continuity considerations are subsumed by the rule-based procedure.] The disparity values thus generated form a coarse range map of the scene and are used for vergence control for the next channel -- for a computer implementation, vergence control merely means that before matching a pair of zero-crossings in, say, the $w_{2D} = 32$ channel, the right-image zero-crossing is shifted by the local disparity value generated by the $w_{2D} = 64$ channel. Of course, it is unlikely that the $w_{2D} = 64$ disparity would be available exactly where it is needed, in that event a neighborhood is constructed around the right-image zero-crossing, and the average of the coarser disparities in the neighborhood is used for vergence control. Further details on such matters may be found elsewhere [12, 13].

2.2. Some Computational Aspects of the MPG Algorithm

The computational steps in each channel of the MPG algorithm consist of the three steps: 1) extraction of zero-crossings, 2) matching of zero-crossings between the two images, 3) disambiguation if multiple candidates are available for matching. In the following subsections, we will briefly discuss the nature of computations involved in each of the steps.

2.2.1. Extraction of Zero-Crossings

In the implementation used for this research, the left and the right images were convolved with three LoG operators for w_{2D} equal to 16, 8 and 4. We did not use any larger operators to save on computational time. The larger operators simply permit the depth range to be greater; in our experiments we limited the maximal depth to what would correspond to a w_{2D} value of 16.

The double convolution implied by the kernel of Eq. (1) has considerable bearing on the computational efficiency of the MPG process. Until recently, the LoG function in Eq. (1) was not considered separable and the convolutions involved had to be implemented as two dimensional integrations. The only way around this difficulty was to approximate the LoG form by a difference of two Gaussion function and exploit the separability of a Gaussian kernel to reduce two dimensional integrations into sequences of one-dimensional integrations. However, more recently, Chen et al. [14] have shown that it is possible to decompose the LoG kernel and express the decomposition as a sum of two parts, each part being separable in its x and y dependences. More specifically, the LoG kernel was shown to be expressible as:

$$\nabla^2 G(x, y) = h_{12}(x, y) + h_{21}(x, y) \tag{5}$$

where

$$h_{ij} = h_i(x) \times h_j(y) \tag{6}$$

$$h_1(\xi) = \sqrt{K}\left[1 - \frac{\xi^2}{\sigma^2}\right] e^{\frac{-\xi^2}{2\sigma^2}} \tag{7}$$

$$h_2(\xi) = \sqrt{K}\, e^{\frac{-\xi^2}{2\sigma^2}} \tag{8}$$

The factor K is a constant, whose quantity does not affect the position of zero-crossings.

2.2.2. Representing Zero-Crossings

Although theoretically the zero-crossing contours are supposed to be closed, in practice if left-right scanning is used to detect the presence of a zero-crossing on each raster line (by looking for sign reversals in the LoG filtered output), the zero-crossings contours come out broken. The breaks are caused by the near-horizontal segments of what would otherwise be continuous contours, since in the vicinity of such segments it is difficult to detect left-to-right sign reversals. In general, breaks can also be caused by the magnitude of the pixel differences on the two sides of a zero-crossing not exceeding an acceptance threshold.

In order to obtain continuous zero-crossing contours, we use the following strategy. All the positive pixels in the LoG filtered image are marked as +1's, and all the negative values as 0's. The zero-crossing contours are then extracted by following the boundaries of the positive regions, where the boundaries are defined as the 4-connectedness neighbor of negative regions. During the contour extraction process, each zero-crossing is tagged as either 'p' or 'n', depending upon whether or not its immediate-left neighbor is lesser or greater than its immediate-right neighbor. If during the the left-to-right comparison, one of the neighbors is on the boundary, the zero-crossing is classified as '0', which is meant to stand for 'other.'

The above procedure results in the formation of two N×N character arrays, $L_w(x, y)$ and $R_w(x, y)$, for the two images, where N is the image size and equals 256 in our system. The contents of these arrays are

$$L_w(x,y) = \begin{cases} 'p' & \text{if a positive zero-crossing exists at (x,y)} \\ & \text{in the left image after the LoG operator is applied} \\ 'n' & \text{if a negative zero-crossing exists at (x,y)} \\ & \text{in the left image after the LoG operator is applied} \\ '0' & \text{else} \end{cases} \qquad (9)$$

$$R_w(x,y) = \begin{cases} 'p' & \text{if a positive zero-crossing exists at (x,y)} \\ & \text{in the right image after the LoG operator is applied} \\ 'n' & \text{if a negative zero-crossing exists at (x,y)} \\ & \text{in the right image after the LoG operator is applied} \\ '0' & \text{else.} \end{cases}$$

2.2.3. Determination of the Search Window Size

As shown in Figure 1, after the zero-crossings are extracted from each of the two images, the outputs of different channels must be matched and the zero-crossings from the left and right images paired in each of the channels. While this matching process is relatively straightforward for the coarsest channel, it must be preceded by vergence for the other channels. In this section, we will first describe how the search windows are set up to match the zero-crossings and the issues affecting this process.

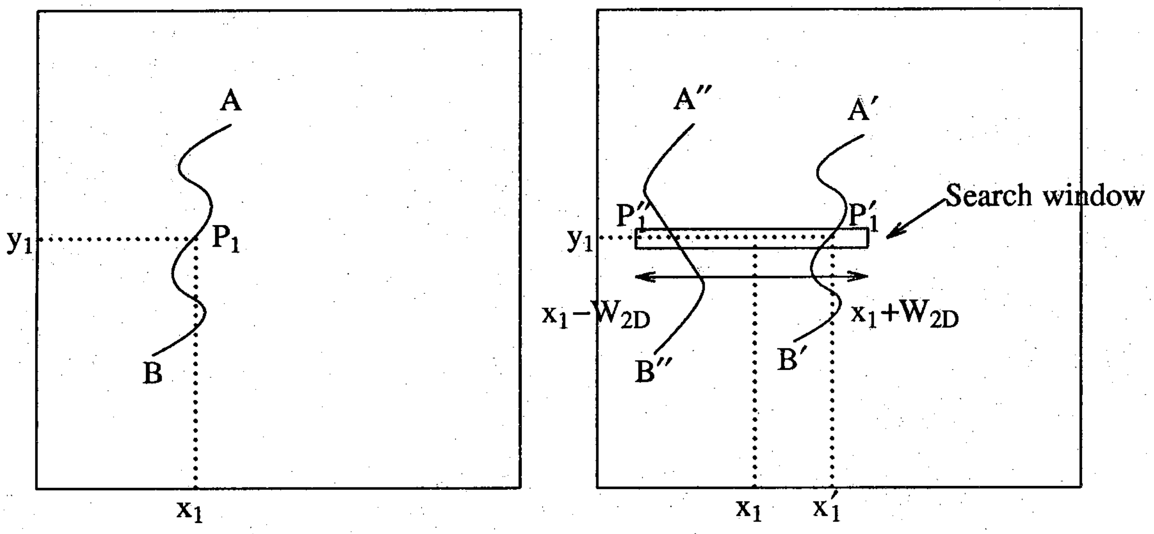
In Figure 2, we show how a search window is set up to find the match in the right image for a left image zero-crossing. We transfer the coordinates of the left image zero-crossing into the right image and then construct a one-dimensional window of width $\pm w_{2D}$ around the resulting point.* Ideally, as was pointed out by Marr and Poggio [3], one should use a search window size of $\pm\frac{w_{2D}}{2}$, since in this case one can show theoretically that with 95% probability there will be only one right-image zero-crossing in the search window. In practice, $\pm\frac{w_{2D}}{2}$ sized search windows are too narrow to be practical, as difficulties are caused by shifts in zero-crossing contours, these shifts being produced by the interaction of gray level changes withing the domain of support of the LoG operator [15]. To demonstrate these shifts, we have shown in Figure 3 the stereo images of a box scene with illumination arranged in such a manner that a shadow is cast on one side of the box in only the left image. To show the effect of this shadow on the shift of the zero-crossing contour, we have used the notion of scale-space filtering, introduced first by Witkin [16], and considered the gray levels on only the line PQ of the images. Figures 4 (a) and 5 (a) show the gray levels along PQ in the two images, respectively. Figures 4 (b) and 5 (b) show the zero-crossings obtained for different choices of w_{2D} for these gray levels. It is clear that the left image zero-crossings suffer greater shifts for larger w_{2D} than the right image zero-crossings, this being a consequence of the presence of the large gray level change at the shadow boundary in the left image.

Due to the zero-crossing contour shifts, it is more common to use $\pm w_{2D}$ search windows, even though this implies that the probability of finding a single zero-crossing within the right image search window will be down to 50% and that in rest of the cases there may be more than one candidate zero-crossing, leading to increased burden on disambiguation procedures.

If only a single zero-crossing is located within the search window, it is accepted as a match provided there is also a match in the orientations of the local zero-crossing contours. If the orientations do not correspond, then the left-image zero-crossing is simply left unmatched.

We will now describe how the zero-crossing orientation information is captured. A method to compute the local contour orientation was presented in a prior study [13]. The approach there consisted of computing the x- and y - directional gradients by the application of Sobel operators and taking the arctan of the two, yielding a value for

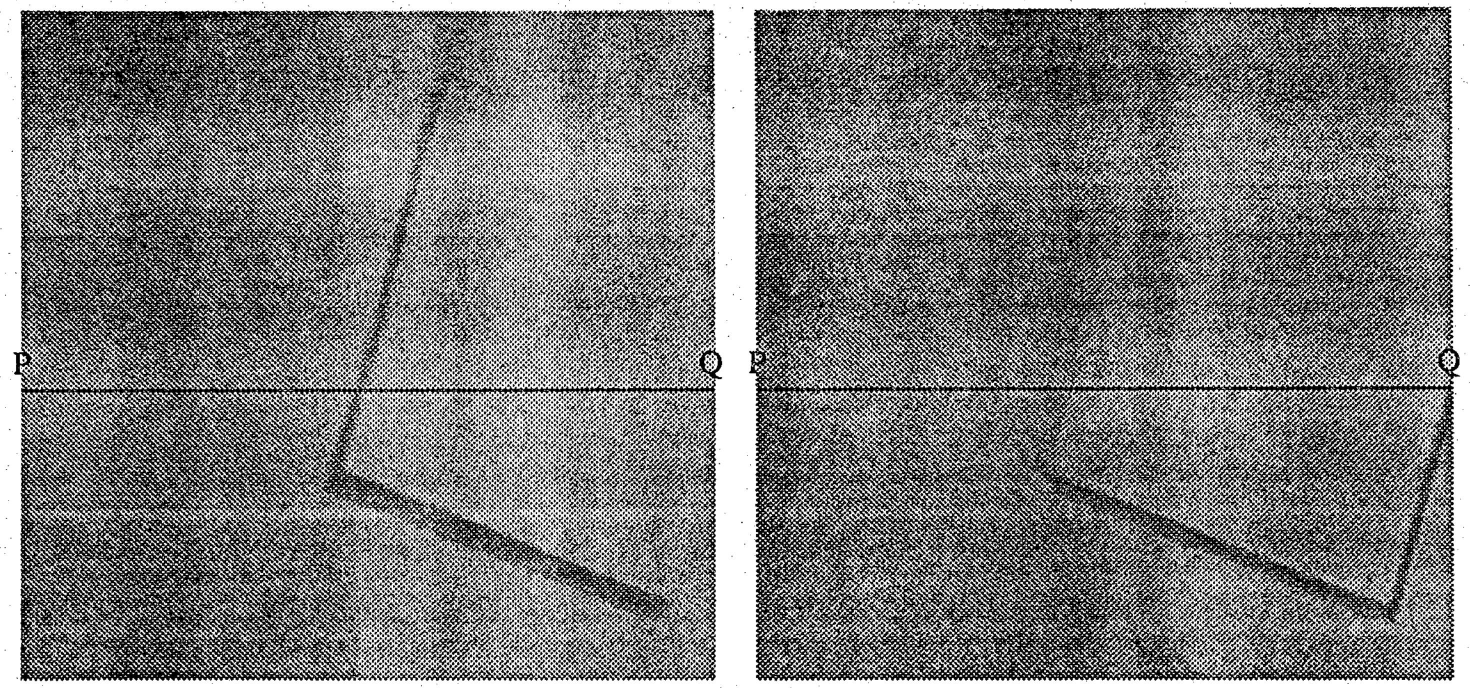
*Strictly speaking, this is only true of the coarsest channel in a multi-channel implementation. For finer channel, we add to the coordinates of the left image the disparities of all the coarser channels before setting up the search window. This, as was mentioned before, is called vergence control.



(a) Left image. **(b) Right image.**

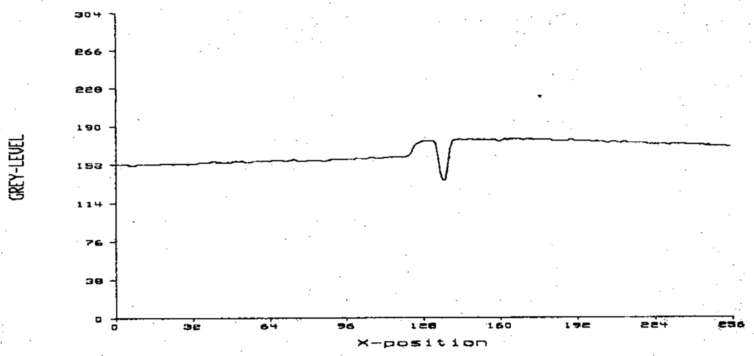
Note: Two contour points P_1' and P_1'' within the search window $\pm\, w_{2D}$, on the coutour segments A′–B′ and A″–B″, resptctively, are the match candidates for the left image zero-crossing P_1.

Figure 2 Search window in the right image for a zero-crossing in the left image.

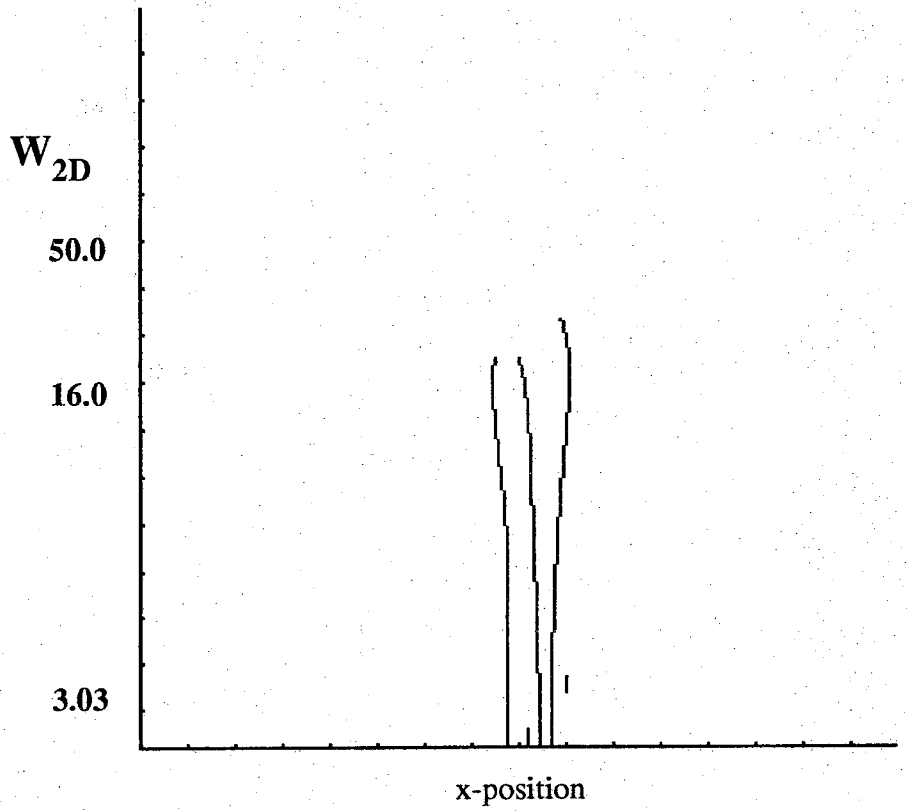


(a) Left image.　　(b) Right image.

Figure 3　One box scene.

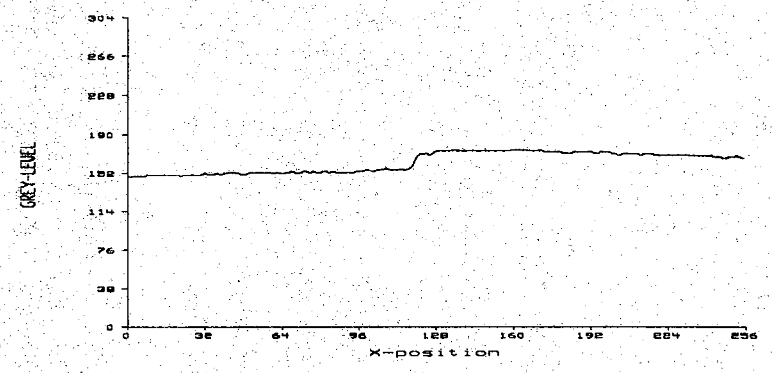


(a) Gray level profile.

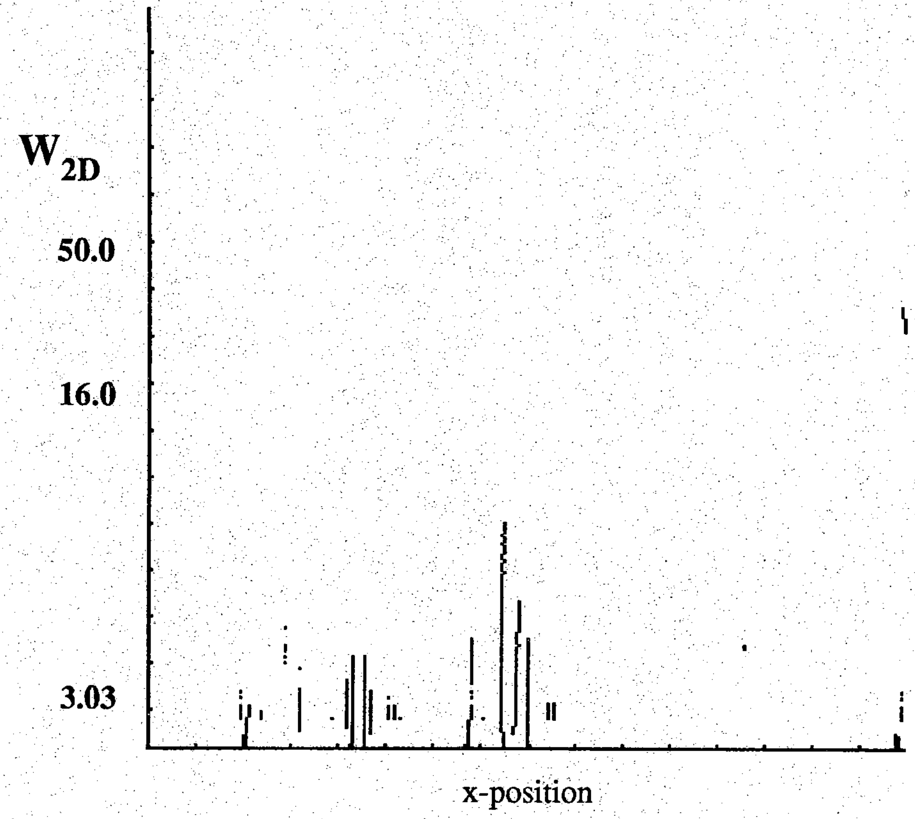


(b) Scale map.

Figure 4   Single blob 1-D image and its scale map (left).



(a) Gray level profile.



(b) Scale map.

Figure 5 Single blob 1-D image and its scale map (right).

orientation which could take any value in the interval [0, 180].

In our implementation of the MPG procedure, we do not directly compute the orientations of the local zero-crossing contours. The contour orientation information is indirectly taken into account by comparing the local binary bit patterns in the binary images generated during the contour extraction procedure presented in Section 2.2.2. We believe this approach is much more efficient.

To explain the comparison of the bit patterns, let A_{il}, and A_{ir} be the values of 3 × 3 binary image patch of left and the right image respectively, as shown in Figure 6. The zero-crossing pixel under consideration is supposedly at the center of the 3x3 matrix. Let S be the matching magnitude computed by the following formula:

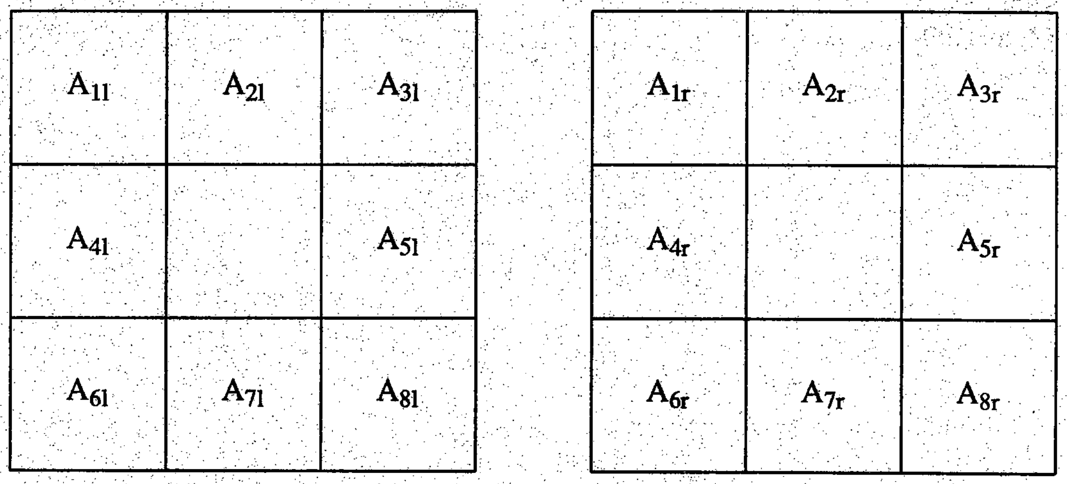
$$S = \sum_{i=0}^{7} \text{NEG} (A_{il} \text{ XOR } A_{ir}), \tag{10}$$

where XOR denotes the logical exclusive or, and NEG denotes the logical negation. If all eight elements of the 3 × 3 matrix match the value S is 8, meaning the two contours have identical orientations. If one of the eight elements differs, then S is 7, which, as should become evident from Table 1, corresponds to a maximum orientation difference of 27 degrees.* The orientation differences for a 1 bit disagreement can be found by comparing the orientations for the successive entries in the table. Again from the table, when the value of S is 6, meaning that two bits in the left and the right image patterns are in disagreement, the angular difference between the corresponding contours is at least 34 degrees. Since we did not want the contour orientations to differ by more than 30 degrees, we used a threshold of 7 on S. In other words, two zero-crossing orientations were considered be the same if there was only a one bit disagreement between their corresponding neighborhood binary masks.

Our binary neighborhood comparison procedure has one advantage over methods that are more explicit in their usage of local orientations. Consider the following two binary patterns in the neighborhood of a zero-crossing.

```
1 1 1        0 0 1
0 1 1        0 1 1
1 1 1        0 0 1
```

*The orientation entries in the table were generated by the application of 3x3 Sobel operators to the bit patterns shown in the second and the fourth columns. Note that the table itself is not used during the matching of the zero-crossings. It is shown here only for the purpose of justifying the threshold used on S.



(a) Left image. **(b) Right image.**

Figure 6 3×3 image patches and their elements.

The Sobel operator would assign the same contour slope to the center point of both the patterns, since at the center point the first derivative in both cases is the same, only the second derivatives are different. Therefore, a matcher that uses gradient calculation based orientations would match the two zero-crossings corresponding to the patterns, although such a match would be erroneous. However, with our scheme, when two patterns are compared, the resulting value of S is 4, which is below the threshold, therefore the zero-crossings would not be matched.

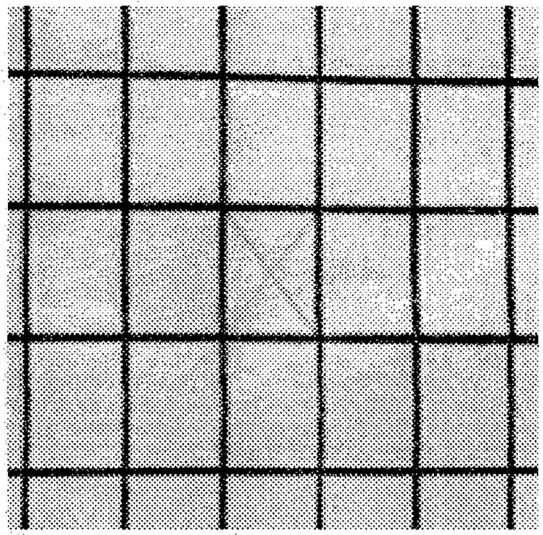
Note that there is an implied assumption here that prior to the setting up of search windows in the manner described above, the two images are already rectified. Rectification means that the epipolar lines in the two images are parallel and correspond to the rows of the matrices representing the images. More specifically, it is assumed that the correspondents of all the pixels on the i-th row of the left image are on the i-th row of the right image. In general, this will not be the case, especially when the optic axes of the two cameras are not parallel. As discussed by Horn [17], when the optic axes are convergent, the epipolar lines in each of the image planes are also convergent. For example, the epipolar lines in the left image must all radiate out from the left-image point corresponding to the camera center of the right image.

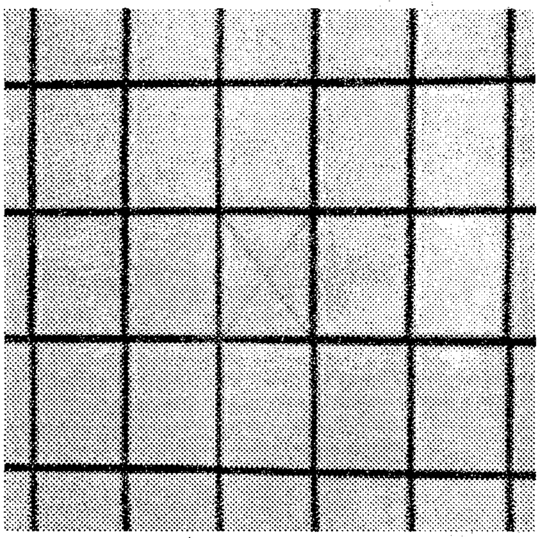
In case the reader is not already familiar, an epipolar line is defined for each image point as that line in the other image on which the corresponding point must lie. Given, say, a left-image point, we can argue that the object point must lie on the ray passing through the image point and the left camera center; if we project this ray onto the right image, we obtain the epipolar line for the left-image point in question. By analyzing the geometry associated with epipolar lines, it can be shown that when the optic axes are parallel, the epipolar lines must also become parallel. We demonstrate this with images of an object consisting of lattice pattern, whose horizontal lines and vertical line are perpendicular, drawn on a sheet of paper. Figures 7 (a) and (b) show a stereo pair of images obtained when the camera axes are convergent with an angle of 45 °. From the slant of the lines in the images, it is clear that in this case it would be inaccurate to use one-dimensional search windows for solving the correspondence problem. The process of rectification consists of transforming each of the images in such a manner that the transformed stereo pair corresponds to the case of parallel optic axes. In general, such transformations can be complicated. In our research, we have circumvented the difficulties of rectification by using keeping the camera optic axes nearly parallel, usually within 10 °, and objects at a relatively large distance from the camera baseline. The camera baseline was typically 20 inches long and the object to baseline distance typically 110 inches. Figures 7 (c) and (d) show the stereo images of the 2-D object taken under such conditions. Since we do not use any rectification explicitly, we modify the search windows somewhat and include the $\pm w_{2D}$ intervals in the row above and the row below.

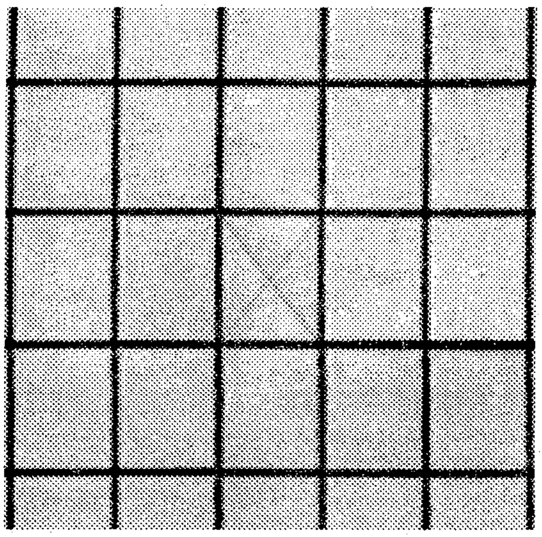
Table 1 3 × 3 binary image patches and their associated orientations.

Orientation (deg)	3 × 3 patch (binary image patch)	Orientation (deg)	3 × 3 patch (binary image patch)
0	0 0 0 * 1 0 1 * 0 0 0 * 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1	180	1 1 1 * 1 1 1 * 1 1 1 * 1 1 1 0 1 0 1 1 1 0 0 0 0 0 0 1 0 1
27	0 0 1 * 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1	207	1 1 1 1 1 1 * 1 1 0 1 1 1 0 0 0 1 0 0
45	0 0 1 0 1 1 1 1 1	225	1 1 1 1 1 0 1 0 0
63	0 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 1 1	243	1 1 0 1 1 1 1 1 0 1 1 0 1 0 0 1 1 0
90	1 1 1 0 1 1 0 0 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1 1 0 0 1	270	1 0 0 1 1 0 1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1 1 0 1 1 1
117	1 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1	297	1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1
135	1 1 1 0 1 1 0 0 1	315	1 0 0 1 1 0 1 1 1
153	1 1 1 * 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0	333	0 0 0 1 0 0 * 0 1 1 1 1 1 1 1 1 1 1 1

Note : 3 × 3 patches with * mark are not used for matching, because their parity is neutral (neither positive nor negative).


(a) Left image (angle = 45 °).


(b) Right image (angle = 45 °).


(c) Left image (angle = 10 °).

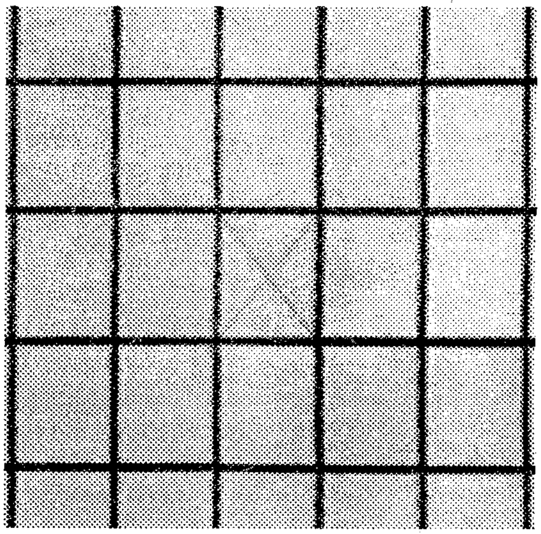

(d) Right image (angle = 10 °).

Figure 7 Perspective disortion in binocular stereopsis.

The reader should also note that a further correction is usually necessary before the zero-crossings can be matched even with the window modification we just described. In general, the plane containing the two optic axes will not intersect the image planes along lines parallel to the scan lines, assuming of course that the two axes are very nearly parallel and ‘‘containable" in a plane. When this happens, the epipolar lines although parallel will not lie along the camera scan lines. To get around this difficulty, the two images must first be row-registered. Although one can visualize high-precision experimental set-ups where this would not be a problem, in our experiments where the cameras are mounted on ordinary tripods, we must do the row-registration manually before matching. Our camera images are 512×480. From the left image, we first extract a 256×256 subimage. Then, from the right image we extract a similar sized subimage that appears to be best row-registered with the left subimage.

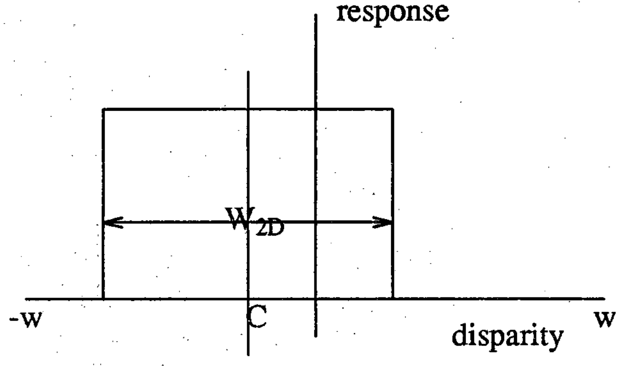
2.2.4. Disambiguation

Since, statistically, there will be many cases where there are two or more candidate zero-crossings inside a search window of size $\pm w_{2D}$, one has to employ some disambiguation strategy to select one of the zero-crossings. Following Marr and Poggio [2], the notion of *pulling effect* is used by us for disambiguation. The pulling effect is intended to lend a certain measure of continuity to the calculated disparities values; meaning that since disparity variations can not be chaotic, if there is a choice one must use that value which is most consistent with the disparities in the neighborhood. Grimson [9] implemented the pulling effect in the following manner: In the k th channel, let c_{k_i}, i=1,2,...,n, be the candidate zero-crossings in a right image search window set up for a given left image zero-crossing. Let d_{k_i} be the disparity of the i th candidate. We will accept that d_{k_i} for which we can find a coarser channel zero-crossing within a specified neighborhood of the left image zero-crossing, the disparity d_{k-1} that is associated with the coarser channel zero-crossing being such that

$$\left| d_{k_i} - d_{k-1} \right| \leq \frac{w_k}{2} \tag{11}$$

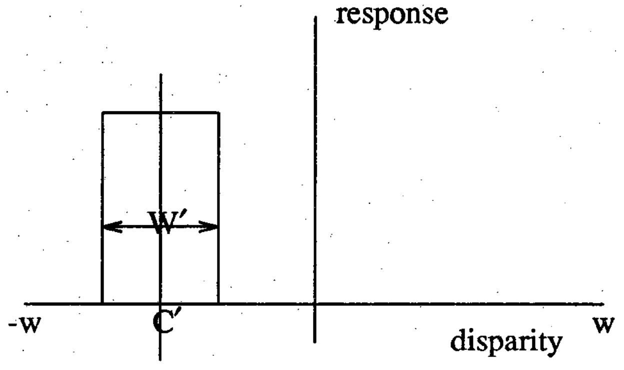
where w_k is the w_{2D} for the k-th channel. This pulling effect is illustrated in Figure 8 (a) where C corresponds to d_{k-1}.

Our implementation is quite different from that of Grimson, since we do not at all use the coarser channel disparities for the pulling effect. Instead, we insist that a candidate zero-crossing be selected such that



C is determined by the coarser channel range data

(a) Grimson's Model



C' is determined by the pulling effect $W' \ll W_{2D}$

(b) Our Model

Figure 8 Disambiguation models.

$$\left| d_{k_i} - \sum_{\text{disamb-neighborhood}} d_{k_j} \right| \leq \frac{w_k}{4} \tag{12}$$

where *disamb-neighborhood* stands for neighborhood around the left-image zero-crossing in question. In most of our programs, the size of this neighborhood is 20x20. This approach to the pulling effect is shown in Figure 8 (b) where C' is the average shown in the above equation.

The size of our pulling-effect window was set empirically. We examined a large number of stereo pairs and concluded that a larger window, say of size $\pm w_{2D}$, allowed more than one candidate match to be accepted, which violated the entire purpose of disambiguation. And, of course, a smaller window, say of size $\pm\frac{w_{2D}}{8}$ rejected too often all the candidates.

2.3. Problems with the MPG Approach

The disambiguation aspect of the MPG approach is not as defensible as the rest of the formalism. Of course, at a theoretical plane, the philosophy behind disambiguation appears to be sound -- as it seeks to enforce continuity and uniqueness on the computed disparities -- however, there is a certain looseness in the implementation of the idea itself. For example, in our own implementation, the size of the window one uses for *disamb-neighborhood* depends upon how rapidly varying the depth values are; however, the nature of this dependence is poorly understood at this time. Our own selection for the size is made by trial and error. Clearly, a window designed for a class of scenes may not work well for another class. We maintain that the same is true for other implementations.

While the above difficulty is more in the nature of how a particular aspect of a theory should be implemented, we will now show, with the help of a simple example, a shortcoming of the MPG formalism that goes to the heart of the theory. We will show that a computational theory of depth perception must not only carry out bottom-up processing of the sensory information, as is done in the MPG approach, but it must also invoke top-down expectation-driven procedures.

For the example, assume that the object surface is made up of three panels, as shown in Figure 9. Also assume that the camera positions are such that the panel 2 is not visible in the left image. If we assume that the surface of the object is randomly textured, it is highly probable that the zero-crossings in the panel 2 portion in the right image will match with some zero-crossings from either panel 1 or 3 in the left image.

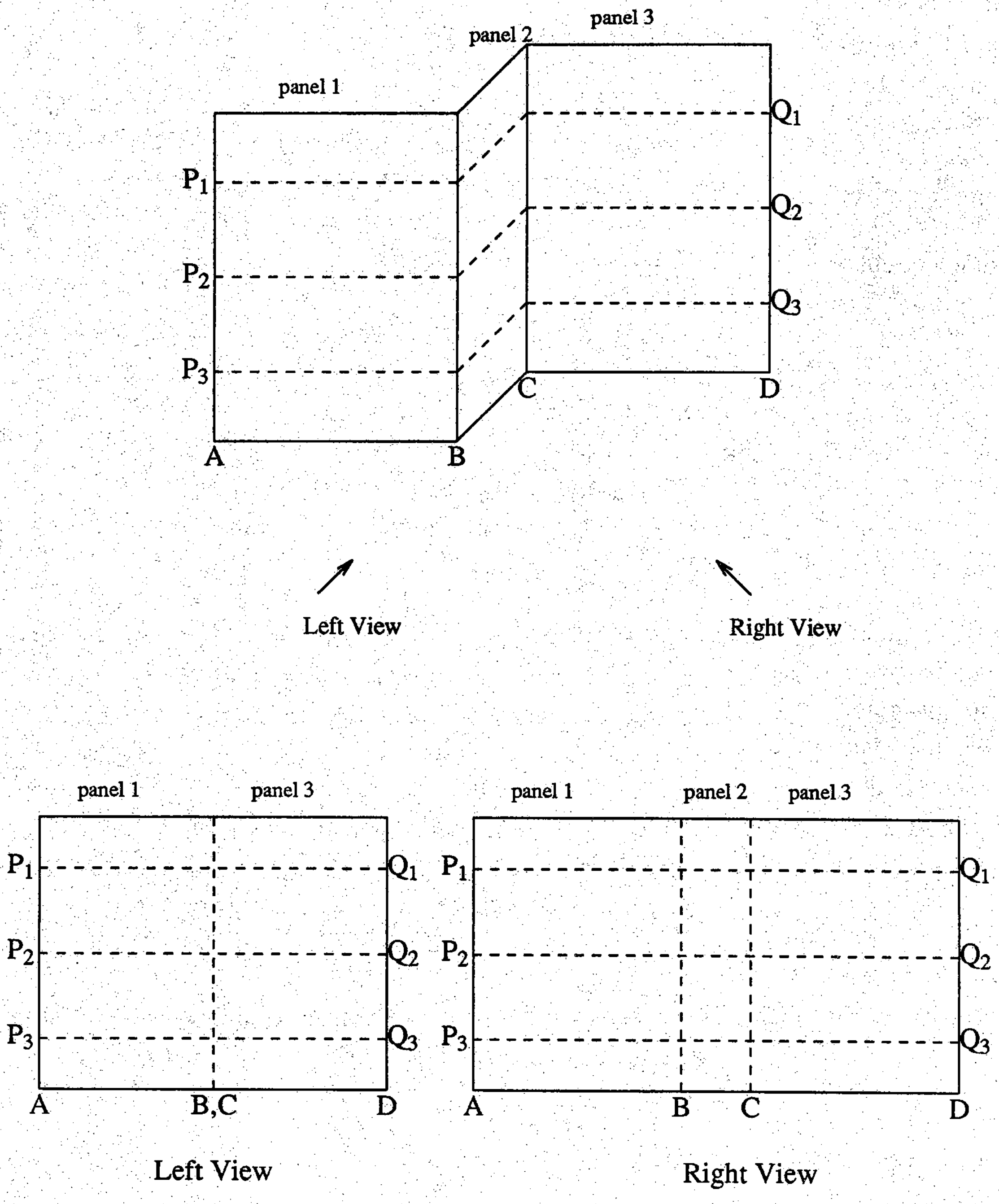
Figure 10 illustrates calculated disparity values along the line PQ shown in the previous figure. The correct depth values are also shown for comparison.

To the human visual system, the object surface in Figure 9 presents no difficulties whatsoever when it comes to depth perception, even in the presence of occlusions, as shown there. We believe the human visual system uses higher level cognitive processing which invokes object level knowledge to place additional constraints on disparity values. We also believe that, in most cases, monocular processing is sufficient to generate these object level expectations that are then used in the process of binocular fusion.

It appears plausible that in the human visual system object-level knowledge for binocular stereopsis is triggered by strong high-level features in a scene. So, clearly, any attempt at generating object-level constraints on acceptable disparities must start with the detection of high-level features. In our work so far, the only high-level features we have used for this triggering process are straight edges that are strong and long. We refer to strong and long straight edges as dominant features.

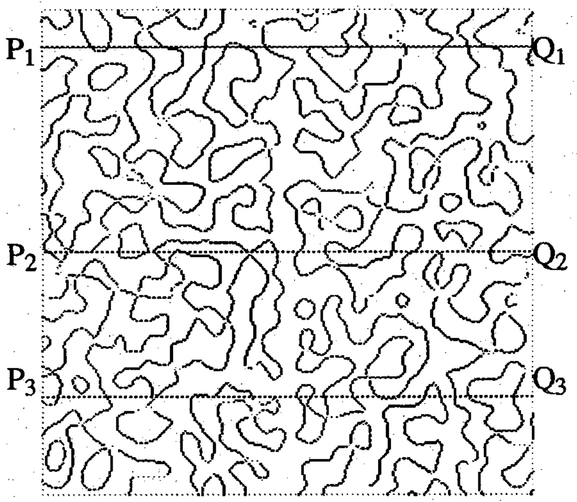
Therefore, if a computational theory of depth perception is not to suffer from the kind of shortcoming exemplified by the three-panel example, the dominant features must first be detected and matched. The resulting structures in 3-D space must then trigger hypotheses about the orientations of surfaces in the vicinity of the dominant features. Finally, these hypotheses must generate constraints for acceptable disparities near the dominant features. Of course, if a sufficient fraction of the available zero-crossings can not be matched under a particular hypothesis, that hypothesis must be rejected in favor of others. Clearly, of no hypotheses do justice to the zero-crossings in the vicinity of a structure generated by matching a pair of dominant features, then that structure should be discarded and other possibilities investigated for how the dominant features should be paired up. THIS IS THE ESSENCE OF OUR RULE-BASED APPROACH.

In the next chapter, we will briefly review the work done so far in stereo matching by invoking higher-level constraints. In the subsequent chapter, we will then discuss how some of these methods were modified for incorporation in our system.

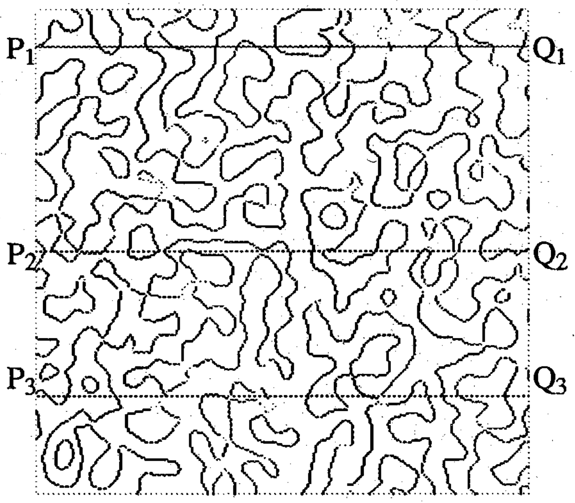


The top portion illustrates an object surface and the two viewpoints. The lower potion shows the left and the right image taken from the two viewpoints. Notice that panel 2 is missing from the left viewpoint.

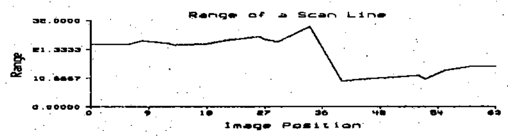
Figure 9 A three panel model of erroneous correspondence.



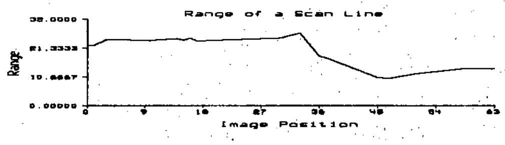
(a) Left contours.



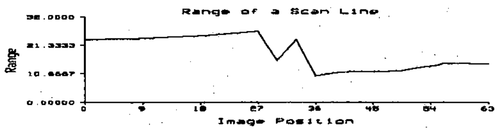
(b) Right contours.



(c) Range on raster P_1–Q_1.



(d) Range on raster P_2–Q_2.



(e) Range on raster P_3–Q_3.

Figure 10 Disparity maps on three scan lines in the sample scene.

CHAPTER III REVIEW OF PROCEDURES FOR STEREO MATCHING UNDER HIGH LEVEL CONSTRAINTS

While the previous chapter discussed the MPG formalism and its shortcomings, this chapter will be a review of the stereo matching techniques that stand "at the other end." By the other end we mean the techniques that accomplish binocular fusion solely by high level, including object level, constraints on disparities.

Matching Using Geometrical Constraints

In some of the existing approaches, geometrical constraints are explicitly used during the process of matching zero-crossings. For example, in the work done by Eastman and Waxman [6], a disparity functional is computed that estimates the coefficients of a local planar patch corresponding to a tentative match of a few adjacent zero-crossings in the left and right images. Matches may then be accepted or rejected based upon the magnitude of the disagreements between the disparity functional and the computed disparity. These authors have made the assumption that their scenes are continuous. In one of their methods, which they call the neighborhood-based algorithm, matching proceeds from the coarsest channel to the finest channel. Initially, for the coarsest channel, it is assumed that the scene is at a constant depth -- which can be the average depth of the scene. Subsequently, the disparities computed at a given scale determine the location in the next finer channel of the windows to be used for searching for correspondents of zero-crossings.

The procedure proposed by Hoff and Ahuja [7] for enforcing geometrical constraints consists of two phases : In the first phase, as in the previous method, planar patches are fitted to tentative matches between zero-crossings, the coefficients of a planar patch being determined by the use of the Hough transform. Subsequently, in the second phase, the neighboring planar patches are clustered into quadratic surfaces by a least squares method. This then allows enforcement of geometrical constraints at the quadratic level. The control-flow for the propagation of matching information from the coarsest to the finest channels is the same as in the method of Waxman and Eastman -- which is basically the same as in Grimson's implementations of the Marr-Poggio paradigm. First, in the coarsest channel, it is assumed that the scene can be represented by a constant (planar) surface at the average depth in the scene. And, then, the disparities computed are used to create a smooth curved surface for predicting the search windows for the next channel.

The notion of using geometrical constraints is, we believe, very useful, particularly for industrial vision applications. Such constraints form an important component of the rule-based approach that will be presented here.

The Constraint on the Ordering of Features

Images of scenes made of opaque object must observe an important constraint : monotonicity of rendition of object points. By monotonicity of rendition, we mean the following : Suppose we mark all the object surface points with a set of marks, no two of which are identical. On any line running through an image of the scene, the order of appearance of the marks must correspond to the order in the scene. In other words, there cannot be position reversals in, say, the left-right ordering of the marks. This constraint makes the stereo correspondence problem ideally amenable to solution by dynamic programming. Of course, since in practice it is not possible to mark up a scene, for using dynamic programming one has to first select a set of distinguished points from each image; how such points are selected sets apart the various implementations of this scheme. Another distinguishing aspects of the various implementations is the distance of function used for measuring the closeness of correspondence achieved between the gray levels along the epipolar lines in the two images.

To mention a few of the implementations that use the dynamic programming approach, Baker and Binford [18] have used for the distance-function features based on edge angles, gray levels on the two sides of an edge, relative disparities, (measured during a reduced resolution phase) and interval compression that is implied by the correspondence. The distance function used by Ohta and Kanade [19] uses the similarity of epipolar line intervals between successive distinguished points. In another study, by Lloyd, Haddow and Boyce [20], the distance function is based upon distances between distinguished points, edge angles and average gray levels. The implementation of this last approach is along the lines of relaxation labeling. Note that in all three approaches, the distinguished points can, for example, be edge segments.

Looser Ordering Constraint

In the dynamic programming approach, the ordering of image points declared distinguished must be strictly maintained in any matching of the two images along an epipolar line. If we loosen the constraint because of, say, the difficulty with the rectification process, the problem of matching image elements, such as edge segments, along epipolar lines can be cast as a graph search problem, as was done by Herman and Kanade [21]. Matching of edge segments along epipolar lines has also been carried out by Medioni and Nevatia [5] using a relaxation type approach.

Some Other Approaches

We would now like to allude briefly to the fact that methods do exist for improving the overall accuracy of the calculated depth information that do not resort to the invocation of higher level constraint knowledge. For example, Nevatia [22] has shown that if a progression of closely spaced views are fused together, it is possible to reduce mismatches in stereo correspondence without sacrificing accuracy in depth calculation. Moravec [23] has also proposed a multi-view stereo-system, with similar results. Tsai [24] has used a statistical approach to combine eight views of a scene using joint moments and window variances.

CHAPTER IV MATCHING METHODS INCLUDED IN THE RULE-BASED PROGRAM

We will now describe in greater detail the matching schemes that can be invoked by the rule-based system. At any given time, a 64×64 control matrix is used to store information on which matching scheme to invoke where in an image, which is usually of size 256×256. Each element of the control matrix, depending upon its integer value, indicates selection of a particular matching strategy. Later in this paper we will discuss how the values of this control matrix are initialized, and, subsequently, how these values are altered dynamically.

4.1. Dominant Feature Matching

In line with the discussion in the previous section, an important matching strategy consists of extraction and fusion of dominant features from the two images. At this time, the only type of dominant feature the system is capable of handling is the straight-line type. We will now describe the procedures used for first extracting straight lines and then fusing them.

4.1.1. Extraction of Straight Line Features

In many previous studies, binocular fusion has been carried out on straight line features [18, 21, 5, 4]. Baker and Binford [18] extract edges from the stereo pair and represent each edge as a concatenation of piece-wise straight segments, associated with each segment being a set of attributes like the orientation, side-intensities, etc. They then scan each image row by row, and match the segments on each row of the left image with the segments in th corresponding row of the right image by using the Viterbi algorithm. The procedure used by Herman and Kanade [21] is very similar except for the fact that matching is accomplished on L-junctions formed by straight edge segments. In the work reported by Medioni and Nevatia [5], edges are extracted by the Nevatia and Babu algorithm [25] and matched by a relaxation based procedure which seeks to minimize what they call as differential disparity. In the Nevatia and Babu edge extraction procedure, edge points are first detected by using window operators, neighboring edge points are then connected to form continuous contours; these contours are then fitted with an iterative straight line fitting algorithm which in the first iteration consists of joining the extremal points on the contour with a straight edge, and, later, if the this straight edge is too poor a fit to the contour, selecting a middle contour point for the purpose of fitting straight edges to the two segments thus formed, and so on. In the work reported by Ayache and Faverjon [4], recursive method is taken in order to approximate a contour with the set of linear segment similar to the one used by Nevatia and Babu.

For our work, we have preferred to use a method that is a variation on a scheme first proposed by Freeman [26]. In our implementation, thresholded outputs of edge operators are represented by chain-codes and tested for straightness. The advantage of this approach is that it can be made insensitive to small local and structural deviations from straightness. The competing approaches, some of which were alluded to in the previous paragraph, would consist of fitting analytically described arcs to edge points. However, being analytic, such approaches are more suitable when we want a straight line extraction procedure to be insensitive to distortions that are somewhat more global in nature. In any case, our choice should be construed less as a categorical judgement on which of the two types of approaches is ultimately suitable for straight line extraction and more as a matter of personal preference.

Freeman has suggested three criteria which must be satisfied by the chain code of a digital straight line [27]:

(1) At most two slopes occur in the chain, and if there are two, they differ by 45° exactly.
(2) At least one of the two slopes occurs in runs of length 1.
(3) The slope occurs in runs of at most two lengths (except possibly at the ends of the arc, where the runs may truncate) and, if there are two lengths, they must differ by 1.

These conditions, postulated for digital renditions of ideal straight lines, are not entirely satisfactory for straight lines extracted from images of objects and scenes. The "real world" straight lines often exhibit small deviations from straightness; a line extraction algorithm must be forgiving of those. For example, a vertical line segment from an image might lead to the following chain code for its representation:

666666676666666665666666

As is evident from Figure 11, this line, even though it violates the three Freeman criteria, would be called straight by most observers. Clearly, we need to "loosen" the criteria in order to be able to process actual images. This we do in the following manner.

In the first Freeman criterion, not more than two code numbers are allowed in the chain code of a straight line; in our algorithm, up to three different codes are allowed. However, we require that when three different codes are present in a chain, the center code occur most frequently and that the values in the non-central bins not exceed a threshold T1, whose value is dependent on the length of the digital arc. By experimentation with the type of imagery that is of interest to us, we have found that an appropriate value for T1 is 3 when the length of the digital arc is 41. The reason for why we want our straight line segments to be 41 pixels long will be explained toward the end of this subsection. Figure 12 (a) shows a digital arc whose length is 41 but whose chain code histogram, as shown in Figure 12 (b), is such that T1 is 4. Clearly, most observers would declare this arc as not straight.

The next Freeman criterion says that one of codes can only occur in unit lengths; we permit the maximum length of the code number which corresponds to the shortest runs in the code sequence to be T2, which is also dependent on the arc length and is 2 in the current model. The number 2 was again arrived at through experimentation. Figure 13 (a) shows a digital arc of length 41 whose histogram, as displayed in Figure 13 (b), is such that T2 is 3; this arc will again not be declared straight by most observers. We do not at all use the third Freeman criterion. In the rest of this section, we will first complete a description of our algorithm and then justify the underlying rationale with the help of a few examples.

Our criteria imply that the decision about the straightness of a contour segment (meaning an arc) can be made entirely by examining its chain code histogram. For each contour segment, a chain-code histogram is constructed such that the bar corresponding to each code number -- the code numbers go from 0 to 7 -- represents the frequency of the appearance of the code number.

From the chain code histogram, we compute two numbers, A and N, in the following manner. Let C(i) be the frequency of the chain code number i. Then

$$A = \left\{ i \,|\, C(i) \right\}, \tag{13}$$

and

$$N = |A|,$$

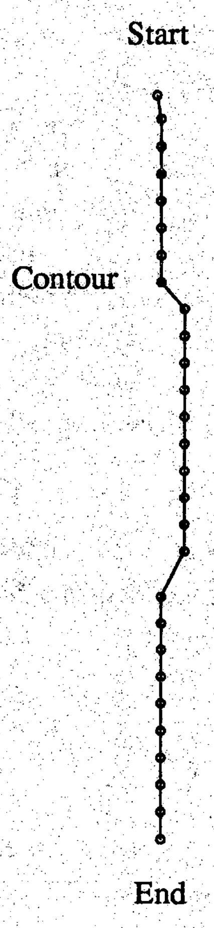
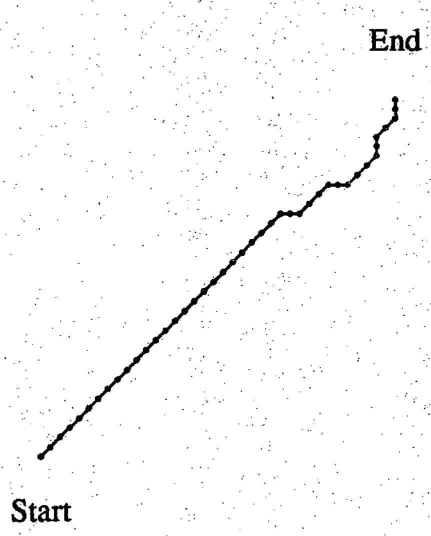
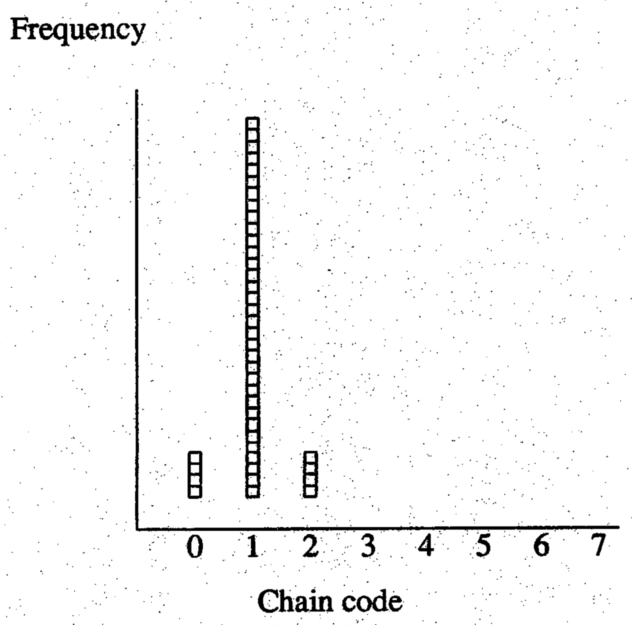


Figure 11 Almost straight contour segment which violates Freeman's criteria.

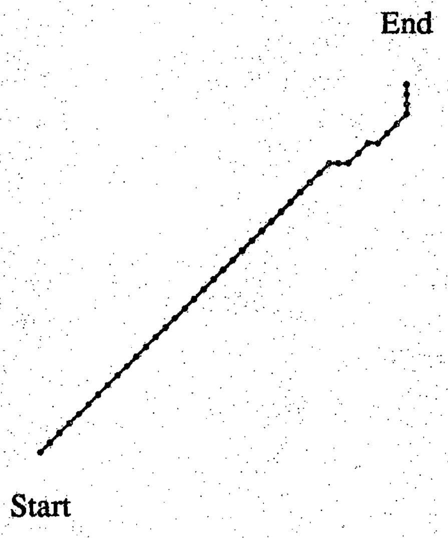


(a) A contour segment.

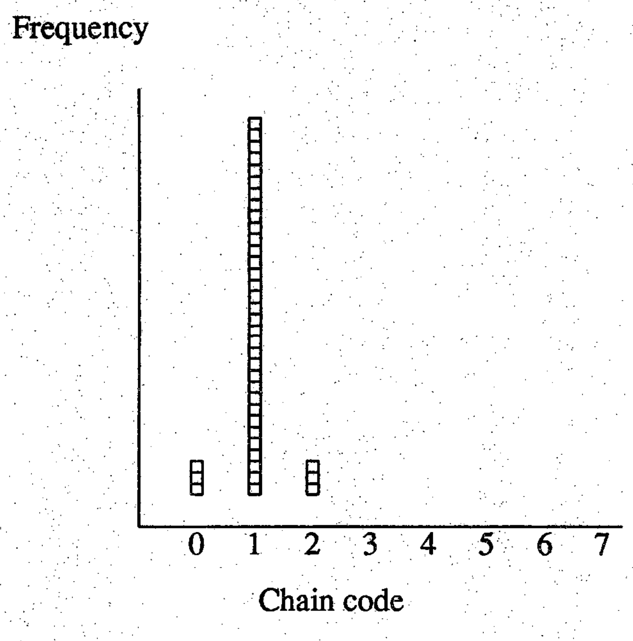


(b) Chain code histogram.

Figure 12 A digital arc and chain code histogram (violation of threshold T1).



(a) A contour segment.



(b) Chain code histogram.

Figure 13 A digital arc and chain code histogram (violation of threshold T2).

where the notation |A| denotes the cardinality of A. In terms of A and N, we can make the following decisions about the straightness property:

(1) If the histogram has more than four bars (N>4), this line is not straight by any means. This line segment has at least four different orientations.

(2) If the histogram has a single bar (N = 1), the line is purely straight, with the orientation of 0°, 45°, 90°, or 135° with the horizontal line.

(3) If the histogram has two bars (N = 2), two cases need be considered (let the more frequently appearing code be the major code and the other code be the minor code):

    A. If the two bars are adjacent to each other, again there are two cases:

        a. If the maximum run length of the minor code is less than a user specified threshold T2, the line is declared straight.

        b. If the maximum run length of the minor code is greater than T2, the line is declared as not straight.

    B. If the two bars in the histogram are not adjacent, this arc is declared as not straight. This arc contains at least two different orientations, and the angles of these orientations differ by at least 90° difference.

(4) If the histogram has three bars (N = 3), the following two cases need be considered.

    A. If the three bars are adjacent to one another, the center bar is the largest, and the height of the closest neighboring bar is less than a user specified threshold T1, then there are the following two cases to consider:

        a. If the maximum run-length of the non-central code (meaning the value in the non-central bin in the histogram) is less than T2, the line is declared straight.

        b. If the maximum run-length of the non-central code is greater than the threshold T2, the line is declared as not straight.

    B. On the three bars in the histogram if no two are adjacent, then the arc is declared as not straight. This arc has at least two different orientations, with angles differing by at least 90 °.

The block diagram of Figure 14 is a depiction of the flow of control in this algorithm.

We would now like to say a few words in support of our modifications to the Freeman criterion. Figure 15 shows a synthetic image consisting of a set of digital arcs; these have been numbered from 1 through 15. The arcs 1 through 5 satisfy all three Freeman criteria. However, the arcs 6 through 10 do not satisfy the Freeman criteria, but they do satisfy our criteria and would be considered as straight by most human observers. For example, for arcs 8 and 9 the value of N, the number of non-zero bins in the chain-code histograms, is 3. For that reason, these arcs would not be accepted as straight by the Freeman criteria. Although the chain codes for the arcs 6

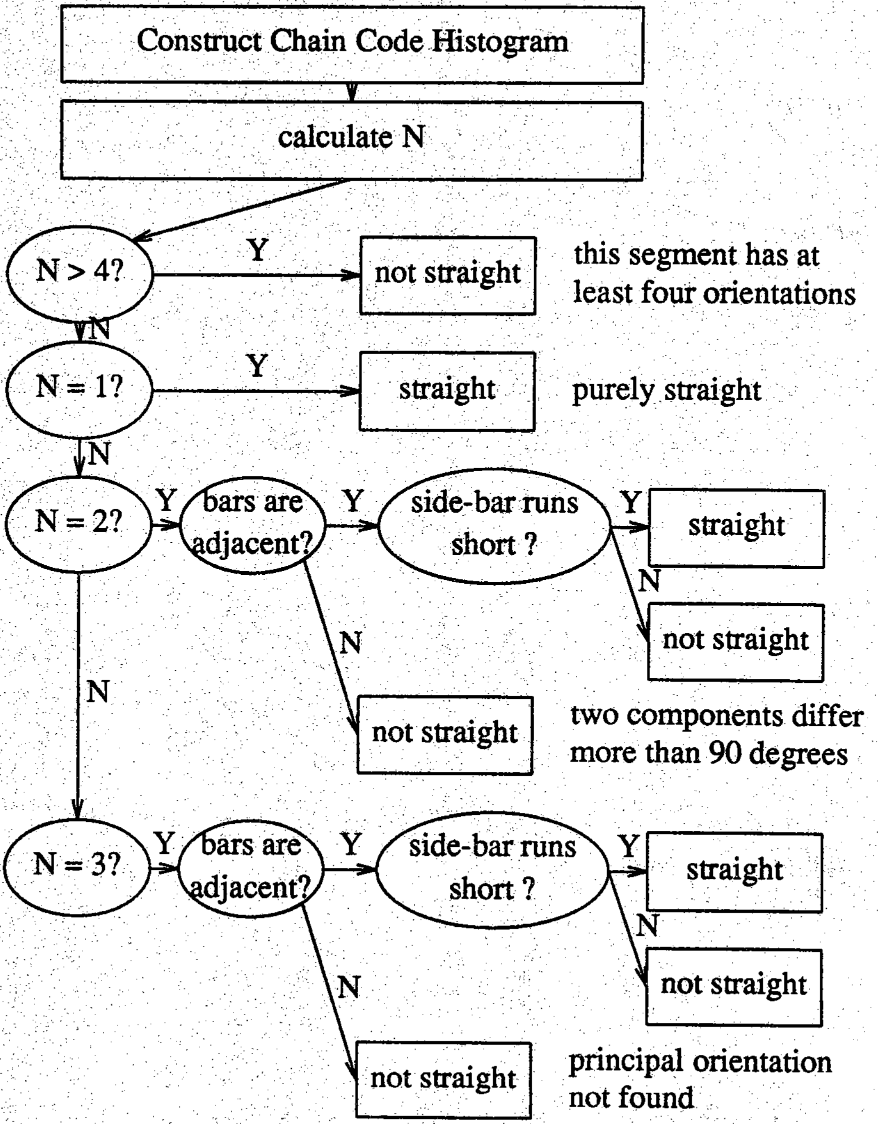
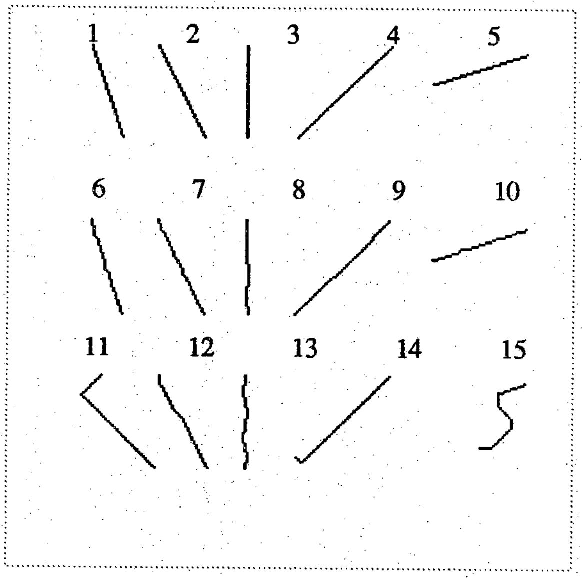
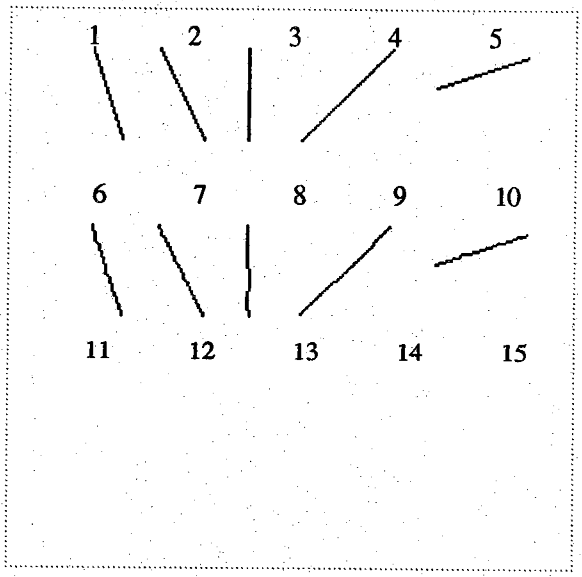


Figure 14 The algorithm for straight line extraction.



(a) Before straight line extraction.



(b) After straight line extraction.

Figure 15 Digital straight arc extraction.

and 7 contain only two slopes, which is one of the requirements of the Freeman criteria, both the slopes occur in runs of lengths more than one -- which is a violation of the Freeman criteria. Similarly, arc 10 is accepted as straight by our method because we do not use Freeman's third criterion.

The reader is probably wondering if some non-straight looking arcs would be accepted by our method. Note that the arcs 11 through 15 would not be considered straight by most observers and are also rejected by our criteria. While arc 11 is rejected because the two slopes differ by more than 90 degrees, arc 12 is not accepted because the run-length of the minor code is greater than the threshold used for declaring an arc short, which in these cases was set to 3. The minor codes of arc 13 appear more frequently than the shortness threshold value of 3. In the chain-code histograms of arc 14, the highest frequency chain-code is not at the center of the three consecutive non-zero frequency codes. The number of non-zero bins in the chain-code histogram for arc 15 is 4, which exceeds the maximum limit of 3.

The application of the straight line extraction processing to Figure 15 (a) results in Figure 15 (b). The chain-code histogram of each contour discussed here is summarized in Table 2.

The choice of the chain code length L in the straight line extraction algorithm is currently set to 41. The main determinant of the what L should be set to is the fact that longer an arc that meets our straightness criteria the straighter it looks to a human observer. Although, this would imply that L should be made arbitrarily large, in practice if L is too large, very few straight line segments would be extracted from an image, if any at all. Through experimentation we determined that the choice of 41 for L seemed to yield most of the straight edges in the scenes of interest to us. We will now establish the point that if an arc satisfies our straightness criterion, longer L implies ''greater" straightness.

Suppose there is a digital arc A of length L whose chain code is given by C(0), C(1), ... , C(L-1) and which has been declared to be straight by our algorithm. The histogram of this arc then has only three nonempty bins; let the codes corresponding to these three consecutive bins be X, M and Y. Let P_0, P_1, P_2, ... P_L, be the points on the digital arc as shown in Figure 16. We will now introduce a parameter which measures the straightness of the arc; this parameter will be the radius R of the smallest circumscribing circle of the arc. (Larger the value of R for a given arc, the straighter the arc must be.) A circumscribing circle is defined as the circle such that at least three points, the starting point P_0, the end point P_L, and an intermediate point like, say, P_2, are on it and any arc segment does not intersect the circle. Although, some digital arcs, usually they have to look very crooked, may not possess a circumscribing circle, in practice such arcs will not be accepted by our straightness criteria and can be ignored for this discussion -- in particular the two thresholds, T1=3 and T2=2, the limit of 3 on the frequencies of X and Y, and the maximum run length of 2 for both X and Y

Table 2 Chain code histograms of computer generated digital arcs.

Arc #	Chain-Code Histogram								N	P-Code	Straight?	
	0	1	2	3	4	5	6	7			F	L
1	0	0	0	0	0	0	28	13	2	6	yes	yes
2	0	0	0	0	0	0	21	20	2	6	yes	yes
3	0	0	0	0	0	0	41	0	1	6	yes	yes
4	0	0	0	0	0	41	0	0	1	5	yes	yes
5	0	0	0	0	27	14	0	0	2	4	yes	yes
6	0	0	0	0	0	0	28	13	2	6	no	yes
7	0	0	0	0	0	0	21	20	2	6	no	yes
8	0	0	0	0	0	1	38	2	3	6	no	yes
9	0	0	0	0	1	39	1	0	3	5	no	yes
10	0	0	0	0	27	14	0	0	2	4	no	yes
11	0	0	0	0	0	9	0	32	2	7	no	no
12	0	0	0	0	0	0	20	21	2	7	no	no
13	0	0	0	0	0	4	33	4	3	6	no	no
14	0	0	0	1	1	39	0	0	3	5	no	no
15	0	0	0	0	12	14	9	6	4	5	no	no

P-Code	: The code number that has the highest frequency
N	: The number of codes whose frequencies are non-zero (For example, for arc 1 only two codes have non-zero frequency. Hence, N = 2.)
F	: By Freeman's criterion
L	: By the loosened craterion

eliminate the possibility of crooked arcs being declared straight. (See the histogram shown in Figure 17.)

To establish that longer L implies larger R for arcs accepted straight by our criteria, we should first note that associated with each L will in general be different R's. For example, for the following two arcs, both with L=20 and both accepted as straight by our criteria, the values of R are equal to 12.8 and 18. The arcs are described by the chain code sequences

Digital-arc 1

77676666666665656566

Digital-arc 2

66667676676665656566

and are shown in Figure 18. The values of R were computed by using the formulas which we will now derive. For this derivation, note that for a given L we should only be interested in the minimum value of R, since the smaller the value of R, the more "non-straight" the arc is. In In general, the radius R will be a minimum for those arcs whose non-central codes, X or Y, appear at either the beginning or the end of the arc sequence. For example, for the arc shown on the left in Figure 18 the beginning segments (at the top of the arc) correspond to the code 7, which is the non-central code in the histogram for this arc. The value of R for this arc is smaller than for the arc on right whose beginning and ending segments both correspond to the code 6, the central code in the histogram. The chain-code of an arc with the smallest R for a given L will be either of the following form, or its mirror image,

$$XXMXMM........MMYYMYM........M \qquad (14)$$

In order to derive the relationship between R and L, we will now differentiate between two types of arcs, given rise to by small and large values for L. When L is large, usually greater than 13, the nature of the circumscribing circle will be such that the three points at which the circle must make contact with the arc will be the first, the third and the last, as shown in Figure 19 on the left. On the other hand, when L is small, usually less than 13, two of the three points are at the beginning and at the end as before, however the third point will now be the fifth point from one of the ends. Given this realization, the radius R of the circumscribing circle may be computed from the formula which gives us the radius R of a circumscribing circle for a triangle of sides a, b and c.

$$R = \frac{a\,b\,c}{4\,S}, \qquad (15)$$

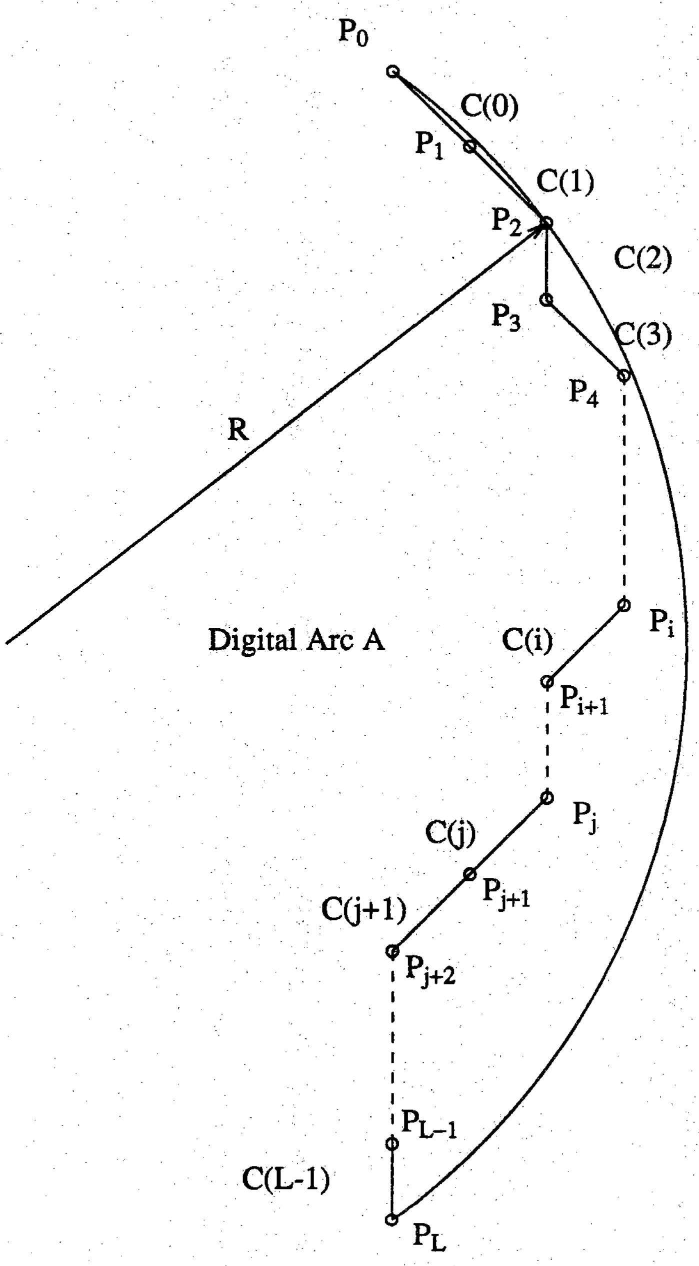


Figure 16 **Illustration of a circumscribing circle.**

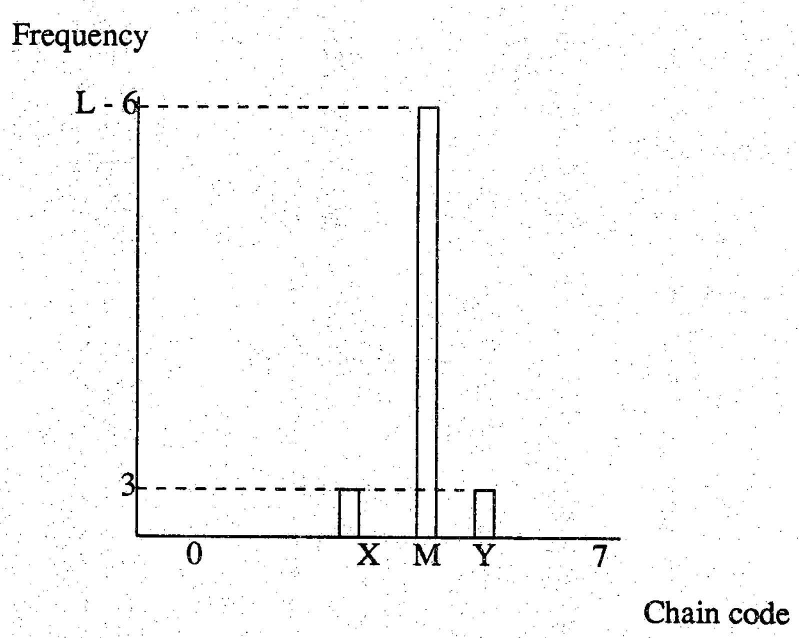


Figure 17 Chain code histogram.

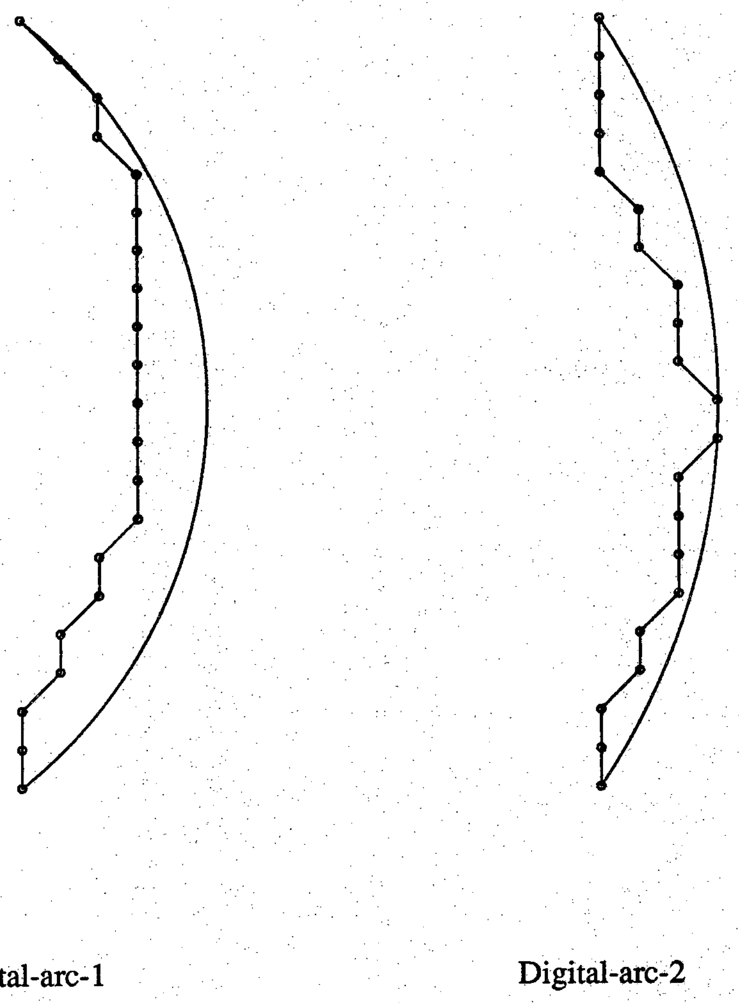


Figure 18 Two digital arcs with different code sequences.

where S is the area of the triangle. The length of the sides of the triangles, $P_0P_2P_L$ or $P_0P_24_L$, are determined by associating a unit length with an arc segment corresponding to a single code that is even, 0, 2, 4, or 6, is 1; and a length of $\sqrt{2}$ with each of other codes. Due to the fact that arc segments for even and odd codes are of different lengths, we must consider two different cases for figuring out how to compute the sides a, b, and c for the above formula. These cases correspond to whether the central code in the arc histogram is even or odd.

In Table 3, we have shown the parameters a, b, c and S for all the four possible situation, these given rise to by L being large or small, and by the central code, M, being odd or even. The values of R for each of these four cases are displayed below.

$$\text{Case–A1} \qquad R = \frac{\sqrt{2}}{2}\sqrt{(L-2)^2 + 2^2} \qquad \text{for } L > 13 \tag{16}$$

$$\text{Case–A2} \qquad R = \frac{5}{6}\sqrt{(L-4)^2 + 3^2} \qquad \text{for } L \leq 13 \tag{17}$$

$$\text{Case–B1} \qquad R = \frac{\sqrt{2}}{2}\sqrt{(L-2)^2 + L^2} \qquad \text{for } L > 5 \tag{18}$$

$$\text{Case–B2} \qquad R = \frac{\sqrt{34}}{6}\sqrt{(L-1)^2 + (L-4)^2} \qquad \text{for } L \leq 5\ . \tag{19}$$

Note that the case B-2 does not occur, because the minimum value of L is 10 as the side-lobes in our arc histograms must each contain exactly 3 elements and the central lobe must be greater than the side-lobes.

Using the above formulas to compute the smallest values of R different L, we obtain R = 5.6, 12.8, 19.8 and 26.9 for L = 10, 20, 30, and 40, respectively. For arcs whose R values are less than 20, which corresponds to L being less than around 35, would be characterized by most human observers as not being sufficiently straight. On the other hand, when R exceeds 20 -- equivalently when L exceeds 35 -- the resulting arcs are deemed to be quite straight.

We have therefore justified the assertion that with our straightness criteria, the arc lengths need to be longer than, say, 40. We have chosen L = 41; this, in most scenes of interest to us, yields a reasonable number of straight lines for dominant feature matching.

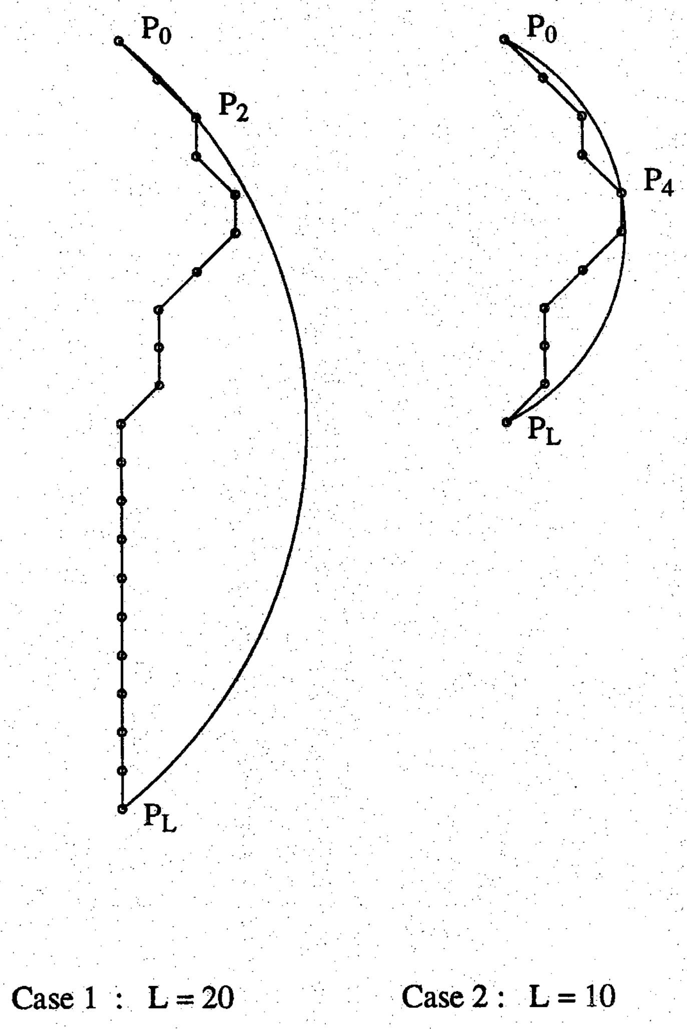


Figure 19 Two digital arcs with different length L.

Table 3 Parameters for the computation of R.

	Case A : Central code is even		Case B : Central code is odd	
	Case A-1 L is large C.C. passes through $P_0P_4P_L$	Case A-2 L is small C.C. passes through $P_0P_2P_L$	Case B-1 L is large C.C. passes through $P_0P_4P_L$	Case B-2 L is small C.C. passes through $P_0P_2P_L$
a	L	L	$L\sqrt{2}$	$L\sqrt{2}$
b	$2\sqrt{2}$	5	2	$\sqrt{17}$
c	$\sqrt{(L-2)^2+2^2}$	$\sqrt{(L-4)^2+3^2}$	$\sqrt{(L-2)^2+L^2}$	$\sqrt{(L-1)^2+(L-4)^2}$
S	L	$\frac{3}{2}L$	L	$\frac{3}{2}L$

C.C stands for circumscribing circle.

4.1.2. Detection and Deletion of Horizontal Straight Lines

Straight line segments that are parallel, or nearly so, to epipolar lines -- for our case these would be horizontal straight lines -- cannot provide reliable disparity information. In general, when we establish correspondence between two straight line segments, we are in effect establishing correspondences between the pixels lying at the intersections of the line segments with the epipolar lines. For example, in Figure 20 if we say that line AB in the left image is the correspondent of line A′B′ in the right image, we are in effect saying that the points a_1, a_2, a_3, etc, which are on the intersections of line AB with the epipolar lines, are the correspondents of the points a_1', a_2', a_3', etc, in the right image. When the line segments are nearly horizontal, there may be no well defined intersections between the segments and the epipolar lines, which can make impossible the calculation of disparities at such points. For example, if the nearly horizontal segment CD on the left is matched with the horizontal segment C′D′, it will be virtually impossible to delineate the correspondent for the pixel marked c_1. Therefore, it becomes necessary to detect and delete horizontal straight line segments. Note that we are not saying that an arc segment not contain any horizontal portions at all, only that if an arc segment is predominantly horizontal it should be deleted before the matching algorithm is applied.

As each contour segment is described by a chain code in our model, the detection of a horizontal line, or a nearly horizontal line is relatively easy. By the definition of chain codes, a horizontal portion of a digital arc is described by either the code 0 or the code 4. Note that straight line segments, each 41 elements long, are accepted as straight on the basis of their chain code histograms, as described in Section 4.1.1. All such histograms must have no more than three adjacent bins and the central bin must contain the largest count. If for a 41 element segment, the central bin corresponds to the chain codes 0 or 4, we reject such a segment from further considerations -- since such a segment would be mostly horizontal.

The digital arcs numbered 6 through 10 in Figure 21 are various examples of arc segments that were deemed as being too close to being horizontal. On the other hand, the digital arc segments 1 through 5 in Figure 21 contain many horizontal portions and yet were accepted for further consideration.

4.1.3. Binocular Fusion of Straight Line Features

We will now discuss the procedure used for the fusion of straight line features from the left and right images. For some applications, such fusion may be considered to be a special case of contour matching algorithms discussed in [28 - 32]. In most of

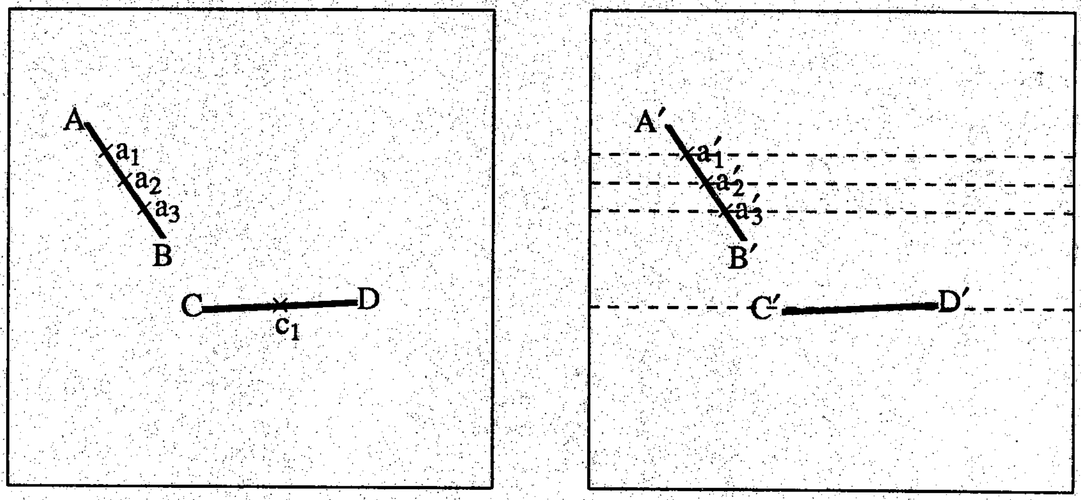
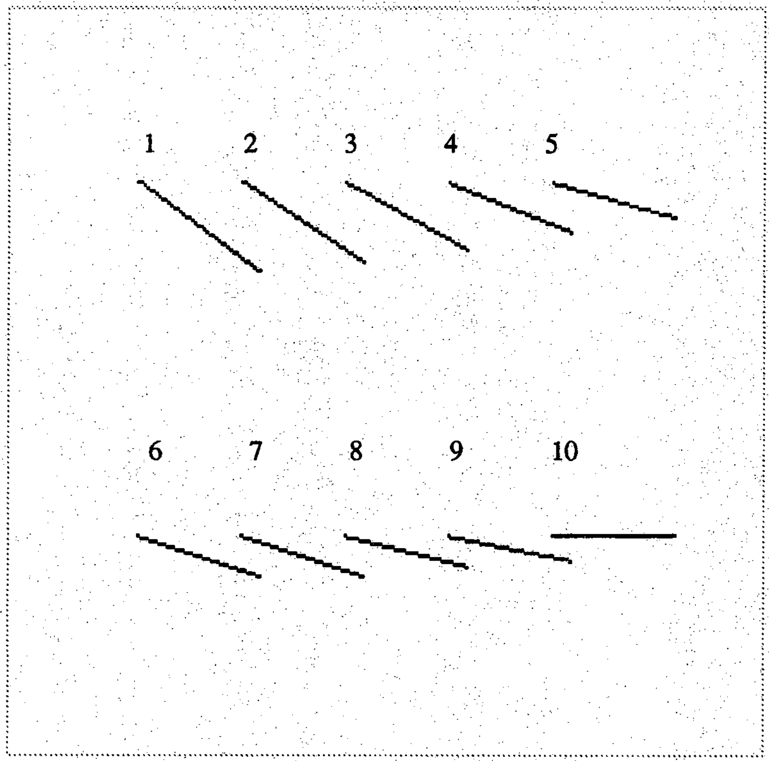


Figure 20 **A nearly horizontal edge and epipolar lines.**



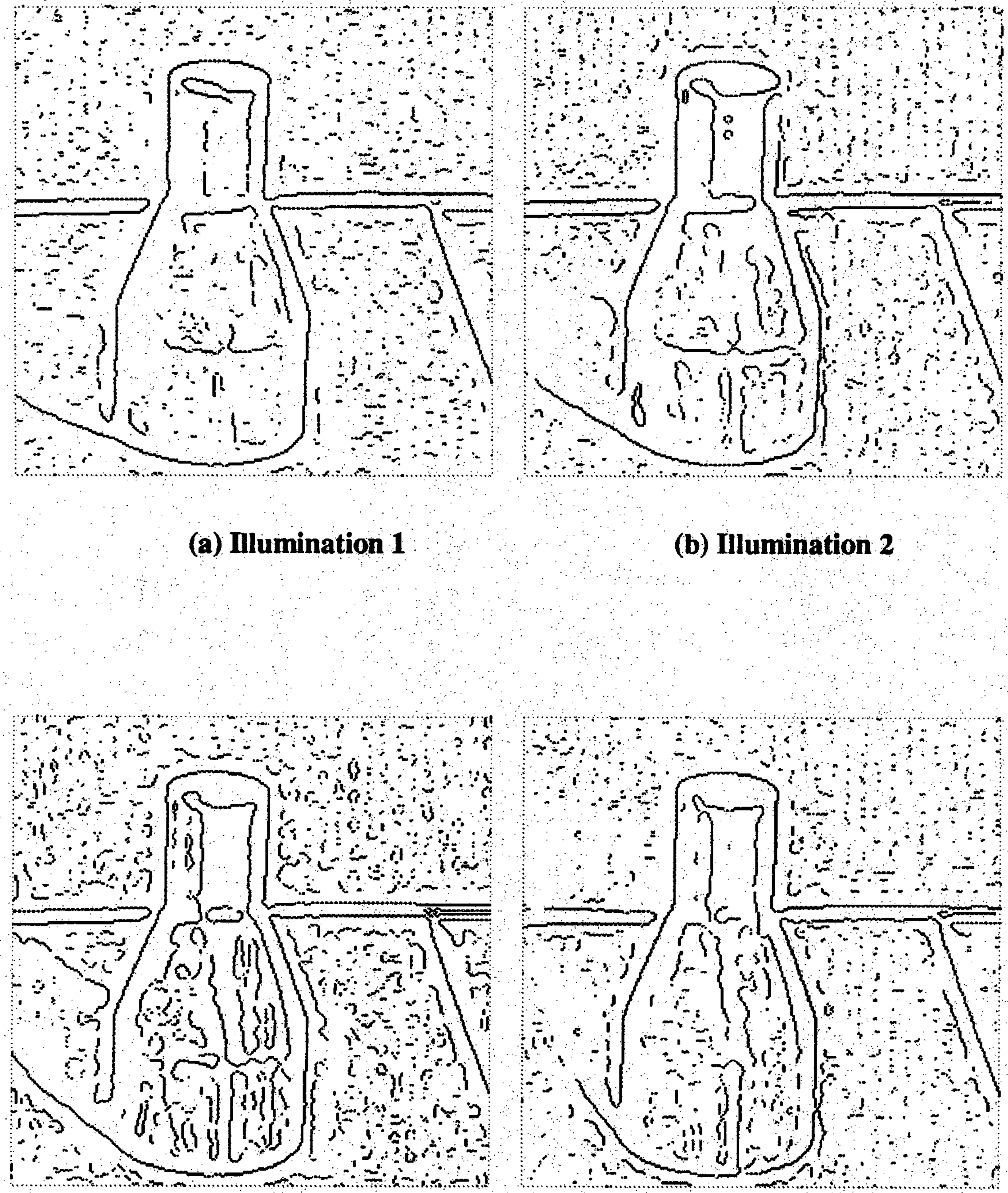
Note : The digital arcs 1 through 5 are not regarded as near-horizontal. The digital arcs 6 through 10 are regarded as near-horizontal. Therefore, arcs 6 through 10 are not supplied to the straight line matching algorithm.

Figure 21 Some examples of nearly horizontal edges.

these algorithms, a small set of parameters is used for measuring the relevant attributes of contours extracted from both images, the similarity between the corresponding contours is then established on the basis of the parameter values. For example, Pavlidis [30, 31] has used polygonal approximations for representing the contours; each contour is described by an ordered sequence of straight line arcs, there being a set of attributes associated with each arc. For each given contour in the left image, a corresponding contour is found in the right image on the basis of the similarity of these attributes. This work was followed by that of Sze et. al. [32], who used only one attribute for each straight line arc, the attribute being the slope of the arc.

Our approach is similar in spirit to that of Sze et al.; the difference lies in our use of chain codes for representing the straight line features, this being tantamount to using the finest possible polygonal approximation to a straight line arc. [The reader should note that, in keeping with the discussion in the preceding section, our straight lines are allowed to contain minor deviations from geometric straightness.] In another major departure from Sze et al., zero-crossing contours corresponding to straight line features are first represented by overlapping straight line segments to help us get around the difficulties caused by a straight line edge in the scene not appearing in its entirety in the final edge image; such a difficulty might be caused by the contrasts produced by illumination, etc. For example, Figure 22 shows four different edge images of a single scene with illumination sources placed at different angles. As the reader will notice, the same scene edge appears in different lengths in the edge images. By using overlapping segments, the straight line arcs generated by the same scene edge can be matched even when the two arcs are of unequal lengths in the two images. This notion is explained schematically in Figure 23. In Figure 23 (a), the arc is 41 pixels long, while it is only 60 pixels long in Figure 23 (b). The matching algorithm first constructs a list of all the 41 pixel long straight line segments it can discover in both images. In our example, for the left image, the list would contain 1 segment, and, for the right, 6 segments. (Table 4 shows all contour segments.) Each segment in the left image is then compared with every segment in the right image whose starting row index is within ±1 of the starting index of the left image segment. By making such comparisons, using a metric to be discussed below, the match discovers that the part PB of the left arc corresponds to the arc P'B' in the right image. As the reader can see, this approach does alleviate the problems that might otherwise be caused by a part of scene edge not showing in one of the two images.

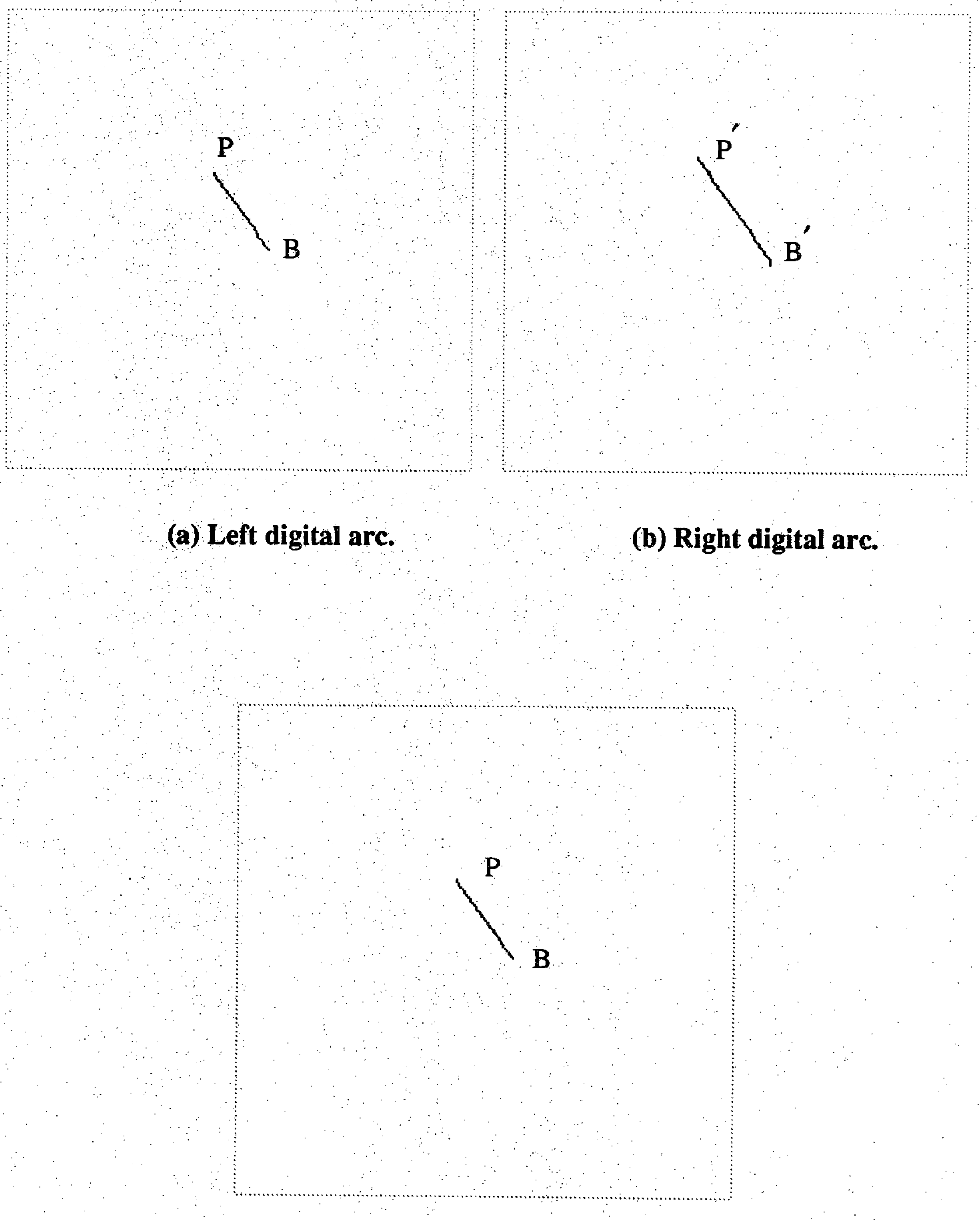
The straight line segments, each of which is 41 pixels long [see rationale in Section 4.1.1], are then given chain code representations. Subsequently, the chain code for each straight line segment in the left image is compared with the chain codes of candidate segments from the right image for the purpose of finding a match. Our method for comparing two straight line segments is based on the notion that their chain codes must be similar. The following algorithm attempts to capture this notion.



(a) Illumination 1 (b) Illumination 2

(c) Illumination 3 (d) Illumination 4

Figure 22 Zero crossings with illumination sources at different locations.



(a) Left digital arc. **(b) Right digital arc.**

(c) Matched digital arc.

Figure 23 Explanation of the overlapping of straight line extraction.

Table 4 The contour segment data structure of the example stereo pair.

Image	Contour No.	start-pix-col-index	start-pix-row-index	parity	chain code
Left [Fig. 23 (a)]	1	113	90	1	67677767....
Right [Fig. 23 (b)]	1	106	80	1	67777677....
	2	108	83	1	76777676....
	3	110	86	1	77676767....
	4	112	89	1	76767776....
	5	114	92	1	67776777....
	6	116	95	1	76777677....

Note:

"Start-pix-col-index" stands for the column index of the starting pixel of an arc.
"Start-pix-row-index" stands for the row index of the starting pixel of an arc.
"Parity" is either 1 or 0 corresponding to the positive or negative of contour parity.

In the overall algorithm for fusing straight line features, which is described in the rest of this section, first a data structure of overlapping segments in each image is created. Then candidate segments are matched by a comparison of their chain codes and a similarity score is computed. If the two segments are found matchable, the algorithm deletes from the left image data structure those segments whose starting pixels are on the segment that participated in the matching process. After all the matches have been discovered in this fashion, the last step carries out the computation of range along the points in space that give rise to matched pairs of segments. The following is a step by step description of the algorithm. The flow of the computation is shown in Figure 24.

STRAIGHT LINE MATCHING ALGORITHM

STEP-1: Each edge image is represented by a data structure which is a list of straight line segments. For example, the edge image in Figure 25 is represented by the list

{ $\text{segment}_{AB(0)}$, $\text{segment}_{AB(1)}$, $\text{segment}_{AB(2)}$, ..., $\text{segment}_{AB(N1)}$,
$\text{segment}_{CD(0)}$, $\text{segment}_{CD(1)}$, $\text{segment}_{CD(2)}$, ..., $\text{segment}_{CD(N2)}$,
$\text{segment}_{EF(0)}$, $\text{segment}_{EF(1)}$, $\text{segment}_{EF(2)}$, ..., $\text{segment}_{EF(N3)}$,
$\text{segment}_{GH(0)}$, $\text{segment}_{GH(1)}$, $\text{segment}_{GH(2)}$, ..., $\text{segment}_{GH(N4)}$ },

where $\text{segment}_{AB(i)}$ represent the i th 41-element long segment extracted from the arc AB.

STEP-2: Each straight line segment is represented by the following data structure

{ start-pixel-col-index, start-pixel-row-index, parity, chain-code },

where the first entries are self-explanatory, the third entry is +1 for
straight line segments that represent positive zero-crossings -- meaning
the gray levels are increasing perpendicular to these segments
as we go from left to right -- and -1
for segments representing negative zero-crossings.
Finally, the last item,
which is the chain code for the segment, is a list which for the purpose
of explanation will be denoted by

$$(L(1),L(2),....,L(N))$$

for the left image arcs and by

$$(R(1),R(2),....,R(N))$$

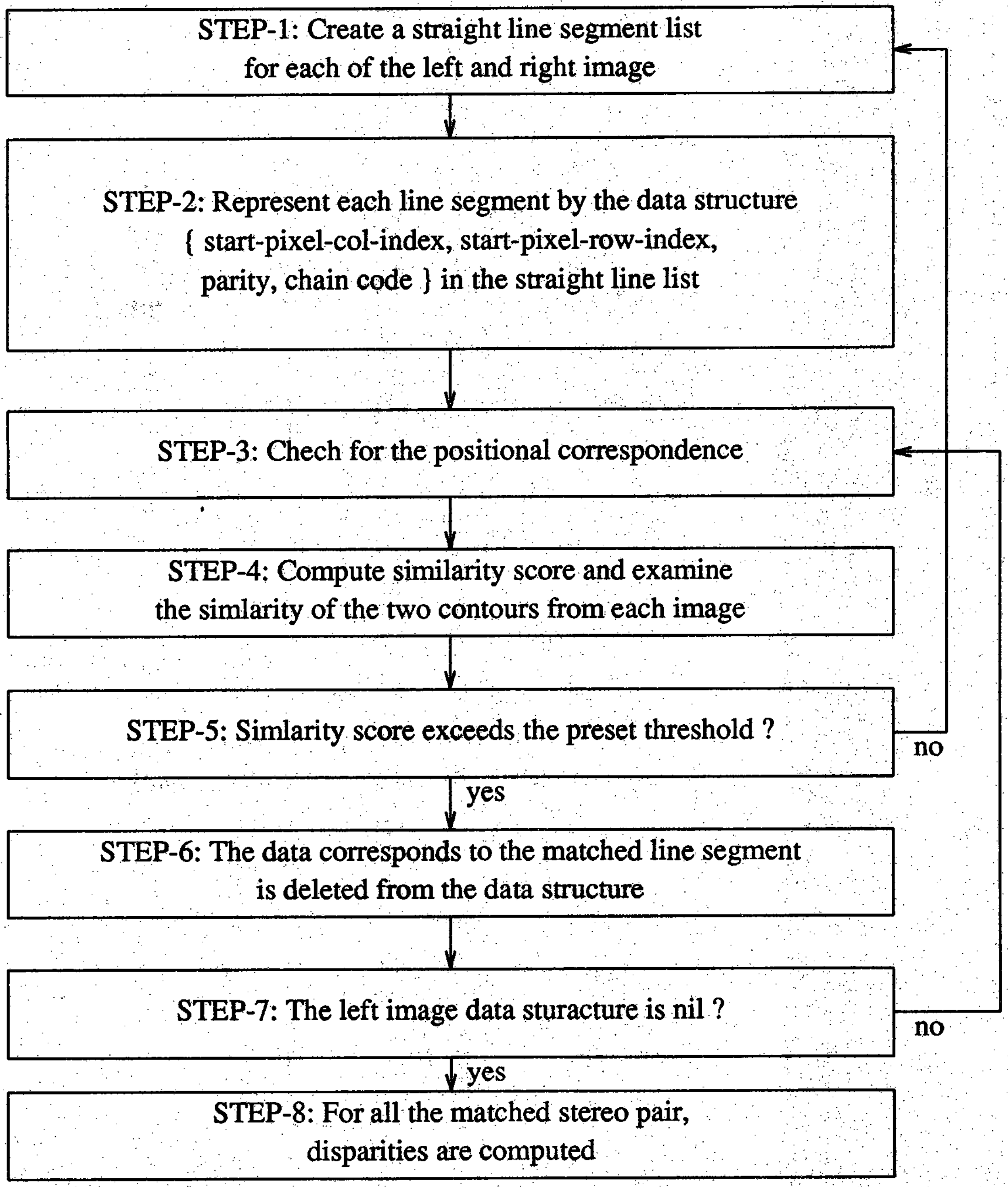

STEP-1: Create a straight line segment list for each of the left and right image
STEP-2: Represent each line segment by the data structure { start-pixel-col-index, start-pixel-row-index, parity, chain code } in the straight line list
STEP-3: Chech for the positional correspondence
STEP-4: Compute similarity score and examine the simlarity of the two contours from each image
STEP-5: Simlarity score exceeds the preset threshold ?
no
yes
STEP-6: The data corresponds to the matched line segment is deleted from the data structure
STEP-7: The left image data sturacture is nil ?
no
yes
STEP-8: For all the matched stereo pair, disparities are computed

Figure 24 Computational flow of the straight line matching.

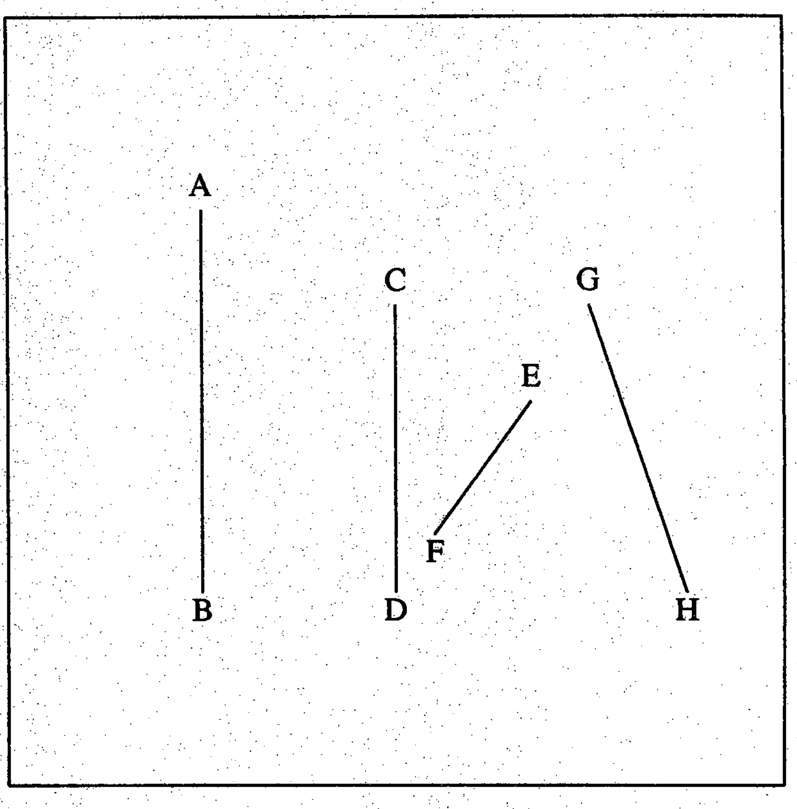


Figure 25 Straight lines into the segment list.

for the right image arcs. Note that both these chain code lists represent arcs that contain N+1 pixels.

STEP-3: In comparing the data structures for two segments from the two images, the algorithm first makes a check for the positional correspondence. This is done by examining {start-pixel-col-index, start-pixel-row-index} for the two segments. The difference of start-pixel-col-index's must be within the range of the maximum permissible value for the disparity. The difference of start-pixel-row-index's must be within a small value which is meant to account for any distortions that might be present in the epipolar geometry; this distortion might be caused by factors such as perspective effects, relative tilt between the cameras and their optic axes while being parallel in the horizontal plane but not so in the vertical plane. In the current implementation, the threshold is set to 1.

STEP-4: A similarity score, denoted by T, is computed for the two segments by comparing their chain codes. Initially, the value of T is set to 0. At step i, let the chain code elements from the two segments be L(i) and R(i). For each i, i = 1,....,N, the total score T is accumulated by using

```
if L(i) = R(i)   then T = T + 1
if L(i) = R(i-1)  then T = T + W
if L(i) = R(i+1)  then T = T + W
```

where $0 \le W \le 1$. The number W, which would always be set to a value greater than zero for matching straight line segments, captures the intuitive notion that if a chain code element is to be matched to a neighbor of its "true" correspondent then such a match must receive a reduced weight. In the current implementation, W has been set to 0.5, although we will show some matching results obtained by setting W to different values. In the event the reader is still wondering about the necessity of using neighbors in the matching process, note the following difficulty caused by the digital representation of straight line segments. Figure 26 shows two straight line segments from the left and right images; these are represented by the following chain codes.

Arc segment A : 1212121212

Arc segment B : 2121212121

Apparently, these 2 chain-code strings represent similar digital contours to the human eyes. By the algorithm shown above, the total score T is zero when W = 0, However, with W set to 0.5, the similarity score increases to 10. Note that the second element of arc A is matchable with the first and

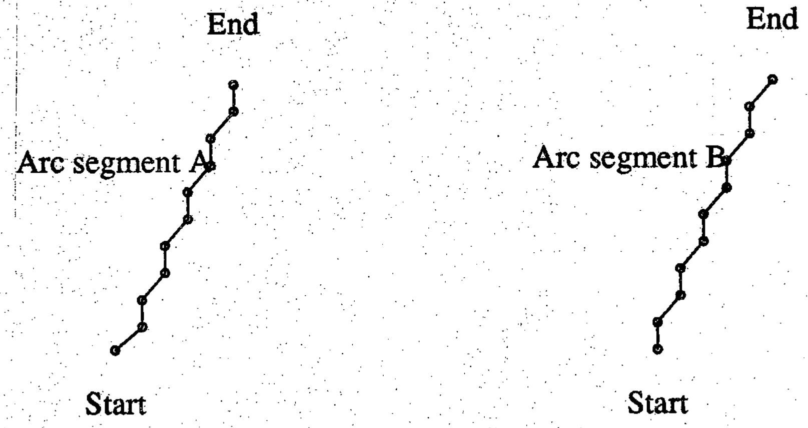


Figure 26 Chain codes of two digital arcs.

the third element of arc B, each contributing 0.5 to the cumulative, and therefore both match possibilities contributing total score of 1. The symmetry of the similarity score of two arcs A and B, which means the equality of the similarity score produced in either the case, A is in the left image and B is in the right image, or the case, B is in the left image and A is in the right image, is clearly guaranteed, referring to the Figure 27.

STEP-5: If the similarity score T does not exceed a pre-set threshold, denoted here by T3, delete the left image segment under consideration from the left image data structure created by STEP 1.

STEP-6: If the similarity score T exceeds a threshold, T3, declare the two segments as matchable. Through experimentation with segments of length 41 pixels, we have found that an appropriate value for T3 is 50. Note that the similarity score of 10 for the example shown in the previous step really does not apply to 41 pixel segments. Delete from the left image data structure shown in STEP 1 the segment which is examined.

STEP-7: If the left image data structure created in STEP 1 is non-nil, go to STEP 3.

STEP-8: In this step, range values to the scene points that lie on the matched segments are computed by the following procedure: Let D(i), i=0,1,..,N be the disparity computed from the ith pixels in the left and the right image segments; as defined in STEP-2, the chain codes of these matching pixels are denoted by L(i) and R(i), respectively. The disparities, D(i), is obtained from the chain codes L(i) and R(i) by the following formula in which A, A1, A2 are temporary variables:

```
D (0) ← CL - CR
     where CL = start-pixel-col-index for the first pixel
                in the left image segment
           CR = start-pixel-row-index for the first pixel
                in the right image segment

For i ← 1 step 1 until N do
begin
 if ( L (i) = (0, 1, or 7) )
A1 = 1;
 else if ( L (i) = (2, or 6) )
A1 = 0;
 else if ( L (i) = (3, 4, or 5) )
A1 = -1;
 if ( R (i) = (0, 1, or 7) )
A2 = 1;
```

```
 else if ( R (i) = (2, or 6) )
A2 = 0;
 else if ( R (i) = (3, 4, or 5) )
A2 = -1;

 A = A1 - A2;
 D (i) = D (i -1) + A;
end
```

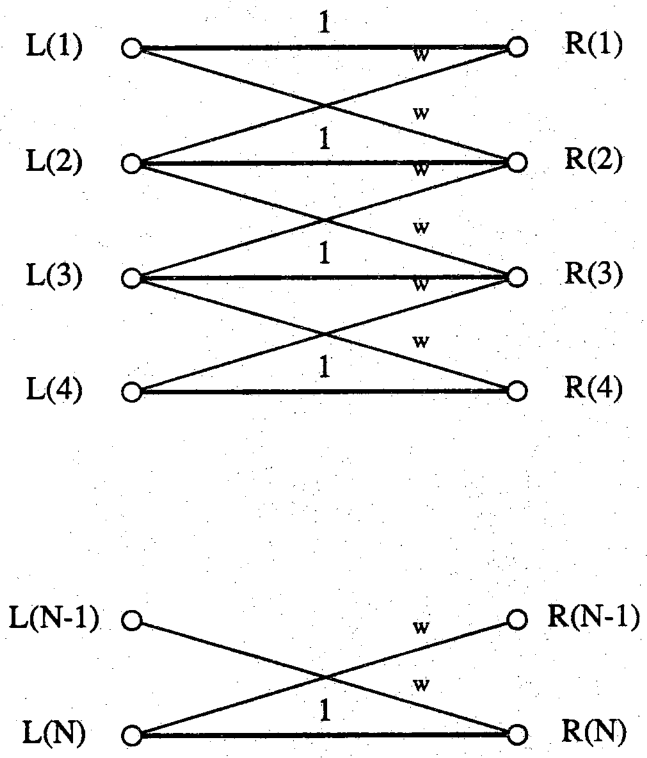
Because of the nature of the overlapping, a contour match may produce a disparity value for the point where the disparity has already computed. In this case the newly computed value is simply discarded.

Although, in most cases, the range map computations are carried out over the 256 × 256 matrices used for representing images, for display purposes the range maps down-sampled to 64 × 64. In Figure 28, twenty pairs of synthetic digital arcs are shown. The similarity score scores of each pair are summarized in Table 5.

4.1.4. Effect of Scene Illumination on the Extraction of Dominant Features

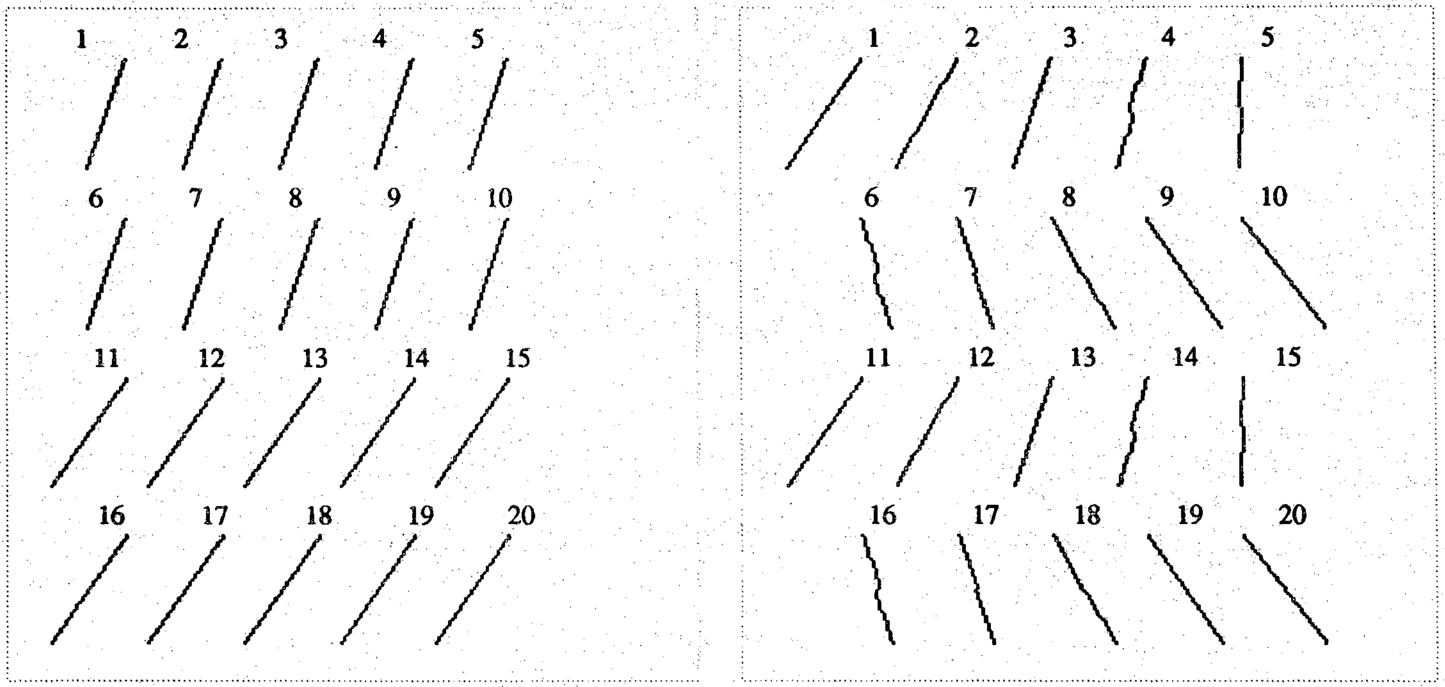
How a scene is illuminated has a considerable bearing on the quality of results obtained with the matching of straight line features. In general, it is possible to use either normal room lighting; or, one can also use the unstructured light illumination proposed first by Nishihara [10] for their PRISM system. Ordinarily, for object surfaces that are of uniform color and texture, and that, therefore, may not yield sufficiently many gray level variations, a large number of zero-crossings will be obtained with unstructured illumination. However, unstructured illumination will tend to obliterate the contrasts between adjoining surfaces, which under normal room lighting may cause the edges to become very noticeable. For this reason, for our experiments we use both illuminations. The images recorded under normal room lighting are used for the extraction of dominant features, while the those recorded with unstructured light are used for the rest of the matching strategies.

To show the reader the different results that are obtained with these two illuminations, the image in Figure 29 (a) was recorded with both types of illuminations present simultaneously; the upper half of the scene was illuminated with unstructured light, while normal room lighting was used for the lower half. Figure 29 (b) shows the extracted zero-crossings, which bear out our claims about the relative advantages of the two illuminations.



Note: L(1) through L(N) denote the chain code of the left arc.
R(1) through R(N) denote the chain code of the right arc.
An edge between L(i) and R(j) indicates that the mathcing between i th code of the left arc and j th code of the right arc is taken into accout for the similarity score computation.
The values 1 and w attached to each edge indicates the weight value for the similarity score.

Figure 27 Symmetry of the similarity score.

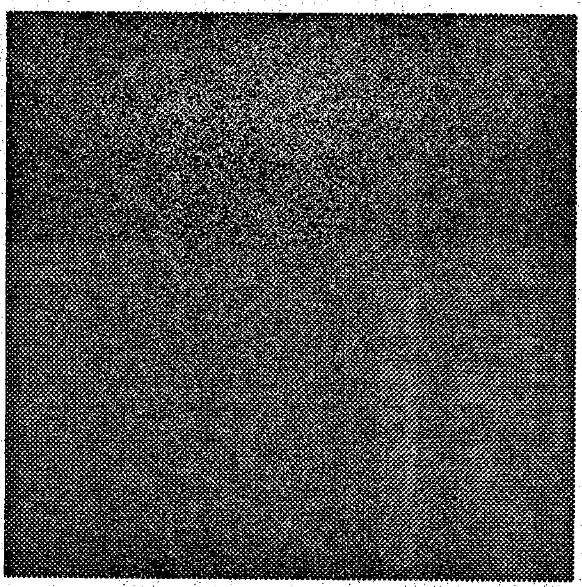


(a) Left image. **(b) Right image.**

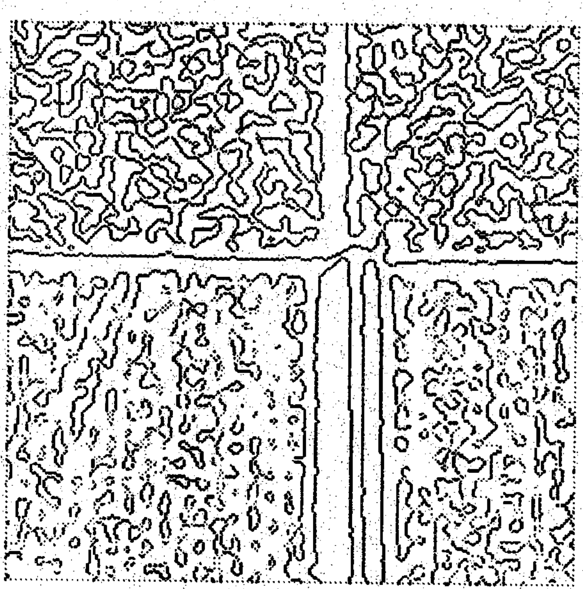
Figure 28 Similarity score measurement.

Table 5 Magnitude of matching by our method.

No.	Left Arc (Angle)	Right Arc (Angle)	Magnitude of Match	Matched? (Threshold=50)
1	71	56	47.5	no
2	71	63	49.5	no
3	71	71	54.0	yes
4	71	76	54.5	yes
5	71	90	58.0	yes
6	71	104	49.0	no
7	71	108	46.5	no
8	71	116	38.5	no
9	71	124	34.0	no
10	71	127	30.5	no
11	56	56	54.0	yes
12	56	63	52.0	yes
13	56	71	47.5	no
14	56	76	45.5	no
15	56	90	41.5	no
16	56	104	35.5	no
17	56	108	34.5	no
18	56	116	29.5	no
19	56	124	27.5	no
20	56	127	26.0	no



(a) Original image



(b) Zero-crossings ($w_{2D} = 8$).

Figure 29 A partly unstructured lighted scene and its zero crossing.

4.1.5. Some Disparity Results Obtained with Just Straight Line Feature Matching

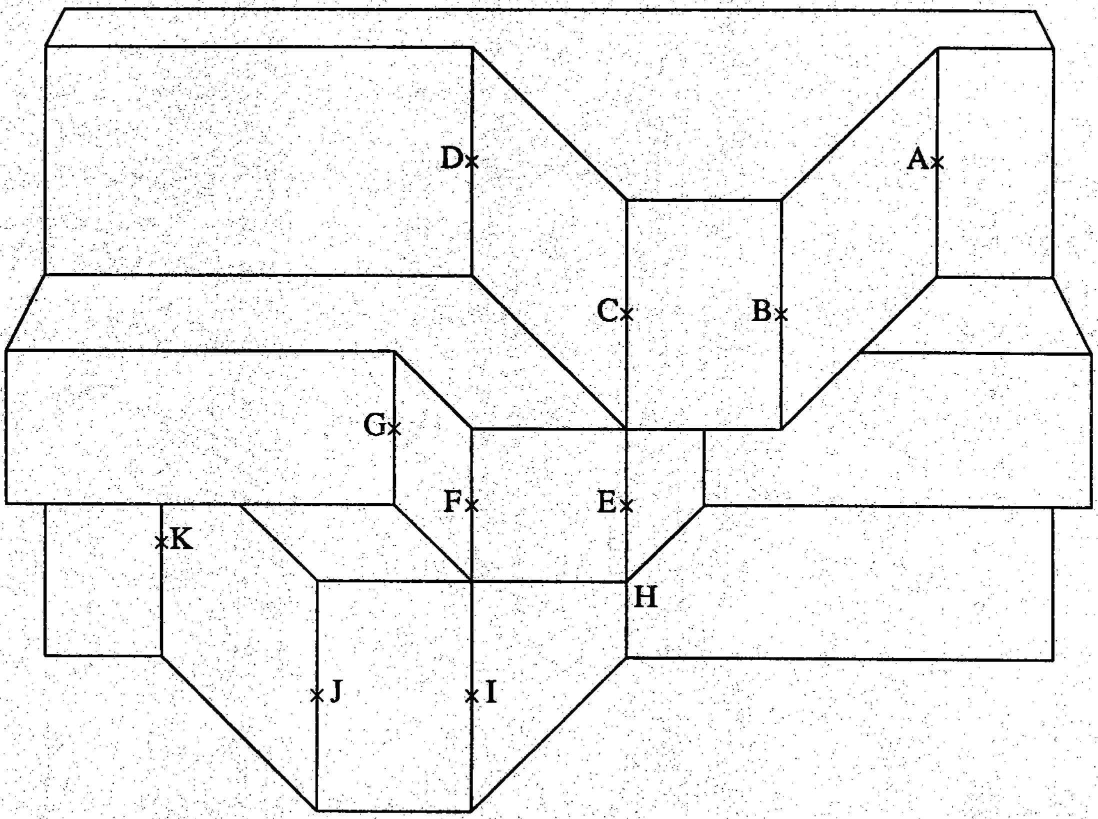
Figure 30 (a) illustrates the surface structure of an object which is rather rich in straight edges; we will use stereo images of this object, shown in Figure 31, to illustrate the results obtained with straight line matching. The stereo images were taken under normal room illumination. The straight line features extracted from the two images are shown in Figure 32. Each image there shows two types of straight lines. We have used continuous dark lines for positive straight lines, these are lines across which the gray levels increase from left to right, and negative straight lines, across these the gray levels decrease in a left to right traversal. The disparities computed by matching the straight lines in the stereo pair of Figure 32 are shown in Figure 33. Table 6 displays a comparison of the computed disparities with those obtained from the ground truth information at a set of points marked as A through K in Figure 30.

4.2. Geometrically Constrained Matching

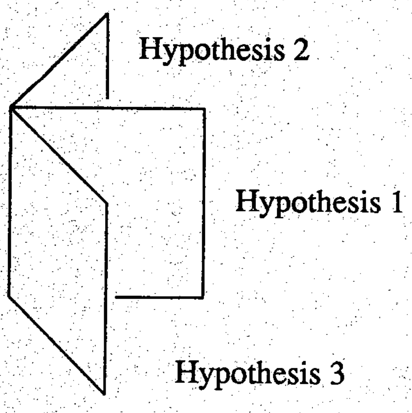
The second major matching scheme invoked by the rule-based system utilizes matching under geometrical constraints. As will be explained later, this type of matching is invoked for image regions that are in the vicinity of the straight line features matched by the method discussed previously. Matching of straight lines from the left and the right images yields range information about a straight edge in 3D space, the geometrically constrained matching then extends this to space in the neighborhood of the straight edge.

Although the mathematical procedure used for matching under geometrical constraints is essentially the same as that used by Eastman and Waxman [6], there are important differences. As mentioned before in Section 5, we only apply this type of matching to the zero-crossings produced by the finest LOG filter. Also, as should be evident from the introduction to this section, the matching is performed only in the vicinity of regions where we are able to match straight line features. Furthermore, a priori known hypotheses about planar surface orientations are used for the geometrical constraints.* This acts as a powerful constraint on the matcher and eliminates many false matches.

*There are important applications where we may assume that the orientations of the major surfaces are known and available to the stereo matcher. For example, for a mobile robot engaged in hallway navigation, we may safely assume that all the major surfaces are either vertical or horizontal. We believe that utilization of such knowledge, when available, can only lead to more robust stereo algorithms.

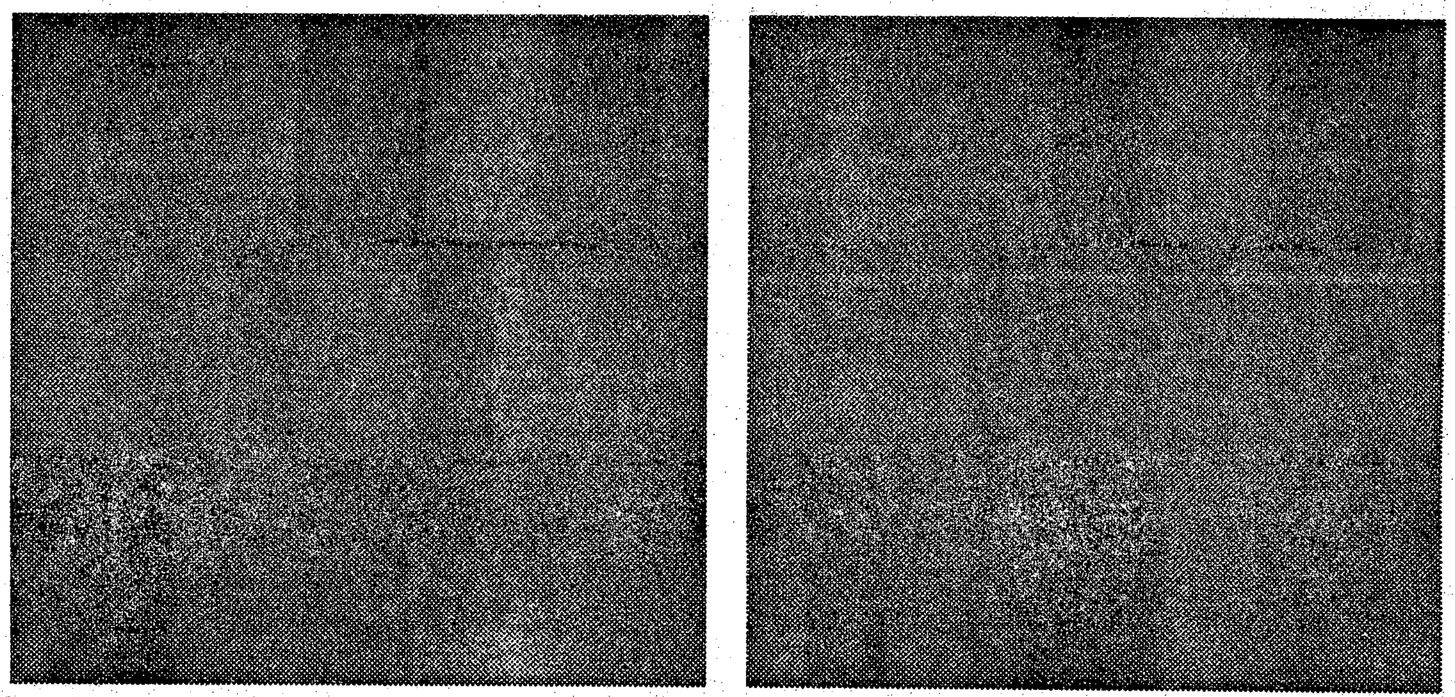


(a) Object surface.



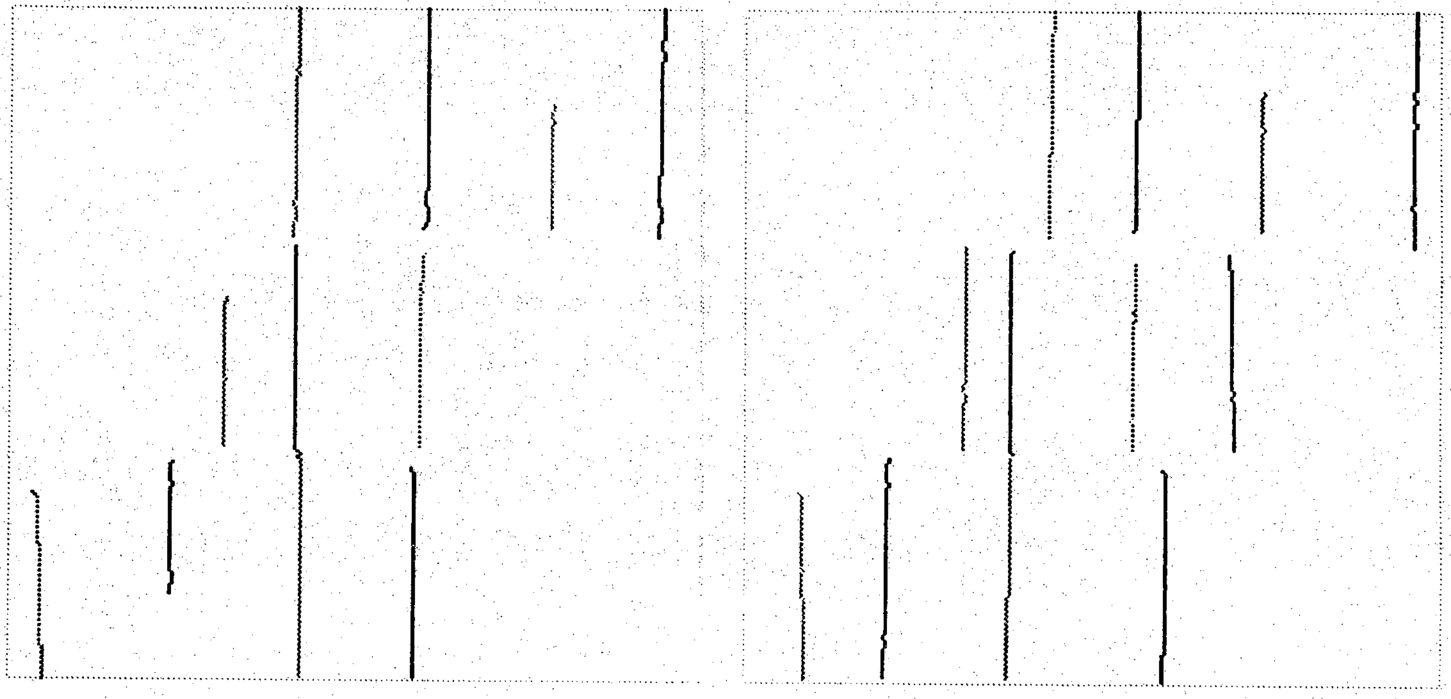
(b) Hypotheses.

Figure 30 An illustration of the object surface with relatively rich feature.



(a) Left image. (b) Right image.

Figure 31 A Stereo Scene of Figure 30.



(a) Left image. (b) Right image.

Figure 32 Straight Lines.

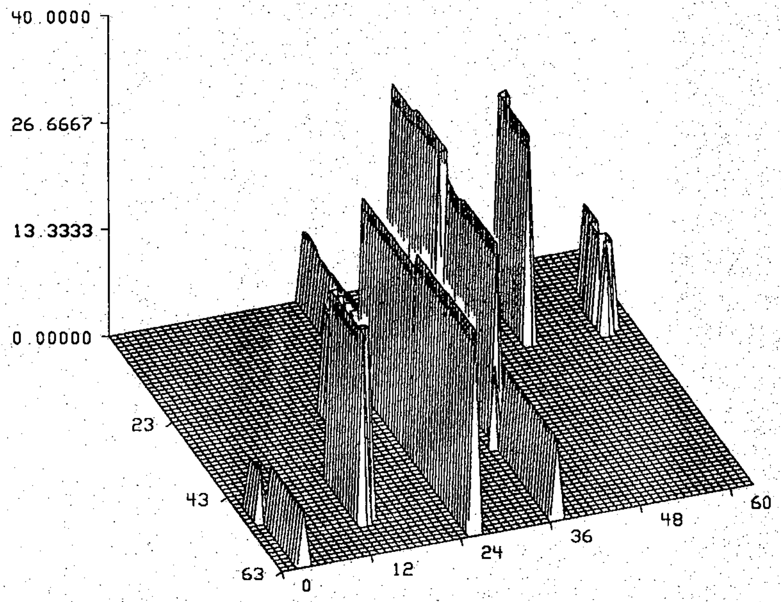


Figure 33 Depth map obtained by the straight line matching.

Table 6 Accurcy of measurement (disparity v. s. actural depth).

Image Point	Disparity in terms of depth [inches]	Actual Depth [inches]
A	-0.8	-0.6
B	3.8	3.6
C	3.6	3.5
D	-1.5	-1.2
E	3.0	3.0
F	2.9	2.8
G	0.0	0.0
H	-1.1	-1.1
I	3.2	3.0
J	2.8	2.8
K	-1.9	-2.5

Image points A though K are the points marked as × in Figure 30 (a).

We will use the triples (x,y,z) to denote our disparity values with x and y taking integer values from 1 through 64, and z taking a floating point value from 0 to 31. As explained in the footnote, this constitutes a reduced resolution disparity map.** In 3D space, let L be a straight line segment derived by the straight line feature matching process discussed in the preceding section. Let the coordinates of the terminal points of L be (x_1,y_1,z_1) and (x_2,y_2,z_2). Let P_θ denote a hypothesized plane, of orientation θ, which contains L and which corresponds to one of the object surfaces. For our algorithms, the orientation is expressed in the following manner: we compute the intersection of the plane with the y=0 plane, the orientation of P is then measured by the angle θ subtended by the intersection line with the x-axis. See Figure 34. As mentioned before, the orientation θ is assumed to be one of the set $\{ \theta_1,\theta_2,..... \}$ of a priori known orientations of all planar surfaces in the scene.

Given an orientation θ for a hypothesized plane and the terminal points $\{ (x_1,y_1,z_1), (x_2,y_2,z_2) \}$, the complete equation of the plane is given by

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_1+1 & y_1 & z_1+\tan\theta & 1 \end{vmatrix} = 0. \qquad (20)$$

which merely expresses the fact that the terminals points and the point $(x_1+1,y_1,z_1+\tan\theta)$ must lie on the plane. The operator $|.|$ stands for taking the determinant of its argument.

In geometrically constrained matching, our first task is to generate the plane that contains L and whose orientation corresponds to one of the hypothesized planes. In the 64×64 representation for disparity maps, this plane will extend to only 4 pixels on either side of the line (Figure 34). The choice of 4 is arbitrary and dictated by the nature of the scenes we are interested in. If it is believed that the planar surfaces in a

**Usually, the disparity maps tend to be of lower resolution than the images from which they are produced (see, for example, the work by Nishihara [10]). In our processing, the images are usually of size 256×256, whereas the disparity maps tend to be defined only over 64×64 matrices. We will denote a disparity map by the triple (x,y,z), where x and y take integer values 0,1,2,.....,63, and z takes floating point values that span the interval [0,32]. The reason why z is floating point over this interval has to do with the fact that when straight line features are matched, the disparities are computed over 256×256 images, implying that the range of computed disparities would be any integer from -128 to +128; in practice, of course, the disparity range is limited to a much smaller interval, usually [-15, 16]. When computing the reduced resolution disparity maps, the disparity values are averaged, leading to floating point numbers.

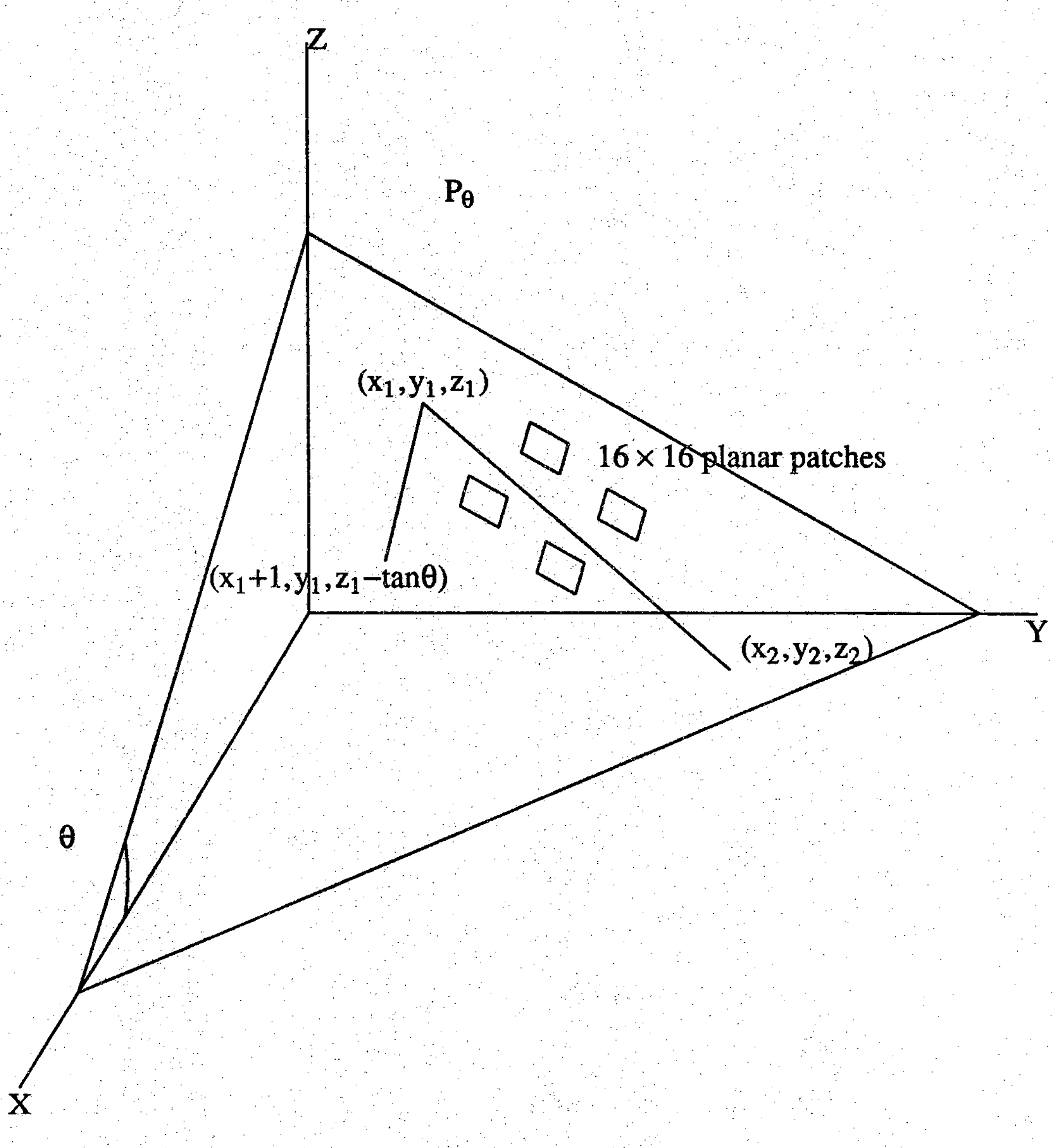


Figure 34 Geometry of the hypothetical range.

polyhedral scene consist of large areas, then this number could be larger. Note that in terms of the row-column resolution in the original images, the width of 4 in the disparity space is equivalent to a width of 16.

In the implementation described by the following algorithm, a pass is made through the 64×64 disparity map and planar strips are fitted to all the straight lines at the same time, all the planar strips thus generated being of the same orientation. As described precisely in a following section (Section 4.2.3), the geometrically constrained matching is carried out in a 16×16 square patch on the planar strip instead of enforcing the match within the entire planar strip at one time. This step is followed by fitting all the lines again with strips of another hypothesized orientation, and the process continued. For each hypothesized orientation, the computation of the geometrically constrained matching can be divided into two stages. The first and the second stages are called *the planar strip generation algorithm*, and *the geometrically constrained matching algorithm*, respectively. The algorithm of each stage is shown in the following part of this section. In advance to the discussion on each algorithm, a data structure to record the matching schemes, which we call the control matrix, is introduced in the following subsection.

4.2.1. Control Matrix

For our 256×256 image matrices, a 64×64 control matrix is used to organize the flow of control during the rule based implementation of binocular fusion. This control matrix also plays an important role in the execution of the geometrically constrained matcher, therefore we will define it in this subsection.

The initial entries for the control matrix are provided by the results of matching the straight line features. To generate these initial entries, the image is divided into 64×64 non-overlapping blocks, each of size 4×4. If in a given 4×4 block, the system is able to successfully match a straight line feature, then in the control matrix an entry of 1 is made for that block. For all the blocks that overlapp with the regions grown around the straight lines features, the entry in the control matrix is 2. Elsewhere, the entry is 3.

During the execution of the rule-based system, the entries of the control matrix can be altered on the fly. For example, if it is concluded that a region cannot be matched with the help of zero-crossing contours, by the method discussed in Section 4.3, then the entries for all the blocks that are strictly within the region are set to 4.

In the next chapter, we will provide other mechanisms that can alter the control matrix entries on the fly during the process of matching.

4.2.2. Planar Strip Generation Algorithm

For the purpose of the planar strip generation, the control matrix can hold the value of either 1 or 2 such that

(1) C(x,y) = 1 : if the range of the coordinate (x,y) in the disparity map is already known through dominant feature matching,
and
(2) C(x,y) = 2 : if (x,y) is in the vicinity of a point (x_1,y_1) where $C(x_1,y_1)=1$. The notion of vicinity is defined by a four-fold growth of a connected region from the pixels where C(x,y) is 1.

In the following algorithm, the first step grows the regions where the control matrix elements should equal 2; the second step then, for a given value of hypothesized plane orientation θ, computes the horizontal component of the corresponding disparities at all points where C(x,y) equals 2; finally, the third step computes the vertical component of the disparities, and, subsequently, the total hypothesized disparities corresponding to the given θ.

In the algorithm, H(x,y) represents the planar strips generated from all the lines, all the strips being of the same orientation θ. The function H(x,y) is generated iteratively by propagating z values outwards from the lines such that the resulting (x,y,z)'s are always on planes of the desired orientation. In this iterative computation, $H_t(x,y)$ denotes the partial strips reconstructed at iteration t; the subscript t takes integer values from 0 through 8. So, when t=0, $H_0(x,y)$ corresponds to the lines themselves.

The STEP-1, in the algorithm described below, creates the control matrix. Subsequently, in the STEP-2 and STEP-3 the planar strip is generated and stored in the 64 × 64 matrix H (x,y).

STEP-1: [Defining the region where the strip is generated]

```
for t ← 1 until 4
 begin
 for x ← 0 until 63
  begin
  for y ← 0 until 63
   begin
   if ( any 8-neighbor of $C_{t-1}(x,y)$ is 1 or 2 )
     $C_t(x, y) = 2$;
   end
  end
```

```
            end
```

STEP-2: [Span the hypothesis in x-direction]

```
          Initially, for all (x,y), $H_0(x,y) \leftarrow D(x,y)$ .
          Note that D(x,y) is the disparity map corresponding
          to the straight lines features found from the stereo pair.
          for t = 1 to 4
           begin
           for x ← 0 until 63
            begin
            for y ← 0 until 63
             begin
             if ( ( $C(x,y) \neq 1$ or  $C(x,y) \neq 2$ )
                 and ( $C(x+1,y) = 1$ or $C(x+1,y) = 2$ ) )
                begin
                 $H_t(x,y) = H_{t-1}(x+1, y) - \tan\theta$;
                end
             else if ( ( $C(x,y) \neq 1$ or  $C(x,y) \neq 2$ )
                 and ( $C(x-1,y) = 1$ or $C(x-1,y) = 2$ ) )
                begin
                 $H_t(x,y) = H_{t-1}(x-1, y) + \tan\theta$;
                end
             end
           end
```

STEP-3: [Span the hypothesis in y-direction]

```
          for t = 5 to 8
           begin
           for x ← 0 until 63
            begin
            for y ← 0 until 63
             begin
             if ( ( $C(x,y) \neq 1$ or  $C(x,y) \neq 2$ )
                 and ( $C(x,y+1) = 1$ or $C(x,y+1) = 2$ ) )
               begin
                $S_v = H_{t-1}(x,y+1) - H_{t-1}(x,y+2)$;
                $H_t(x,y) = H_{t-1}(x, y+1) + S_v$
               end
```

```
    else if ( ( $C(x,y) \neq 1$ or  $C(x,y) \neq 2$ )
        and ( $C(x,y-1) = 1$ or $C(x,y-1) = 2$ ) )
        begin
            $S_v = H_{t-1}(x,y-1) - H_{t-1}(x,y-2)$;
            $H_t(x,y) = H_{t-1}(x, y-1) - S_v$;
        end
    end
```

The planar strips H(x,y) is provide by the final result of the recursive method of the STEP-3, which is denoted by $H_8(x,y)$. Figure 35 shows the flow chart of this algorithm.

4.2.3. Matching Zero Crossings Under Planar Strip Orientation Constraint

Suppose we have N hypotheses for planar surface orientations in a scene. The algorithm in the preceding section will, for each matched pair of straight line segments, generate a planar strip in 3-D space around the fused straight line feature. The next order of business is to accept only those hypotheses which lead to a maximum number of zero-crossing matches on these planar strips. Again from the previous section, each planar strip is represented as a disparity map on a 64×64 matrix. Therefore, for a pair of stereo images, at this point we will have N 64×64 disparity maps corresponding to the N hypotheses. In making the transition from 256×256 images to 64×64 disparity maps, the disparity of a pixel (i,j) in the image plane is stored at the coordinates $(x = \lceil \frac{i}{4} \rceil, y = \lceil \frac{j}{4} \rceil)$ in disparity map.

In enforcing the constraints corresponding to the different available hypotheses, in practice it is not advisable to use each strip in its entirety. This is owing to fact that an edge would in general form a boundary between planar faces of different orientations -- the planar faces corresponding to different hypotheses. It is also possible that two or more objects might be lined up in just the right manner so that a long edge is created in the images, but, clearly, at different places along such an edge the planar faces would have different orientations. For all these reasons, each planar strip is divided into $M \times M$ overlapping square patches, as shown in Figure 36, where M equals 16. Each $M \times M$ patch is referred to by its upper left hand corner and corresponds to a $M/4 \times M/4$ submatrix in the control matrix.

For example, a patch at location (x_0,y_0) in a disparity matrix spans the region within the range of $x_0 \leq x < x_0+M/4$ and $y_0 \leq y < y_0+M/4$. The same patch spans the region given by $4x_0 \leq i < 4x_0+M$ and $4y_0 \leq j < 4y_0+M$, in the image frame. When M is 16,

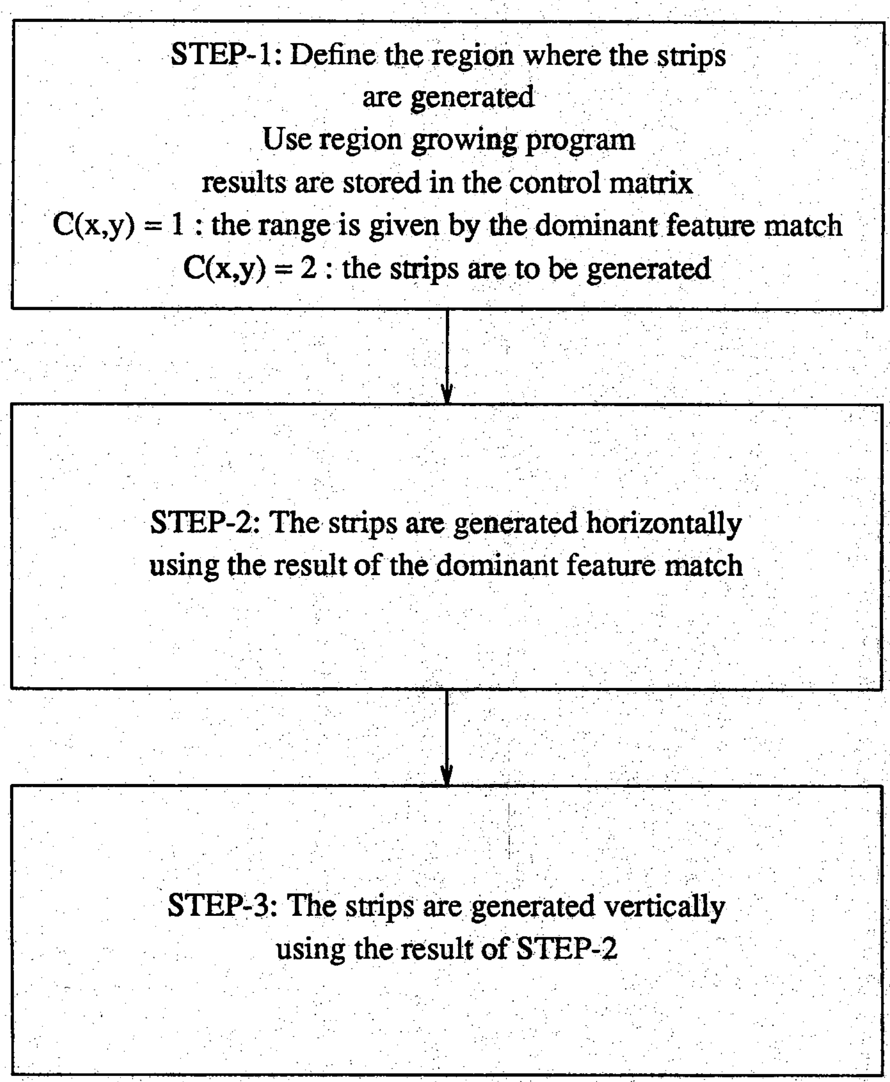

STEP-1: Define the region where the strips are generated
Use region growing program
results are stored in the control matrix
C(x,y) = 1 : the range is given by the dominant feature match
C(x,y) = 2 : the strips are to be generated
STEP-2: The strips are generated horizontally using the result of the dominant feature match
STEP-3: The strips are generated vertically using the result of STEP-2

Figure 35 Flow chart of the strip generation algorithm.

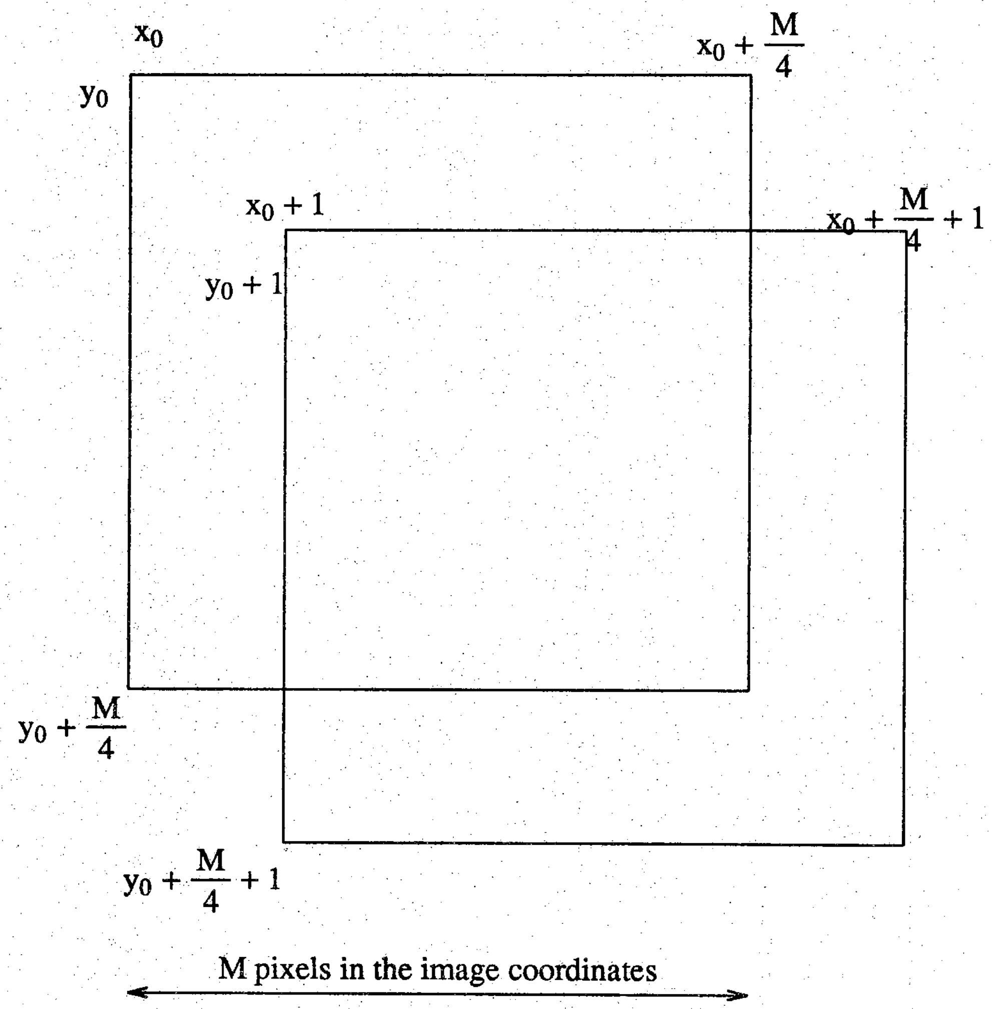


Figure 36 Planar patch coordinate system.

note that in a disparity matrix, the maximal coordinate values for a patch are (61, 61); that is because for locations beyond these values, complete 16×16 patches cannot be accommodated in a disparity matrix. Also note that adjacent patches overlap for all but one row and one column. For example, the patches located at (x_0, y_0) and $(x_0 + 1, y_0 + 1)$ overlap as shown in Figure 36.

In the algorithm described below, two variables T and S play important roles in determining whether a given hypothesis is good or not. A hypothesis is considered to be valid if a sufficient number of zero-crossings can be matched on a planar patch, this number being a fraction of the total number of zero-crossings in the patch. The variable S is equal to the total number of zero-crossings in a patch and T the number of zero-crossings which can be successfully matched according to the constraints imposed by the hypothesis corresponding to the patch. The validity of the hypothesis is then established by comparing T/S against a threshold.

Say, a left image zero-crossing is located at coordinates (x,y). Then, in order to establish correspondence for the zero-crossing, we construct a search window of size $\pm\delta$ in the right image located at x+d where d corresponds to the disparity value in the left image patch, the value of this disparity is given by $H_m(x/4, y/4)$. By trial and error, we have concluded that in practice the value of δ should be limited to unity. In other words, the search windows are of size ± 1.

We have also observed through experimentation that the acceptance threshold on T/S -- this threshold will be denoted by T4 -- should be a function of the orientation of the plane corresponding to the hypothesis. For example, when the orientation of a plane is given by θ=0, meaning the plane is parallel to the image planes, the threshold T4 should be approximately 0.7, whereas for planes with θ=45deg, the best value for T4 is around 0.5.

The following is an algorithmic description of the above procedure for accepting or rejecting a planar surface hypothesis in the vicinity of a straight edge. For a given patch, the procedure, which consists of two steps, STEP-1 and STEP-2, in invoked only if the control matrix entries for the entire patch are 2.

STEP-1: [Searching correspondence]

Let LP (i,j) and RP (i,j) be the left and right image "zero-crossings matrices," defined in the following manner: For i and j spanning the range (0, 255), we define the matrices LP and RP in the following manner:

LP or RP (i , j) = 'p' if the coordinate (i,j) in the image is
on a positive contour,

LP or RP (i , j) = 'n' if the coordinate (i,j) in the image is
on a negative contour.

and
LP or RP (i , j) = 'h' if the coordinate (i,j) is the image is
on a horizontal contour.

For every patch,
let (x_0, y_0) be the patch coordinate in the disparity
space.
Now, the number T and S for the patch at
(x_0, y_0)
can be computed by the following algorithm.

```
S ← 0;
T ← 0;
for i ← 4 $x_0$ until 4 $x_0$ + M - 1
 for j ← 4 $y_0$ until 4 $y_0$ + M - 1
   begin
     if ( LP ( i, j ) = 'p' )
     begin
                 T ← T + 1;
       if(RP($i_1$+H(⌈ $i_1$/4⌉ ,⌈ $j_1$/4⌉ ),$j_1$)='p'
        or
        RP($i_1$+H(⌈ $i_1$/4⌉ ,⌈ $j_1$/4⌉ )+1,$j_1$)='p',
        or
        RP($i_1$+H(⌈ $i_1$/4⌉ ,⌈ $j_1$/4⌉ )−1,$j_1$)='p')
          S ← S + 1;
     end
     if ( LP ( i, j ) = 'n' )
     begin
       T ← T + 1;
       if(RP($i_1$+H(⌈ $i_1$/4⌉ ,⌈ $j_1$/4⌉ ),$j_1$)='n'
        or
        RP($i_1$+H(⌈ $i_1$/4⌉ ,⌈ $j_1$/4⌉ )+1,$j_1$)='n',
        or
        RP($i_1$+H(⌈ $i_1$/4⌉ ,⌈ $j_1$/4⌉ )−1,$j_1$)='n')
          S ← S + 1;
     end
   end
```

STEP-2: This step is for computing the ratio R of T and S. If R exceeds a certain threshold, then all stereo correspondence in this patch are regarded as correct.

```
$R \leftarrow \frac{T}{S}$

if ( $R \geq T4$ )
    return ("This planar patch is good hypothesis")
if ( $R < T4$ )
    return ("This planar patch is not good hypothesis")
```

Figure 37 shows a flow chart for the algorithm. To demonstrate with an example the workings of the algorithm, consider the three-panel scene shown in Figure 38 (a). For this scene, the dominant straight lines, marked as AB and CD, will be matched by the method of Section 4.1.3. At this point, our system assumes that a set of hypotheses about planar surface orientations is available for matching regions around the dominant straight lines. For the sake of discussion, let's say that the available hypotheses are as illustrated in Figure 38 (b). This means that at each pixel in the vicinity of the matched straight lines, the system will invoke one of these hypotheses and select the best possible one on the basis of number of zero-crossings matched.

To demonstrate how well this procedure works, we have shown in Figure 39 a pair of stereo images of an actual 3-panel scene. As in the sketch in Figure 38 (a), the dominant edges in the scene are at the junctions of the panels. The top image in Figure 40 (a) shows the pixels that were matched by the geometrically constrained matcher using planar orientation hypothesis corresponding to *orientation1* shown in Figure 38 (b). The figure has a "blocky" appearance because, as mentioned before, the computed disparity maps are over 64×64 matrices, while the results are displayed using 256×256 matrices. The bottom plate in Figure 40 (b) shows the pixels matched using the hypothesis that these pixels are on a plane of orientation *orientation2*.

In Figure 41, we show the results obtained for the stereo images of Figure 31. We used three hypotheses for the orientations of the planes (the range map of three hypotheses are displayed in Figure 42); these are displayed in Figure 30 (b). The three plates in Figure 30 (b) show the pixels matched using each of the three hypotheses.

4.3. Matching of Zero-Crossing Contours

A combined execution of the dominant-feature and the geometrically-constrained matchers yields good depth maps in the vicinity of strong edges in a sense. Elsewhere, the system is dependent upon the two-level matcher that will be presented in this

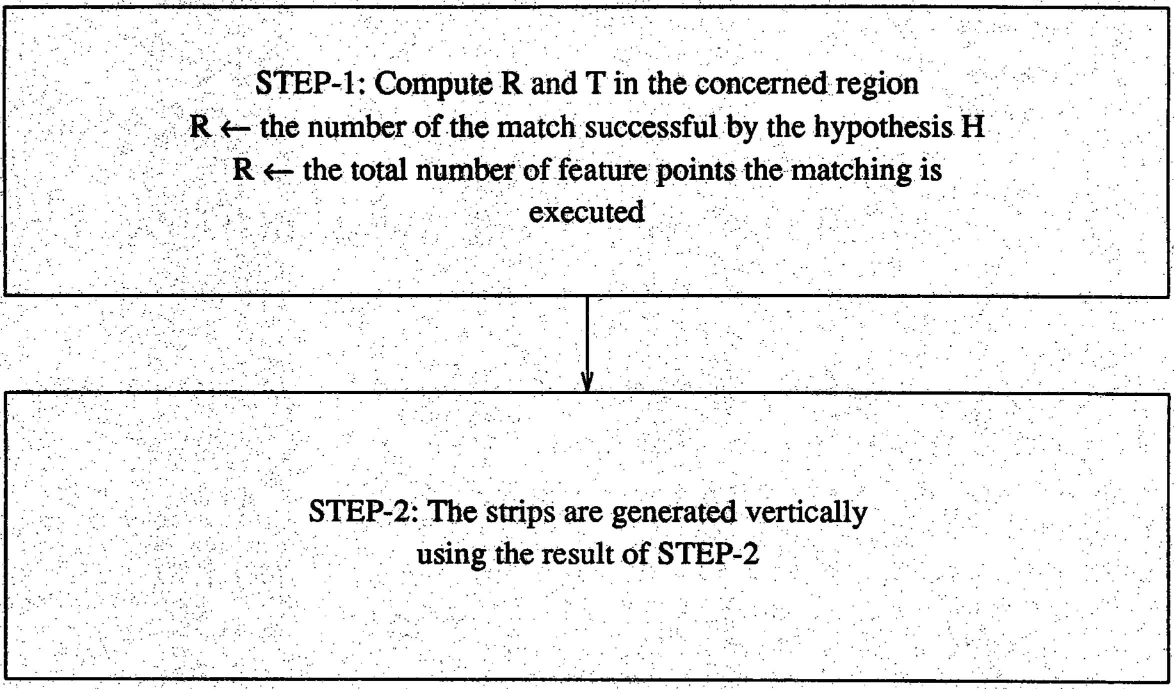
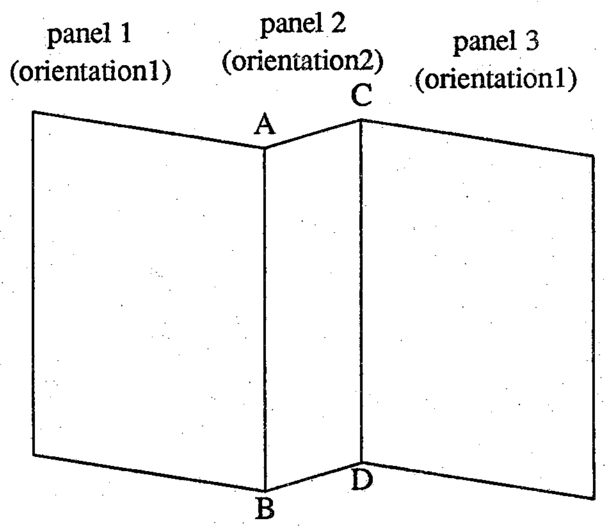
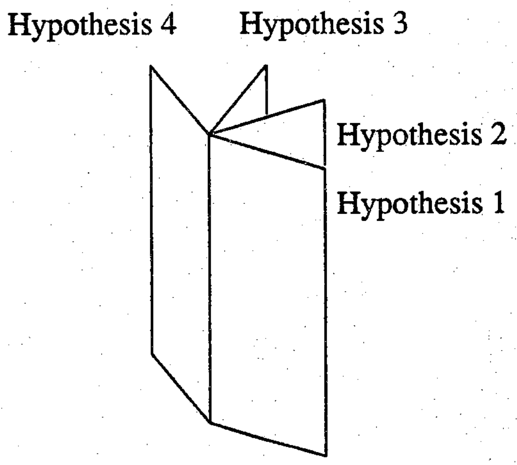

STEP-1: Compute R and T in the concerned region
R ← the number of the match successful by the hypothesis H
R ← the total number of feature points the matching is executed
STEP-2: The strips are generated vertically using the result of STEP-2

Figure 37 Flow chart of the geometrically constrained matching algorithm.

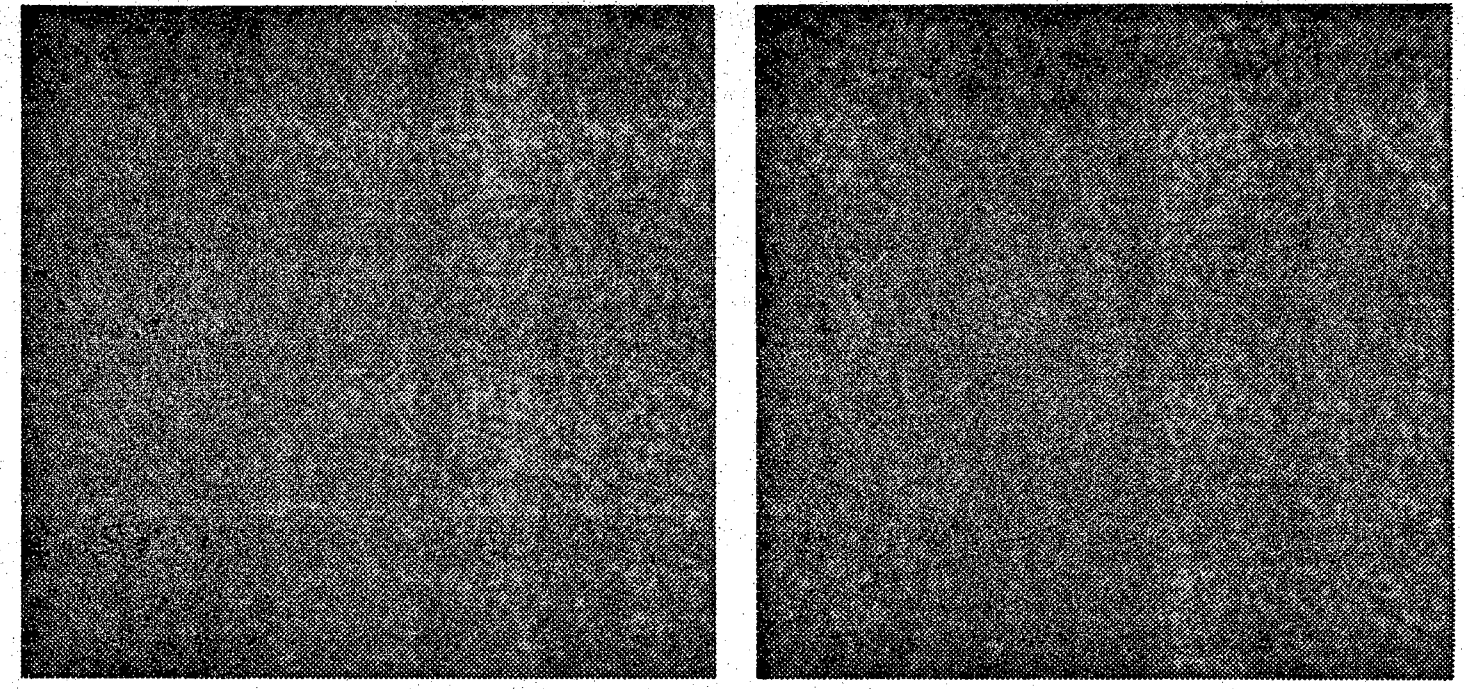


(a) The surface shape.



(b) Hypotheses.

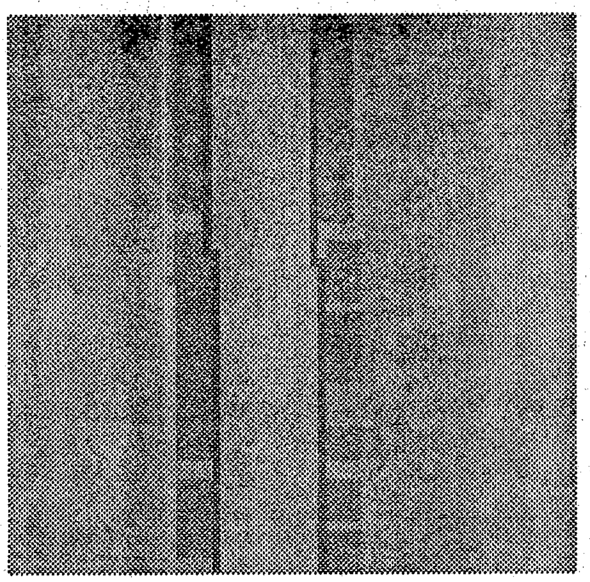
Figure 38   An illustrative three panel scene.



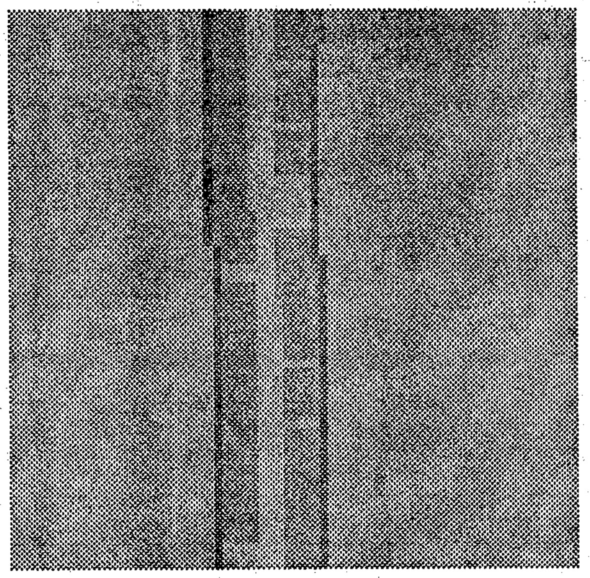
(a) Left image.

(b) Right image.

Figure 39 An illustrative scene for geometrically constrained matching.

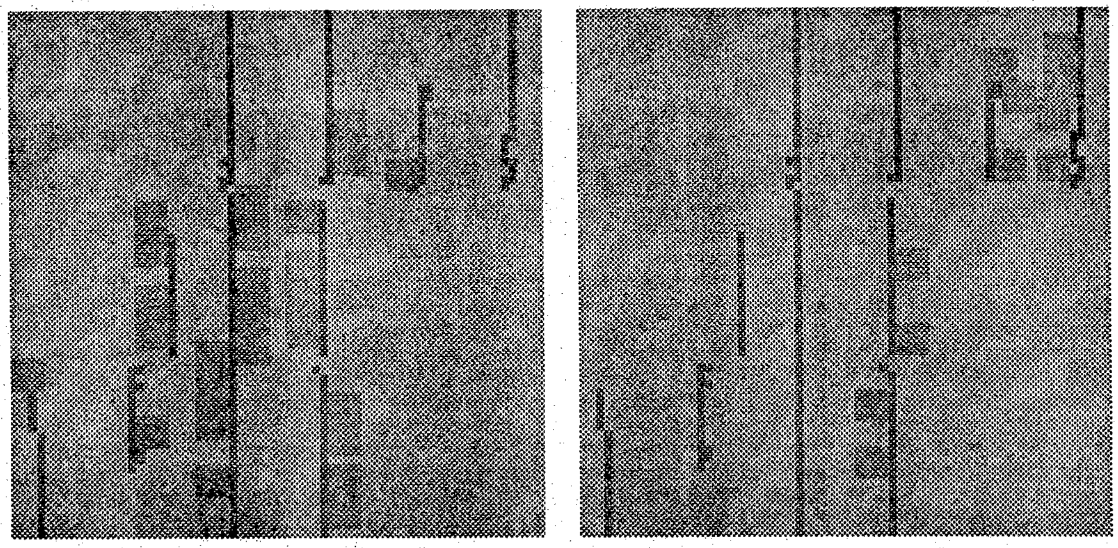


(a) Hypothesis 1.



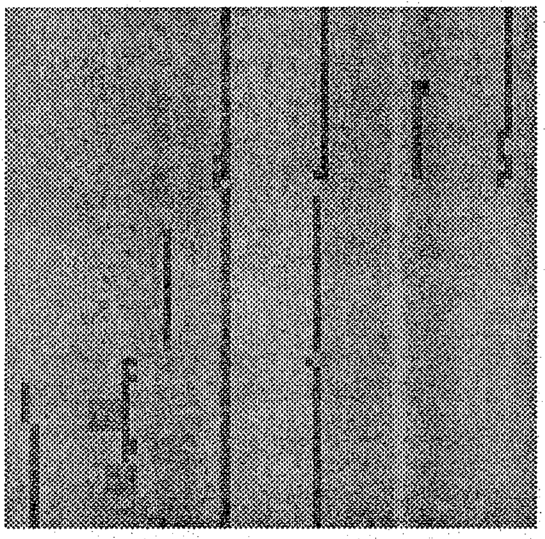
(b) Hypothesis 2.

Figure 40 The region where the geometrically constrained matching is successful.



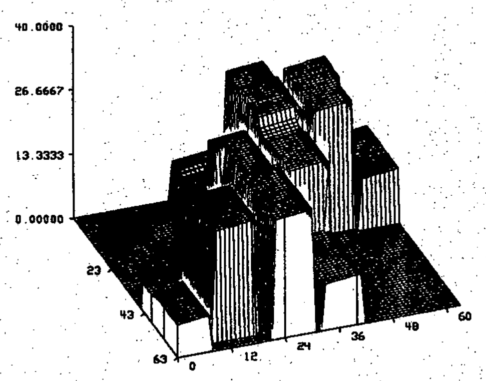
(a) Hypothsis 1.

(b) Hypothesis 2.

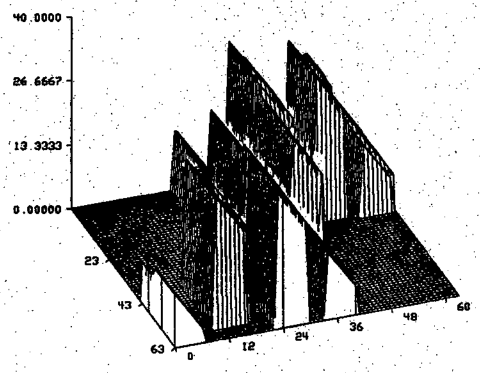


(c) Hypothsis 3.

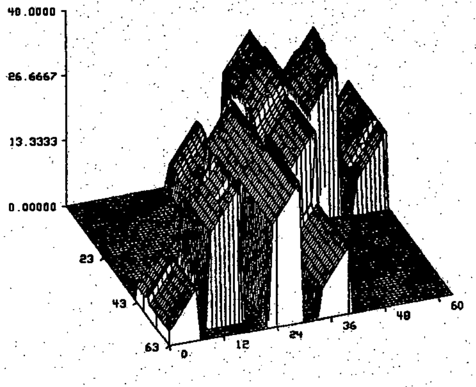
Figure 41 The area where the hypothesis is successful.



(a) Hypothesis 1



(2) hypothesis 2



(c) Hypothesis 3

Figure 42 Three hypotheses generated for the scene of Figure 30.

section and, if that does not work either, the full implementation of the MPG algorithm is used. The matcher is this section is based on the notion of figural continuity of zero-crossing contours; this continuity constraint says that when the planar surface is viewed from two neighboring viewpoints, the zero-crossing contours in both the stereo images must have nearly identical shapes. The idea of figural continuity was first suggested by Mayhew and Frisby [8] because they felt that the sign and the orientation of a zero-crossing contour did not constitute sufficient constraints for binocular fusion in the human visual system. Subsequently, figural continuity was incorporated by Grimson [9] in the MPG algorithm.

In the two level matching scheme described here, figural continuity constraints are invoked for matching segments of the zero-crossing contours produced by the w_{2D}=8 channel. The disparity map so obtained is then used to generate a higher resolution version by matching the zero-crossings for the finest channel for which w_{2D} is 4. The decision about the resolution level at which matching should be performed by invoking figural continuity constraints is not as arbitrary as it seems and is dictated by the procedure we use for testing the shape similarity of two contour segments. The shape similarity testing algorithm is such that it needs at least 10 pixels to arrive at reliable conclusions. The zero-crossing channel for w_{2D}=4 generates too many contours that are too short to be processed by the shape similarity algorithm. Note that random-dot illumination is used to generate images for all matching schemes except the first one that is based on the matching of straight line features. Our statement about too many contours being too short is applicable only under this illumination.

There is an important point to be made about why two channels are sufficient for generating depth values over large depth ranges with the scheme presented in this section. In the Marr-Poggio theory, if in each channel the search window for finding the correspondent in the right image of a left image zero-crossing is equal to $\pm w_{2D}/2$, then in over 95 percent of the cases there will be a single zero-crossing in the search window. If the search windows are made much larger than this, one has to contend with the problem of disambiguation; since large windows will contain multiple zero-crossings, from these one must be selected to serve as a correspondent of the left image zero crossing. On the other hand, if the search windows are much smaller, then the probability of completely missing the correspondent is increased. In practice, most implementations of the MPG algorithm use $\pm w_{2D}$ for search windows, this larger size being made necessary by the displacements in the zero-crossing contours that was discussed in Section 2.2.3.

Because, of the limitation on the maximum size of the search window in an MPG type of an algorithm, one is forced to use a number of channels, usually four, to cover a depth range over which humans are capable of perceiving depth through stereopsis.

The disambiguation problems are not as severe with the two level matcher discussed in this section. This is because the MPG matching process, even despite

Grimson's figural continuity implementation, is essentially a pixel level matcher. On the other hand, the first level of the matching scheme described here implements stereopsis by actually fusing contour segments. This allows us to use larger search windows without suffering from the disambiguation problem. Contour segments tend to be much richer in detail compared to zero-crossing pixels even when contour slopes are associated with the latter. For this reason, it is sufficient for us to use two channels.

In the following subsections, we will first show with examples that the shapes of zero-crossing contours do indeed stay the same for the two viewpoints we use for stereo images. This observation is important since it forms the cornerstone of the matching strategy discussed in this section. We will then discuss the figural continuity implementations of Mayhew and Frisby, and of Grimson; and point out the shortcomings of those implementations. This will be followed with our implementation, which we believe is superior to the other two.

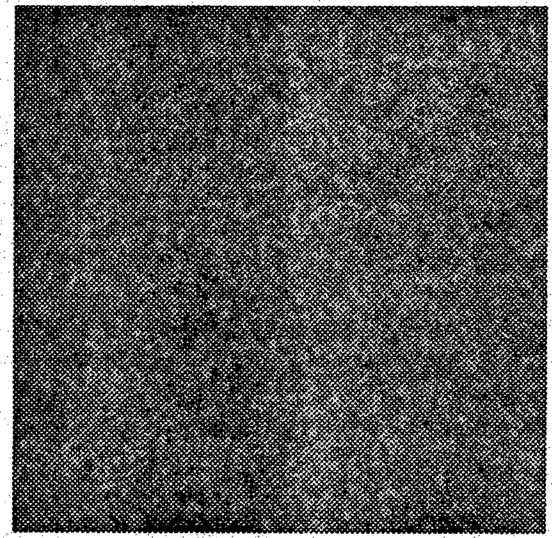
4.3.1. Photometric Invariance in Stereo Vision

Photometric invariance refers to the fact that the corresponding areas in the left and right images exhibit similar gray level variations. Since gray level variations are captured by the zero-crossings generated by the application of LOG filtering to images, photometric invariance translates into the invariance of the shapes of the corresponding zero-crossing contours. In Figure 43, we have shown stereo images of a scene with large flat surfaces. Figure 44 displays the zero-crossing contours obtained via the w_{2D} channel. The similarity of the corresponding contours is evident.

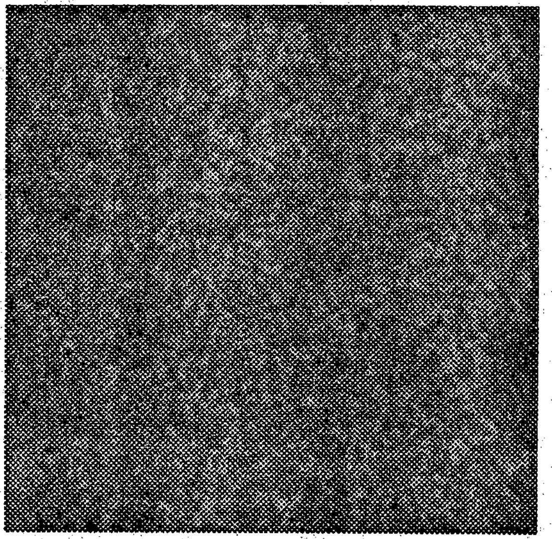
The similarity of zero-crossing contours is violated in the vicinity of range discontinuities and shadows. Some of the mechanisms that lead to distortions of the contours between the two images are the same as those discussed in Section 2.2.3.

4.3.2. Mayhew and Frisby Implementation of Figural Continuity

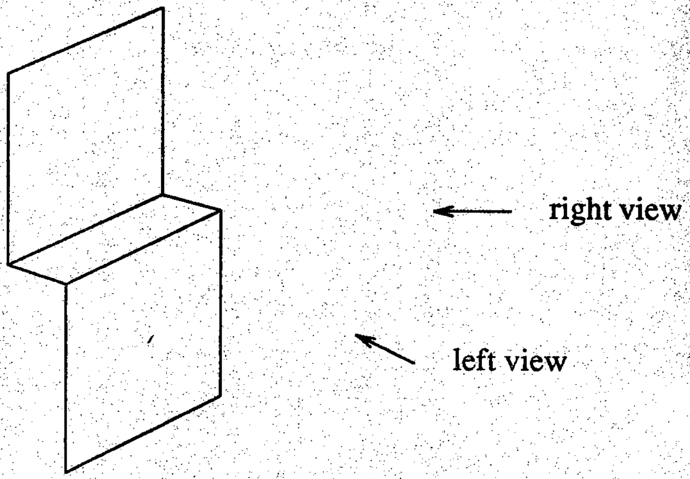
Mayhew and Frisby [8] have shown that figural continuity of zero crossings plays an important role in binocular fusion in human stereopsis. As for the evidence, they presented the result of an experiment which measured the latency time for stereopsis for image pairs with and without figural continuity cues. Shown in Figure 45 is a random dot stereogram, superimposed on which are boxes that define the regions which undergo disparity shifts from the left to the right image. Mahew and Frisby showed that the time it takes to fuse the stereogram is shorter when the box-



(a) Left image.

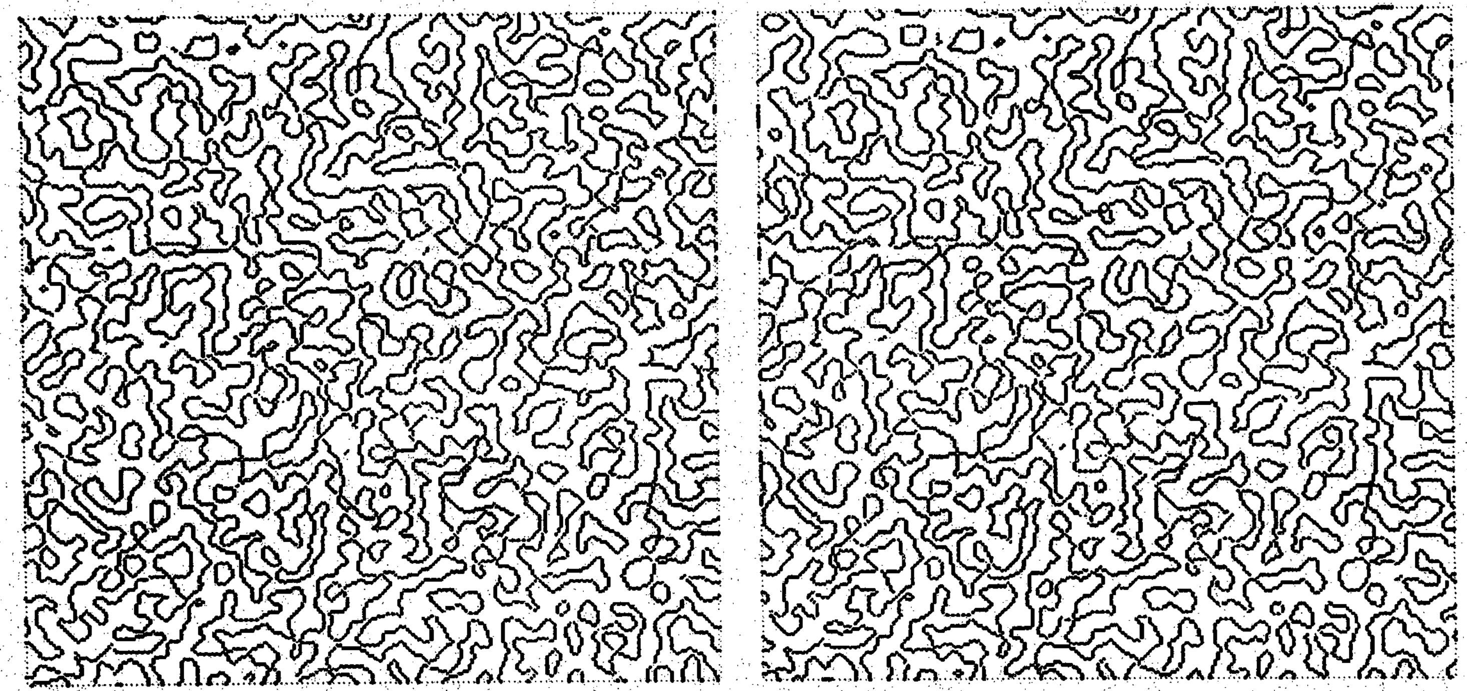


(b) Right image.



(c) Observed surace and viewpoints.

Figure 43 A step scene with unstructured light.



(a) Left image.

(b) Right image.

Figure 44 Zero crossings of the step scene ($W_2 = 8$).

shaped line features are present, in comparison with the time that it takes without the box outline. They argued that the structure of the box shaped outlines generated figural continuity cues that facilitated the process of stereoscopic fusion.

To quantify their results, Mahew and Frisby measured the ratios G/M in their experiments, where M is the number of zero-crossings on a zero-crossing contour in the left image and where M+G is the total number of right image candidate zero crossings within the respective search windows. G is referred to as the number of ghost zero-crossings and, ideally, we would want G to be zero. Clearly, G/M is a measure of the possibility of a false match.

Mahew and Frisby showed that in a particular experiment, G/M took a value of 2.17 when matching was accomplished on a raster line by raster line basis -- meaning that for each zero-crossing in a given raster line in the left image the search window was constructed on the same raster line in the right image -- provided no figural continuity information was used. On the other hand, with figural continuity enforced, the G/M ratio decreased to 0.317. For this particular result, for each left image zero crossing the figural continuity was enforced over 5 raster lines by comparing the horizontal coordinates of the contour segments associated with the left image zero crossing and a candidate right image zero crossing, and insisting that they be within $\pm$ 1. Further reductions in ghost matches could be obtained with figural continuity enforced over longer segments of zero-crossing contours; for example, instead of using five raster lines, we could use seven or nine.

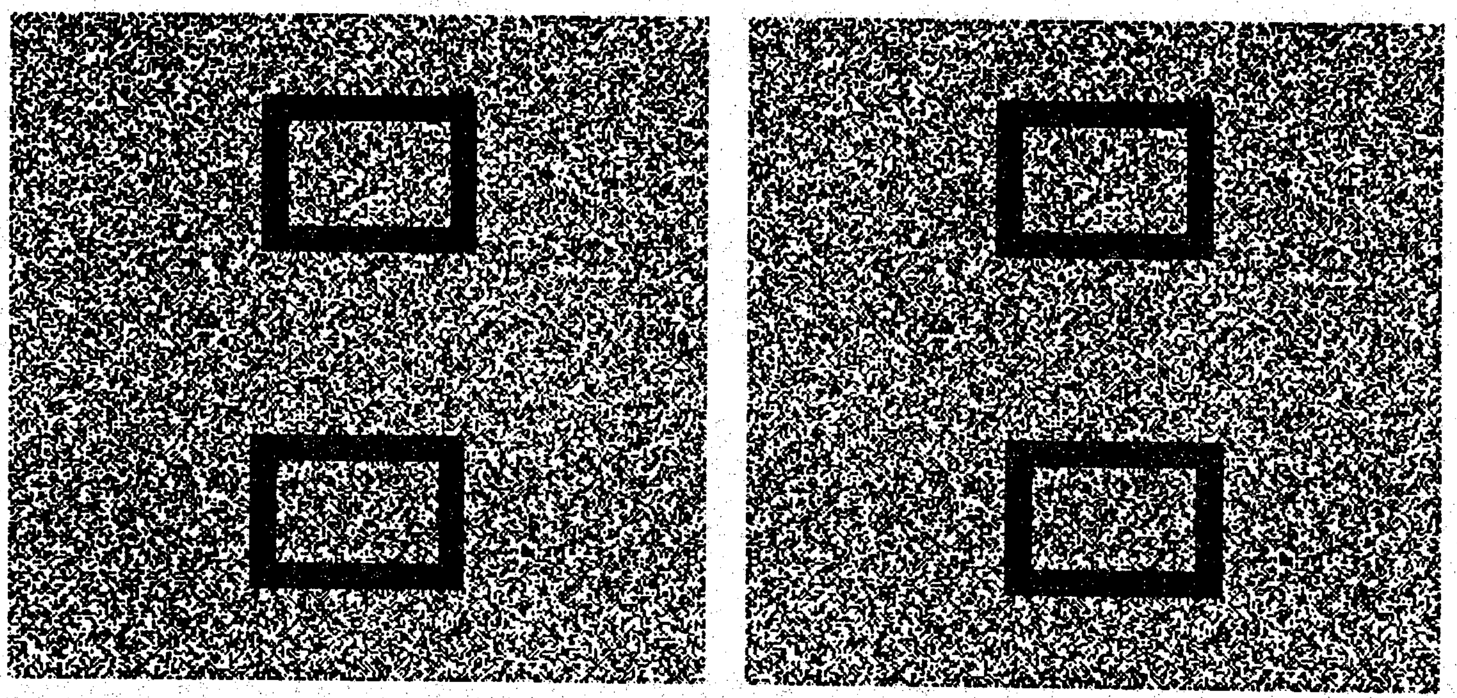
4.3.3. Grimson's Implementation of Figural Continuity

Grimson [12] implemented the concept of figural continuity such that a point P on a zero-crossing contour C in the left image was accepted as a reasonable candidate to be matched if all points within a certain distance from P on C succeeded in finding correspondences in the right image. The outline of Grimson's implementation is as follows. Suppose a point $P = (x_0, y_0)$, on a zero-crossing contour in the $N \times N$ left image has a corresponding point in the right image. Now, we examine whether this point satisfies the figural continuity constraint.

Let us consider a zero-crossing contour segment C, of length Λ whose components are (x_0,y_0), (x_1,y_1), (x_2,y_2), ... $(x_{\Lambda-1},y_{\Lambda-1})$.

Let L(x,y) be a $N \times N$ array which represents the zero crossing of the left image such that

$$L(x,y) = 1 \qquad \text{if a pixel } (x,y) \text{ is on a positive contour}$$



(a) Left image. (b) Right image.

Figure 45 A pair of synthetic stereo images with monocularly-discriminable cues.

$L(x,y) = -1$ if a pixel (x,y) is on a negative contour

$L(x,y) = 0$ if a pixel (x,y) is not on any contour

Let MT be a $N \times N$ matrix, whose element MT(x,y) is 1 bit binary number and is determined such that

$MT(x,y) = 1$ if L(x,y) is non-zero and there exists a corresponding zero-crossing in the search window.

$= 0$ if either L (x,y) is zero or there exist no corresponding zero-crossings in the search window.

Now a predetermined constraint K, which determine the sufficiency of figural continuity, is introduced. In this example, it is assumed that Λ is longer than K. (In other words, a concerned contour segment is long enough for examining figural continuity.) Note that for the simplicity, it is assumed that any contour point can be classified either as positive or as negative.

Then the algorithm can be described as follows.

```
done ← 0;
i ← 1;
g ← 0;
l ← 0;

while ( done = 0  or i < Λ - 1 ) do
  begin
   if ( L ( $x_i$ , $y_i$ ) = 1 or L ( $x_i$ , $y_i$ ) = -1 or l < K)
     l = l + 1;
   else
     begin
      call gap_processing (l,g,success)
      if (success = "yes")
        begin
         record this contour in the accepted contour map;
         done = 1;
        end
        else if ( success = "no" )
         done = 1;
     end
```

```
end
```

In this algorithm, a function gap_processing (l, g, success) return the answer as the variable success whether the contour, with length l and gap g, thus far examined is acceptable candidate or not.

```
success = "yes"   if this contour is accepted as the candidate
success = "no"    if this contour is accepted as the candidate
success = "  "    if further examination is necessary
```

4.3.4. Our Implementation of Matching by Figural Continuity

ZERO-CROSSING FIGURE MATCHING ALGORITHM

The algorithm employed for the zero-crossing figure matching is very similar to the straight line matching algorithm shown previously. In this section the algorithm of the zero-crossing figure match is presented. It should be noted that the length of the contour segment is 15 in this algorithm, while it is 41 in the case of the straight line matching.

STEP-1: Each zero-crossing image is represented by a data structure which is a list of zero-crossing segments described as

$$\left\{ \text{segment}_1, \text{segment}_2, \text{segment}_3, \ldots\text{segment}_M \right\}.$$

STEP-2: Each zero-crossing segment is represented by the following data structure

{ start-pix-col-indx, start-pix-row-indx, chain-code,
 start-pix-col-indx-of-conj, start-pix-row-indx-of-conj }

where the last two items are instantiated only after a right image contour segment is found that matches the left image contour segment with starting indices (start-pix-col-indx, start-pix-row-indx). The matching contour segment may be called the conjugate segment, hence the symbol 'conj.' In the event a left image segment matches more than one right image segment, the last two items can be replicated and instantiated with the corresponding starting values. Information about multiple matches this stored can be used subsequently when relational constraints are invoked. Each chain code is represented in the same manner as in straight line representation. For the discussion to follow, a contour segment from the left image will be denoted by

$$(L(1),L(2),....,L(N))$$

and a segment from the right image by

$$(R(1),R(2),....,R(N))$$

STEP-3: In comparing the data structures for two segments from the two images, the algorithm first makes a check for the positional correspondence. The positional correspondence is considered as sufficient if the following condition is satisfied :

$$\left| \text{start-pixel-col-index}_l - \text{start-pixel-col-index}_r \right| \leq D_{max}$$

and

$$\left| \text{start-pixel-row-index}_l - \text{start-pixel-row-index}_r \right| \leq 1,$$

where D_{max} is the range of the maximum permissible value for the disparity.

STEP-4: A similarity score, denoted by T, is computed for the two segments by comparing their chain codes. Initially, the value of T is set to 0. At step i, let the chain code elements from the two segments be L(i) and R(i). For each i, i = 1,....,N, the total score T is accumulated by using

if L(i) = R(i) then T = T + 1

Note that the weight value W used in the straight line matcher is not employed here. This is because the alternation of two codes discussed in the straight line matcher is unlikely to occur in the case of zero-crossing contour segments extracted from random dot patterns used at this stage of our algorithm.

STEP-5: If the similarity score T does not exceed a pre-set threshold, denoted here by T5, delete the left image segment under consideration from the left image data structure created by STEP 1.

STEP-6: If the similarity score T exceeds a threshold, T5, declare the two segments as matchable. In the current computational model T5 and N are set to 11 and 15, respectively. Delete from the left image data structure shown in STEP 1 those segments whose (start-pixel-col-index, start-pixel-row-index) lie on the matched segment in the left image.

STEP-7: If the left image data structure created in STEP 1 is non-nil, go to STEP 3.

STEP-8: In this step, range values to the scene points that lie on the matched segments are computed by the following procedure: Let D(i), i=0,1,..,N be the disparity computed from the ith pixels in the left and the right image segments; as defined in STEP-2, the chain codes of these matching pixels are denoted by L(i) and R(i), respectively. The disparities, D(i), is obtained from the chain codes L(i) and R(i) by the following formula in which A, A1, A2 are temporary variables:

```
D (0) ← CL - CR
      where CL = start-pixel-col-index for the first pixel
                   in the left image segment
                CR = start-pixel-row-index for the first pixel
                   in the right image segment

For i ← 1 step 1 until N do
begin
  if ( L (i) = (0, 1, or 7) )
A1 = 1;
  else if ( L (i) = (2, or 6) )
A1 = 0;
  else if ( L (i) = (3, 4, or 5) )
A1 = -1;
  if ( R (i) = (0, 1, or 7) )
A2 = 1;
  else if ( R (i) = (2, or 6) )
A2 = 0;
  else if ( R (i) = (3, 4, or 5) )
A2 = -1;

  A = A1 - A2;
  D (i) = D (i -1) + A;
end
```

The range maps down-sampled to 64 $\times$ 64, similar to the straight line extraction algorithm.

In Figure 46, the contour segments, which are successfully matched by our zero-crossing figure matching for the pair image shown in Figure 44, are presented.

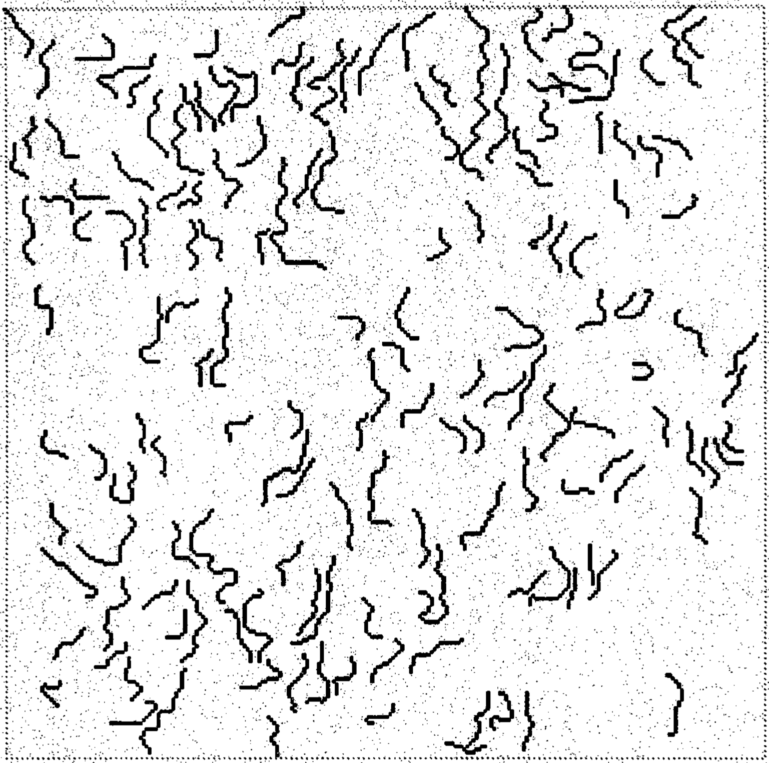


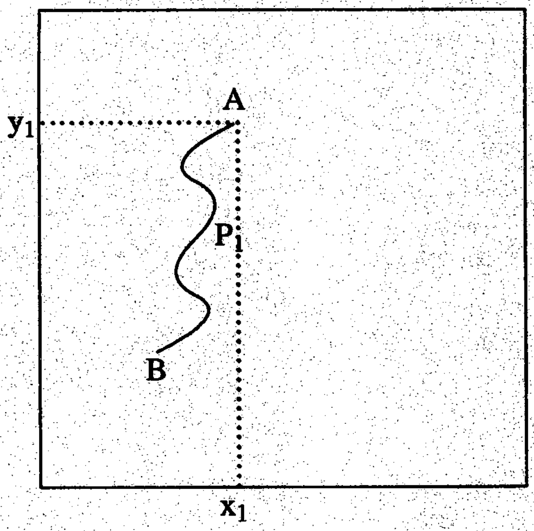
Figure 46 Matched zero crossing contour segments of Figure 44.

4.3.5. Matching of Individual Zero-Crossings

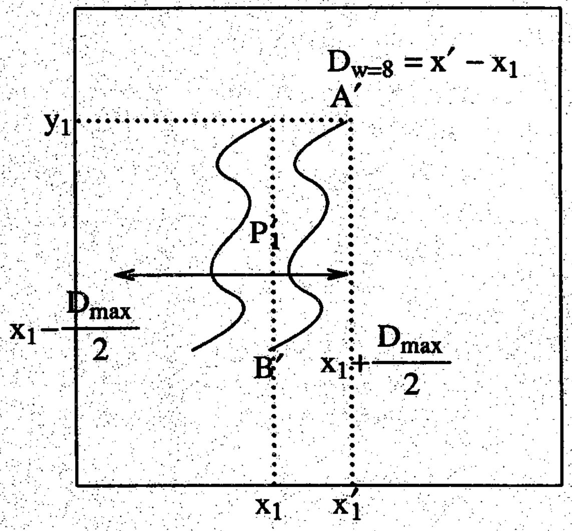
As described before, the matching process of this section is carried out at two levels, the contour segment level (the first level) in the $w_{2D}=8$ channel and the pixel level (the second level) matching of zero-crossings in the $w_{2D}=4$ channel. With the help of Figure 47, we will illustrate how the two levels act in concert. In Figure 47, a pair of the contour images of the first level are shown in (a) and (b), and the second level image pair are shown in (c) and (d). Contour segment AB in the left image is matched with the contour segment A'B' from the right image by the figural continuity constraint on the first level. Note that when search for the matching contour segment for AB is conducted we make sure that the horizontal coordinate of A' is within the horizontal coordinate of A $\pm\frac{D_{max}}{2}$, as shown in Fig. 47. This match is used to compute a disparity map along each matched contour and the resulting disparities stored in a 64×64 matrix. Subsequently, the zero-crossings from the $w_{2D}=4$ channel are matched individually, as in any channel of the MPG algorithm. For example, if P_2 is a zero-crossing produced by the $w_{2D}=4$ channel with its x-coordinate equal to x_1, then to find its correspondent we first construct a search window of size $\pm\frac{w_{2D}}{2}$ around the point x_1+d, where d is the disparity value pulled out of the 64×64 disparity matrix mon-tioned earlier.

The reader should note that the search window has half the size of those used in the MPG algorithms. Whereas each channel of the MPG algorithm uses a search window of $\pm w_{2D}$, the search window used here is only half that. As was pointed out in Section 2.2.3, theoretical considerations dictate that in MPG algorithm the size of the search window should be $\pm\frac{w_{2D}}{2}$, however when such algorithms are implemented the search windows are set to $\pm w_{2D}$ to account for the shifts in zero crossing contours caused by the interaction of different intensity variations in an image. To add to the discussion of Section 2.2.3, this interaction will be illustrated with the help of a one-dimensional gray level variation shown in Figure 48 (a). This pattern can be parametrized by the variables A and B, which control the relative heights of two of the levels in the pattern, and C which controls the separation of the spike from the other variation. Figure 48 (b) shows the boundaries of the shifts in the zero-crossings for different normalized values of the parameters. For example, when the normalized value of separation is given by $\frac{C}{w_{2D}}=0.5$ and the value of B/A is 0.06, the shift of the zero-crossing corresponding to the edge X is given by point P.

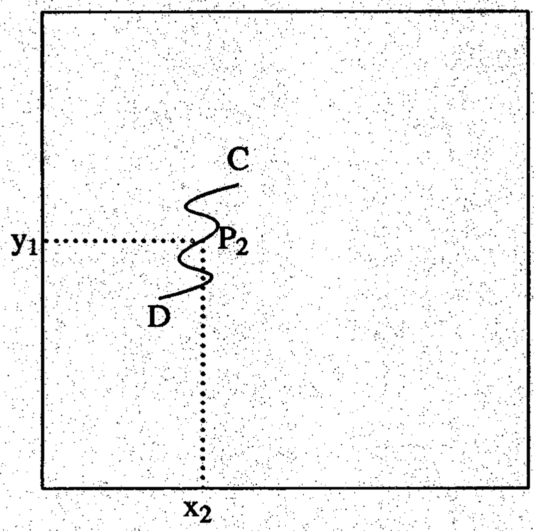
We believe that if the parameters of gray level variations in an image are such as to cause zero-crossing shifts comparable to, say, w_{2D}, then the resulting zero-crossing



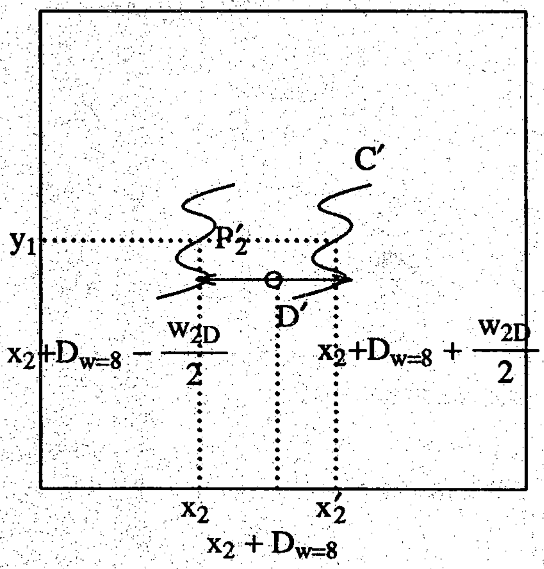
(a) **Left contour image ($w_{2D} = 8$).**



(b) **Right contour image ($w_{2D} = 8$).**



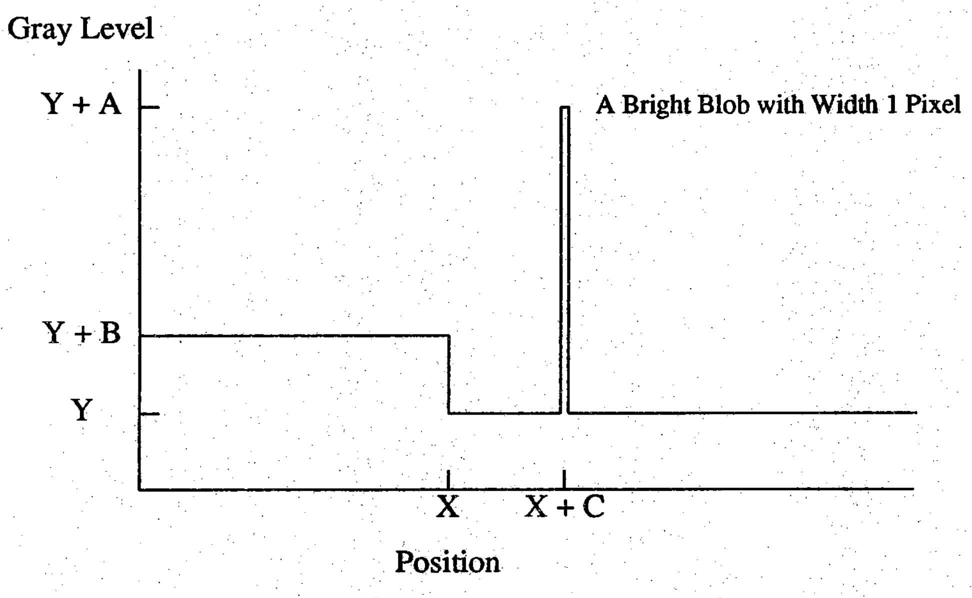
(c) **Left contour image ($w_{2D} = 4$).**


C′
y_1
P_2'
D′
$x_2 + D_{w=8} - \frac{w_{2D}}{2}$
$x_2 + D_{w=8} + \frac{w_{2D}}{2}$
x_2
x_2'
$x_2 + D_{w=8}$

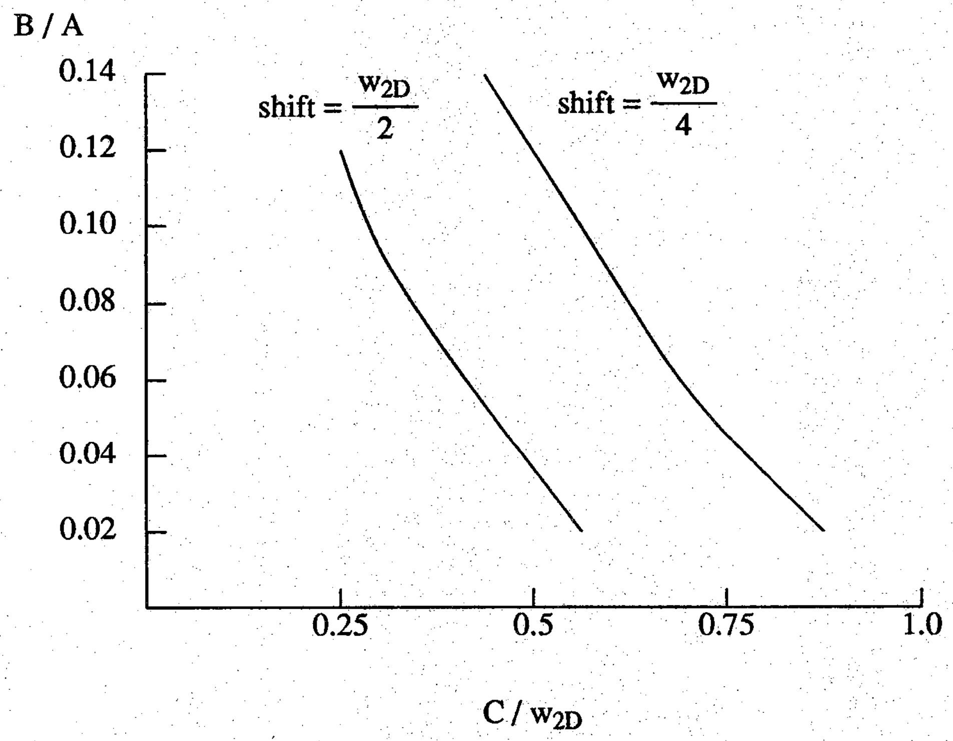
(d) **Right contour image ($w_{2D} = 4$).**

Note : The zero-crossing figure match is applied only to the fusion of the images through the channel with the size $w_{2D} = 8$ ((a) and (b) in the illustration shown above). The default matcher is applied to the fusion of the finest channel images ((c) and (d) in the illustration shown above).

Figure 47 Concept of the zero-crossing figure match.



(a) **1-D Synthetic Image**



(b) **Boundaries of Misleading Matching**

Fig. 48 Simulation about the zero-crossing shift by a small blob.

contours would be unmatchable by the figural continuity matcher. If this conjecture is true, then the matching of zero-crossing contour segments should provide us with an accurate disparity information wherever such a match can be accomplished. The point we are trying to make is that after matching contour segments, the resulting disparity information is quite robust and there is less of a reason to use $\pm w_{2D}$ windows for subsequent matching. Since smaller windows lead to fewer disambiguation problems, we chose the $\pm\frac{w_{2D}}{2}$ windows.

The point being made is that if the similarities between contour segments are such that they can be matched, then it is unlikely that there would any significant deviations in the zero-crossings at the next finer channel. After all, contour wise similarity arises because of area wide similarity between the two images. It is hard to conceive of a case where the photometric variances are such that there would be similarities at the $w_{2D}=8$ channel for the contour matching to occur, yet there would be significant deviations at the $w_{2D}=4$ level to require a search window larger than dictated by theory.

4.4. The Default Matcher

When all else fails, the systems invokes the full implementation of the MPG algorithm; so, in a sense, this is the default matcher. We have already discussed this algorithm in Section 2.2.

CHAPTER V A REVIEW OF SOME IMPORTANT RULES

5.1. Overview of the Rule-Based Procedure

Unless properly organized, a rule-based system can quickly outgrow the ability of a researcher to keep the entire system in mental perspective. Each rule usually embodies a small measure of control information; and, in a system with a large number of rules the interactions between the rules can be sufficiently overwhelming to the point that it may not be possible to feel confident about the robustness, stability and the convergence of the reasoning process. For these reasons, we have grouped the rules in our system in two different ways; the first type of grouping is simply to facilitate the comprehension of the system by a human, and the second type for the flow of control in the computer program. The first type of rule categorization is referred to as Groups and the second type Stages.

The three Groups of rules in the system can be described in the following manner:

GROUP-1

These rules are used for dynamically altering the entries of the control matrix. Note that the initial entries are generated by a deterministic procedure on the basis of the output of straight line matching. An important function of these rules is to generate the control matrix entries in such a manner that the straight lines extracted from the right image satisfy the relational constraints on their correspondents in the left image. This group also includes rules for invoking different possible surface-orientation hypotheses for the geometrically constrained matcher. A rule-based implementation here allows the incorporation of heuristics like: Of the many possible choices for surface orientations, we should first choose one that is the same as the orientation of one the neighboring surfaces; if that does not work then choose an orientation that is closest to that of one of neighboring surfaces. Of course, if no orientations are known in the immediate neighborhoods, then choose at random a permissible orientation hypothesis.

GROUP-2

Rules for determining the search window size for the implementation of the finest MPG channel. These rules make sure that if the MPG matching is being carried out without a prior geometrically constrained matching, then the search window size is set equal to $2 \times W_{2D}$. On the other hand, if the finest channel is being invoked as a follow-up on the geometrically constrained matcher, then these rules set the search window size to W_{2D}.

GROUP-3

These are metarules for coordinating the flow of control. These rules make sure that the alteration of an entry of the control matrix is followed up by the invocation of the appropriate matching strategy.

The Stages represent another categorization of the same rules, as shown in Figure 49. As is made clear by the following description, the Stages play an important role in the organization of the flow of control.

Stage-1	In this stage, either the geometrically constrained match or the zero-crossing figure match is executed, depending upon the entry in the control matrix. This stage allows the generation of multiple hypotheses for both the surface orientations and the candidates for zero-crossing figure mathing. In the current implementation, although multiple candidates for figure matching are permitted, only a single hypothesis is retained for surface orientation.
Stage-2	In this stage, the best hypothesis or candidate is selected from amongst those generated in the previous stage. The hypothesis or candidate selection strategy includes examination of the valid hypotheses/candidates in neighboring regions.
Stage-3	The default matcher is called in this stage.

Hypothesis Arrays

In general, in geometrically constrained matching it will be possible for multiple surface orientation hypotheses to meet the hypothesis selection criteria. By the same token, in zero-crossing figure matching, for each zero-crossing contour in the left image there will be multiple candidates from the right image. The multiple orientation

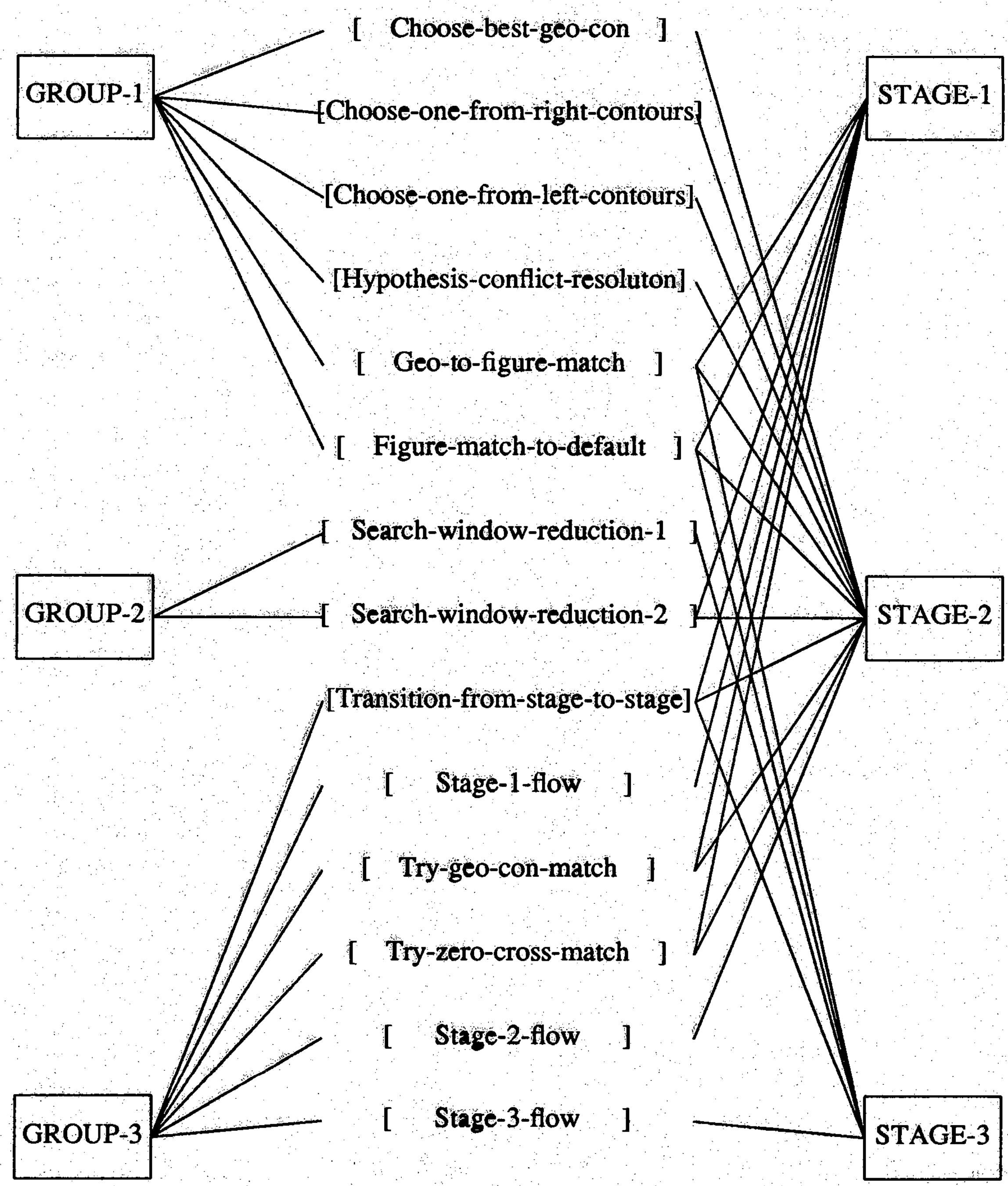

GROUP-1
GROUP-2
GROUP-3
[Choose-best-geo-con]
[Choose-one-from-right-contours]
[Choose-one-from-left-contours]
[Hypothesis-conflict-resoluton]
[Geo-to-figure-match]
[Figure-match-to-default]
[Search-window-reduction-1]
[Search-window-reduction-2]
[Transition-from-stage-to-stage]
[Stage-1-flow]
[Try-geo-con-match]
[Try-zero-cross-match]
[Stage-2-flow]
[Stage-3-flow]
STAGE-1
STAGE-2
STAGE-3

Figure 49 Two types of classification of rules.

hypotheses and zero-crossing contour candidates are stored, respectively, two arrays called Θ and Z.

As was explained in Section 6.2, the 256×256 is divided into an array of 64×64 patches, each of size 4×4 pixels, for the construction of the control matrix and the calculation of disparities. Also, as explained there, each orientation hypothesis is computed over a an array of 4×4 patches -- meaning 16×16 pixels -- and, as can be seen from Figure 50, these patch arrays are overlapped. Therefore, the size of Θ matrix is 61 × 61 to represent orientations at all possible 16×16 pixel subarrays. Each entry in the Θ matrix is a list of all possible hypotheses for surface orientations at the corresponding 16×16 subarray of pixels; therefore, each such entry can be represented as the following list

$$\Theta(x,y) = \left\{ \text{cardinarity\_of\_entry}, \ \theta_1, \theta_2, \ldots\ldots, \theta_K, \right\},$$

For example, if we have three hypotheses for a 16×16 subarray of pixels, $\Theta(x,y) = \{3, \theta_1, \theta_2, \theta_3\}$. When the i th hypothesis is tried and found to be unacceptable, it is removed from the list. In the discussion to follow, we will use the notation $\Theta(x,y,i)$ to describe the i-th hypothesis from the list at $\Theta(x,y)$.

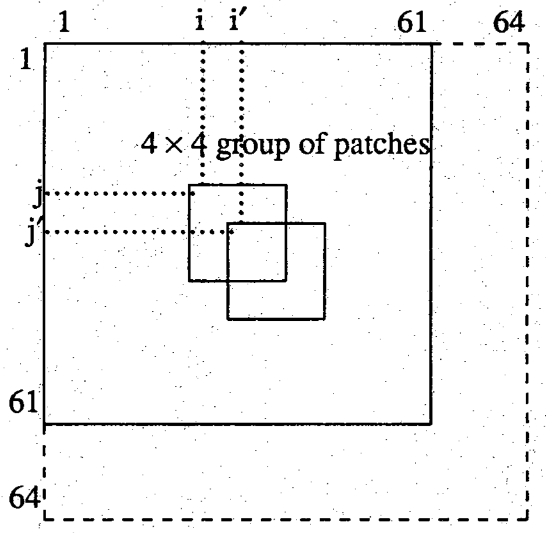
From the discussion in Section 6.3, it follows that the size of Z should be 64×64. Each entry in the Z matrix is again a list of possible candidates and can be represented by the following form: As was illustrated in Fig. 50, the right-image candidate contour is stored as a list of disparities at the pixels of the left-image contour. Each entry in the Z matrix therefore looks like

$$Z(x,y) = \left\{ \text{cardinarity\_of\_entry}, \ z_1, z_2, \ldots\ldots \right\},$$

where z_i is the list of disparity values computed by the zero-crossing figure matcher for the i-th candidate match.

Explanation on the rule description in the if-then form

We will now briefly describe the syntax we will be using for presenting some of the rules in this chapter. The rules will be shown with the help of IF-THEN forms. To improve the readability of presentation, some part of each rule may be presented as an English sentence. The following points are to be noted for the algorithmic parts of the rules: The reader should bear in mind the difference between the variables and constants, and also the differences between a C program function and a logical predicate. The left hand side (LHS) of a rule can have both mathematical expressions and logical predicates; this being consistent with the syntax of OPS83. The following is a summary of the notation used for describing the rules:



Θ (i,j) : a list with N+1 elements

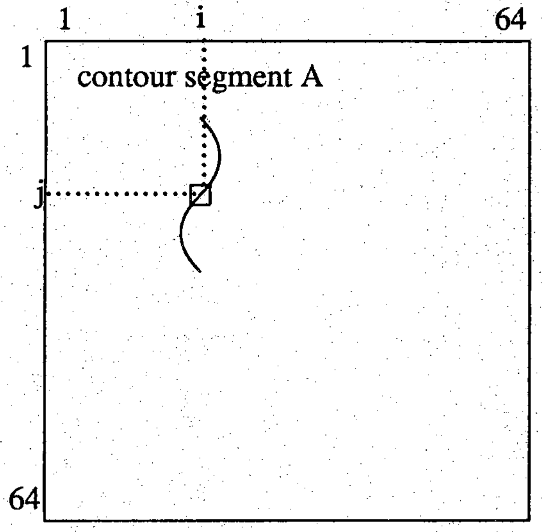
$\{ N, \theta_1, \theta_2, \cdots, \theta_N \}$

N : The number of accepted hypothesis

θ_i : The orientation of the i th hypothesis

(The coordinates of Θ, (i,j), represent the patch spans within the range (i,i+3) horizontally and (j,j+3) vertically.)

(a) Hypothesis array Θ



Z (i,j) : A list with M+1 elements

$\{ M, \theta_1, \theta_2, \cdots, \theta_M \}$

M : The number of disparity values computed by the zero crossing contour matching.

(The number M can be greater than 1 when more than two contours pass through the 4 × 4 patch specified by the coordinates (i,j).)

(b) Hypothesis array Z

Figure 50 Hypotheses Arrays Θ and Z.

(1) The asterisk (*) indicates that the word following the asterisk stands for the purposes of explanation as a "working memory element" in OPS-83. Such a word can be matched with the LHS of a rule, and if the match is successful, the RHS of the rule can be executed. In RHS of a rule, a working memory element is created, modified, or removed by "make", "modify" and "remove" commands, respectively.

(2) A variable is denoted by a word whose first letter is capitalized.

(3) A constant is denoted by a word with lower case letters. In the case of a string constant, it is denoted by the mark ' '.

(4) A procedure written as a C-Program is denoted by a name with upper case letters.

5.2. GROUP-1 Rules

These rules come into play when processing is in Stage-2. Therefore, these rules have the following two functions: 1) When for a given region of an image, as represented by an element of the control matrix, more than one hypothesis for the local planar surface orientation is found to be applicable, one of these rules must select the best of those by examining the hypotheses in the neighboring regions. 2) When for an arc of a zero-crossing contour in one of the images, the figural matching procedure yields more than one possible match from the other image, these rules must again select one on the basis of such decisions in the neighborhood.

[Rule Choose-best-geo-con]
IF two or more geometrical constraints are satisfied at a point (x_0,x_0) *in the control matrix, THEN choose one by taking the neighboring hypotheses into account.*

```
IF ( Stage = 2
    and C($x_0$,$x_0$) = 2
    and $\Theta(x_0, y_0, 1) > 1$  [ more than two entries] )
THEN
    {
    CHOOSE_MOST_LIKELY_THETA ($x_0$, $y_0$, $\Theta$)

    make *stage_2_rule_worked ( status = 'yes' )
    }
```

where CHOOSE_MOST_LIKELY_THETA is a function defined in the last section of this chapter. This function polls the neighboring regions as given by the neighborhoods defined by the Θ matrix.

[Rule Choose-one-from-right-contours]
IF there are two or more candidates for the zero crossing figure match, as evidenced by the existence of multiple entries for an element of matrix Z, THEN choose one by polling the 8-neighbors using the Z-matrix neighborhoods. This situation will arise when an arc in the left image has more than one possible match in the right image.

```
IF ( Stage = 2
    and $C(x_0,y_0) = 3$
    and $Z(y_0,y_0,1) > 1$   [ two or more entries in Z] )
THEN
    {
    CHOOSE_MOST_LIKELY_Z ($x_0$, $y_0$, Z)

    make *stage_2_rule_worked ( status = 'yes' )
    }
```

where the function CHOOSE_MOST_LIKELY_Z is defined in the last section. Basically, this function chooses a single disparity value from amongst the multiple choices available by taking into account those neighborhood disparities that are known unambiguously. Again, the neighborhoods used are those corresponding to the Z matrix.

[Rule Choose-one-from-left-contours]
IF two or more arcs from the left image have the same correspondent from the right image, THEN discard all but one of the matches by using the relational information with neighboring digital arcs; the relational information is being used only in the positional sense defined below.

In our model, only the horizontal relations corresponding to positional differences between the arcs are taken into account. (A study of stereopsis where more elaborate relations are utilized in stereopsis is presented in [34].) To illustrate what we mean by relational information generated by positional differences, consider the arcs A_0 in Figure 51 which can be matched with the digital arc A′. Let's say that the arc labeled A can also me matched with A' and that the neighboring arc B has only a single correspondent in B'. We will use REL(A, B) to denote the positional relation between two contour segments in an image. For the arc positions shown in Figure 51, we may write

$REL(A_0, B) = x_0 - x_2$
$REL(A, B) = x_1 - x_2$
$REL(A', B') = x' - x'_2$

Note that these relational values can easily be computed from the information that resides in the contour segment list. The rule under discussion will select either A or A_0 for matching with A' by using the relational values in the manner discussed below.

```
IF ( Stage = 2
    and right digital A′ has
    two left digital arcs, A and A₀,
    possibly to be matched )
THEN
    {
    Consider a nearest stereo pair B, and B′
    Compute
      REL(A₀, B)
      REL (A, B)
      REL (A′, B′)
    if ( | REL(A₀, B) - REL (A′, B′) | > | REL (A, B) - REL (A′, B′) | )
      {
      take A as the correct match
      CHANGE_C_MATRIX ( A₀ );
      }
    else
      {
      take A₀ as the correct match
      CHANGE_C_MATRIX ( A );
      }

    make *stage_2_rule_worked ( status = 'yes' )
    }
```

The control matrix entries for elements corresponding to the regions through which the discarded left image arc traverses are changed to 4.

[Rule Hypothesis-conflict-resolution]
IF a planar-surface orientation hypothesis is not in agreement with orientation hypotheses at any of neighbors in the THEATA *matrix whose images regions overlap with the region under consideration, THEN discard the hypothesis.* A most important implication of this rule is that geometrically constrained matching will not be used in

the immediate vicinity of corners formed by scene surfaces. Unfortunately, this has to be the case since orientation hypotheses are enforced at the same time over large image blocks -- 16x16 to be precise -- and so geometrically constrained matching would lead to erroneous matches in the vicinity of junctions between surfaces of different orientations. Therefore, in the immediate vicinity of surface junctions, the system must rely on other matchers.

```
IF ( Stage = 2
    and $C(x_0,y_0) = 2$
    and $\Theta(x_0,y_0,1)$
    and $\Theta(x_0,y_0,2) \neq \Theta(x_1,y_1,2)$ )
    [ where $x_0 \leq x_1 < x_0 + 4$, and $y_0 \leq y_1 < y_0 + 4$ ]
THEN
    {
    discard  $\Theta(x_0,y_0,2)$ and $\Theta(x_1,y_1,2)$
    make *alter_C_request ( $X = x_0$, $Y = y_0$)
    make *alter_C_request ( $X = x_1$, $Y = y_1$)
    make *stage_2_rule_worked ( status = 'yes' )
    }
```

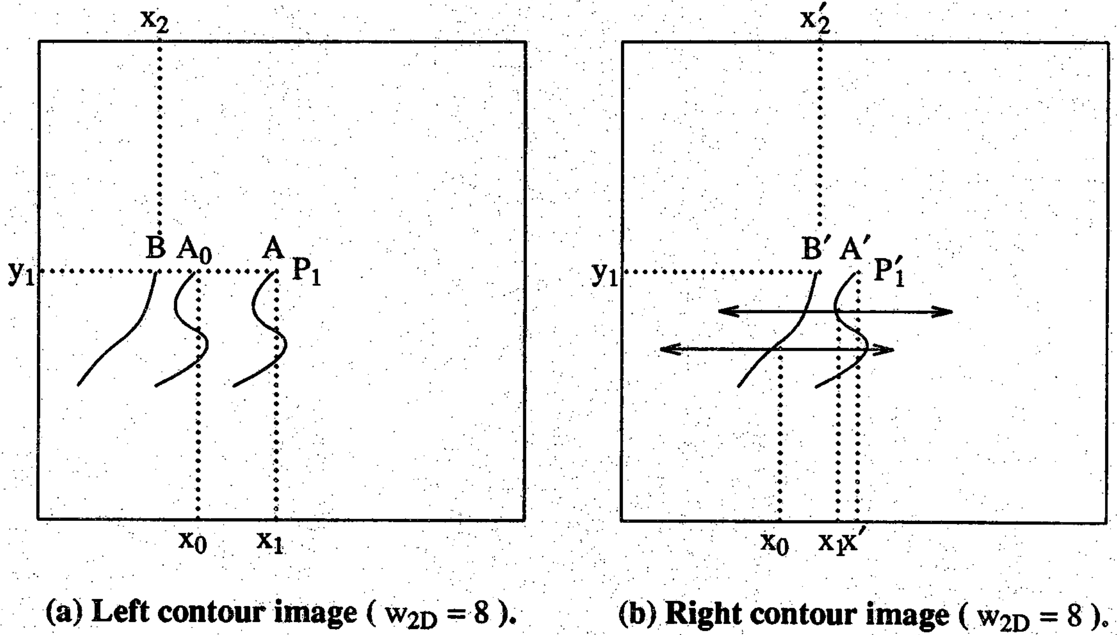
[Rule Init-C-matrix-set]
IF the OPS83 program is initiated, THEN the control matrix must first be initialized.

```
IF ( Stage = 1
    *start )
THEN
    {
    CREATE_INIT_C_MATRIX (C);
    remove *start
    In_stage_processing_done = 'no' )
    }
```

The control matrix is initialized on the basis of dominant feature matching; this is done by the function CREATE_INIT_C_MATRIX defined at the end of this chapter.

[Rule Geo-to-figure-match]
IF geometrically constrained matching is not successful, THEN the zero-crossing figure matching is to be invoked.

[Rule Figure-match-to-default]



x_2
y_1
B
A_0
A
P_1
x_0
x_1
(a) Left contour image ($w_{2D} = 8$).
x_2'
y_1
B′
A′
P_1'
x_0
x_1
x'
(b) Right contour image ($w_{2D} = 8$).

Fig. 51 Relational constraints of the zero-crossing figure match.

IF neither geometrically constrained matching nor the zero-crossing figure matching is successful, THEN the default matching algorithm should be applied.

```
IF (*alter_C_request ( X = $x_0$, Y = $y_0$) )
THEN
   {
   if (C ($x_0$,$y_0$) = 2)
      then C ($x_0$,$y_0$) $\leftarrow$ 3;
   else if (C ($x_0$,$y_0$) = 3)
      then C ($x_0$,$y_0$) $\leftarrow$ 4;

   remove *alter_C_request ( X = $x_0$, Y = $y_0$)
   }
```

5.3. GROUP-2 Rules

The rules in this category control the size of the search window in the finest MPG channel. Note that this is the only channel that is invoked after zero-crossing figure matching. Also, note that when we establish a planar surface orientation in a region of the image, verification of this orientation is done by matching zero-crossings by using a single MPG channel. The size of the search windows used in this channel is also controlled by the rules in this group.
[Rule Search-window-reduction-1]
IF the zero-crossing figure match is successful, THEN reduce the search window to half of the finest channel in the default MPG matcher.

```
IF ( Stage = 3
   and W value is unspecified
   and Cs = 4;
   and C( $x_0$ , $y_0$ ) = 3 )
THEN
   W $\leftarrow$ 2.
```

Note that the search window is specified by $\pm W$ and that Cs is the value of the variable w_{2D} in a MPG channel. The LHS condition that Cs=4 means that only the finest channel of the MPG algorithm would be affected.

[Rule Search-window-reduction-2]

IF in geometrically constrained matching a planar surface orientation hypothesis is under verification THEN the search window is made very small and corresponds to a small disparity interval around that corresponding to the hypothesized plane. on both sides of the hypothetical plane.

```
IF ( Stage = 1
    and  W value is unassigned
    $C( x_0, y_0 ) = 2)$
THEN
    W ← 1;
```

5.4. GROUP-3 Rules

The GROUP-3 rules are created for coordinating the flow of control. These rules make sure that the alteration of an entry of the control matrix is followed up by the invocation of an appropriate matching strategy.

5.4.1. Control of Transition between Stages

[Rule Transition-from-stage-to-stage]
IF stage 1 has terminated, THEN proceed to stage 2. IF stage 2 has terminated, THEN proceed to stage 3.

This is a serial transition of stages from 1 to 2, and then from 2 to 3.

```
IF (  Stage = 1
    and In_stage_processing_done = 'yes' )
THEN
    {
    Stage = 2
    In_stage_processing_done = 'no'
    make *stage_2_rule_worked ( status = 'yes' )
    }

IF (  Stage = 2
    and In_stage_processing_done = 'yes' )
```

```
THEN
    {
    Stage = 3
    In_stage_processing_done = 'no'
    }

IF ( Stage = 3
    and In_stage_processing_done = 'yes' )
THEN
    {
    End of the processing
    }
```

5.4.2. Control of Stage 1

[Rule Stage-1-flow]
IF processing is in stage 1, THEN a matcher, which can either be the geometrical constrained matching or the zero-crossing figure matching, is sought from left to right and from top to bottom.

The rule is applied starting at the coordinates (0,0) in the control matrix. Let (x_0, y_0) denote the current working element in the control matrix. The function INCREMENT_COORDINATE computes the next control matrix element to be considered.

```
        IF ( Stage = 1
    X= x0, Y= y0
          and Done = 'no' )
        THEN
          {
          make *matcher_select(x0 , y0);
          INCREMENT_COORDINATE (x0, y0, Done);
          }
```

where x_0 and y_0 stand for the coordinates in the control matrix and the working memory element *matcher_select will match with all the WME's that are capable of triggering strategy selection rules. In this case, the strategy selection rules will be fired for the location (x_0, y_0) in the control matrix.

```
          IF ( Stage = 1
      X= $x_0$, Y=$y_0$
            and Done = 'yes' )
          THEN
            In-stage-processing-done($x_0$, $y_0$) $\leftarrow$ 'yes'
```

As will be shown presently, the variable In-stage-processing-done is again set to 'no' if stage 2 is invoked for the same location in the control matrix.

```
[ Rule Try-geo-con-match ]
          IF (  *matcher-select
      X =$x_0$, Y = $y_0$
            C( $x_0$, $y_0$ = 2 ) )
          THEN
            {
            TRY_GEO_HYPOTHESIS ( X, Y, W, D, $\theta$, Success)
            if (success $\neq$ 'yes' )
            make *alter_C_request ( X = $x_0$, Y = $y_0$)
            remove *matcher_select ( X =$x_0$, Y = $y_0$)
            }

[ Rule Try-zero-cross-match ]

          IF (  *matcher_select
      X =$x_0$, Y = $y_0$
            C( $x_0$, $y_0$ = 3 ) )
          THEN
            {
            TRY_FIGURE_MATCH (X, Y, Z, Success )
            if (Success $\neq$ 'yes' )
            make *alter_C_request ( X = $x_0$, Y = $y_0$)
            remove *matcher_select ( X =$x_0$, Y = $y_0$)
            }
```

5.4.3. Control of Stage 2

[Rule Stage-2-flow]
IF stage = 2, THEN the Stage-2 rules are to be made available for firing as long as the LHS's of any of them can be satisfied.

```
IF ( Stage = 2
    and In_stage_processing_done = 'no'
    and *stage_2_rule_worked (status = 'yes' )
    )
THEN
    {
    fire rules;
    }

IF ( Stage = 2
    and In_stage_processing_done = 'no'
    and *stage_2_rule_worked (status = 'no' )
    )
THEN
    {
    In_stage_processing_done ← 'yes'
    CREATE_DISPARITY_MAP
    }
```

5.4.4. Control of Stage 3

[Rule Stage-3-flow]
IF the stage is 3, THEN the default matcher is called, the calling order being from left to right and from top to bottom in the control matrix.

For all points (x_0, y_0) where the control matrix indicates the necessity of the default matcher (in other words $C(x_0, y_0) = 4$), the default matcher is invoked. For the coarsest channel, the value of vergence is set to zero, which is in keeping with our assumption that the optic axes of the cameras are nearly parallel. In the subsequent channels, for the points where the previous channels have not provided any disparities, the value of vergence is determined by interpolating the previous channel disparity map. The following is the flow of the computation in the default matcher.

```
IF ( Stage = 3
    and Done = 'no' )
THEN
    {
    make *default_match ($x_0$ , $y_0$);
    INCREMENT_COORDINATE ($x_0$, $y_0$, Done);
    }

IF (*default_match ( X = $x_0$, Y = $y_0$)
THEN
    DEFAULT_MATCH ($x_0$, $y_0$,Dn,W)
    remove *default_match ($x_0$ , $y_0$);
```

5.5. Functions Called by the Rules

We will now define the functions that were used in the rules.

(1) MATCH ---- Point-Wise Matcher

For a given zero-crossing in the left image, this function is invoked to find the matching zero-crossing in the right image by the MPG process.

MATCH (X,Y,D,W,Cs,Dn).

This function returns the disparity value Dn, computed at the position (X,Y) in the control matrix for the MPG channel with w_{2D} equal to Cs. D is the current disparity value generated by the previous channels and constitutes vergence for the current call. The number W is the search window size.

(2) INCREMENT_COORDINATE (X,Y, Done)

```
    {
            if ($X \neq 64$ )
              $X \leftarrow X + 1$;
            else
              if ($Y \neq 64$ )
                $Y \leftarrow Y + 1$;
                $X \leftarrow 1$;
              else
```

```
                Done ← 'yes'
            }
```

(3) CREATE_DISPARITY_MAP

```
    {
        if ( $C(x_0,y_0) = 1$ )
            $D(x_0,y_0) \leftarrow$ the value given by straight line matching
        if ( $C(x_0,y_0) = 2$ )
            $D(x_0,y_0) \leftarrow$ Dn of MATCH
        if ( $C(x_0,y_0) = 3$ )
            $D(x_0,y_0) \leftarrow Z(x_0,y_0)$
        }
```

(4) TRY_GEO_HYPOTHESIS (X, Y, W, D, θ, Success)
For a 4×4 planar patch (we will remind the reader that a patch is defined over the control matrix; therefore, a 4x4 planar patch translates into a 16x16 image block) with top left coordinates (X,Y), verify the geometrical constraint corresponding to planar scene surface orientation of angle θ. Since this function must call the function MATCH, the arguments X, Y, W and D must also be supplied. If the MATCH function succeeds in matching a sufficiently large number of zero-crossings, and TRY_GEO_HYPOTHESIS returns 'yes' for the variable Success. For each patch, the function TRY_GEO_HYPOTHESIS is called as many times as there are hypotheses for that patch in the Θ matrix. If none of the hypotheses can be verified, then a failure of geometrically constrained matching is declared for that patch.

D ← this is the disparity that must be valid in the patch if all the image points in the patch did indeed fall on the planar surface corresponding to the orientation hypothesis under verification.

W ← determined by the [Rule Search-window-reduction-2] explained earlier. The window size to be used in finding a match for a left image zero-crossing will be +—W.

Cs ← W_{2D}=4. The parameter Cs specifies which channel output will be used for hypothesis verification. Cs=4 means that the finest MPG channel zero-crossings will be used.

The function TRY_GEO_HYPOTESIS calls the MATCH function for each control matrix element in the patch -- for a total of 16 times since each patch is 4x4. During each such call, if MATCH is successful, the updated disparity value is computed for that element of the control matrix. Out of 16 such attempts for a patch, at least a fraction T4 of them must be successful for TRY_GEO_HYPOTHSIS to return 'yes' for the variable 'Success', where T4 is a planar-surface-orientation dependent threshold discussed in Section 4.2.3.

(5) TRY_FIGURE_MATCH (X, Y, Z , Success)
If at location (X, Y) in the control matrix, the entry is 3, then this function will be invoked to attempt a zero-crossing figure match. For this figure match, all 15-pixel contour segments will be considered whose beginning pixels are within the 4x4 block of the image represented by the location (X, Y) of the control matrix. The disparity values generated by figure matching are stored in the Z matrix. Finally, a value is returned for the variable 'Success' on the basis of

Success $\leftarrow$ 'yes' if there is at least one successful match
Success $\leftarrow$ 'no' if there is no successful match

(6) DEFAULT_MATCH (X, Y, Dn, W)

This function invokes the full MPG process when the control matrix entry at (X, Y) is 4. The MPG process is invoked by repeated calls to MATCH for different zero-crossing channel outputs. Note that the channel selection in MATCH is controlled by the parameter Cs. For example, when Cs is 16, then the zero-crossings produced by the w_{2D}=16 channel will be used for stereopsis.

This function is also called after the figural matching is carried out. Note that figural matching is carried out on only the zero crossing contours produced by the w_{2D} = 8 channel. Wherever, contour segments produced by that channel can be successfully matched by the function TRY_FIGURE_MATCH, we then invoke the finest channel of the MPG process by a call to the function DEFAULT_MATCH. This is taken care of by the second IF clause in the following algorithmic description of the function.

```
{
    if ( C(X, Y) = 4)
        begin
        call MATCH(X, Y, 0, 16, 16, Dn)
            if (Dn is empty)
                D ← propagate the disparity of the neighbors by
                    the interpolation technique
            else
```

```
                D ← Dn
            call MATCH(X, Y, 0, 8, 8, Dn)
            end
        if ( C(X, Y) = 3 or 4 )
            begin
                if (Dn is empty)
                    D ← propagate the disparity of the neighbors by
                    the interpolation technique
                else
                    D ← Dn
                    or  C(X, Y) = 4)
                call MATCH(X, Y, 0, W, 8, Dn)
                  (where the value W is given by
                        [ Rule Search-window-reduction-1 ])
            end
}
```

(7) CHOOSE_MOST_LIKELY_THETA (X, Y, Θ)

For the location (X, Y) in the control matrix, examine the 8-neighbors by examining the entries in the Θ matrix. (Note that there is one-to-one correspondence between the elements of the control matrix and the elements of the Θ matrix, the elements in the latter represent 4x4 overlapped groupings of the patches corresponding to the former.) From such a neighborhood, find the most likely orientation hypothesis by counting the number of the single entry elements in Θ.

(9) CHOOSE_MOST_LIKELY_Z(X, Y, Z)

From the list of disparities in the 8-neighbors of the (X, Y) location in the control matrix, the neighborhood being obtained from the Z matrix, retain for further consideration those that are unambiguous in the sense that only a single disparity entry exists in the Z matrix. Now take the average of these unambiguous entries and compare those possible at (X, Y) with this average. Retain one that is the closest.

(10) CREATE_INIT_C_MATRIX (C)

Using the results from dominant feature matching, initialize the control matrix. This process corresponds to the STEP-1 of the *planar strip generation algorithm* discussed in Section 4.2.2.

CHAPTER VI IMPLEMENTATION WITH AN EXPERT SYSTEM LANGUAGE

6.1. Introduction

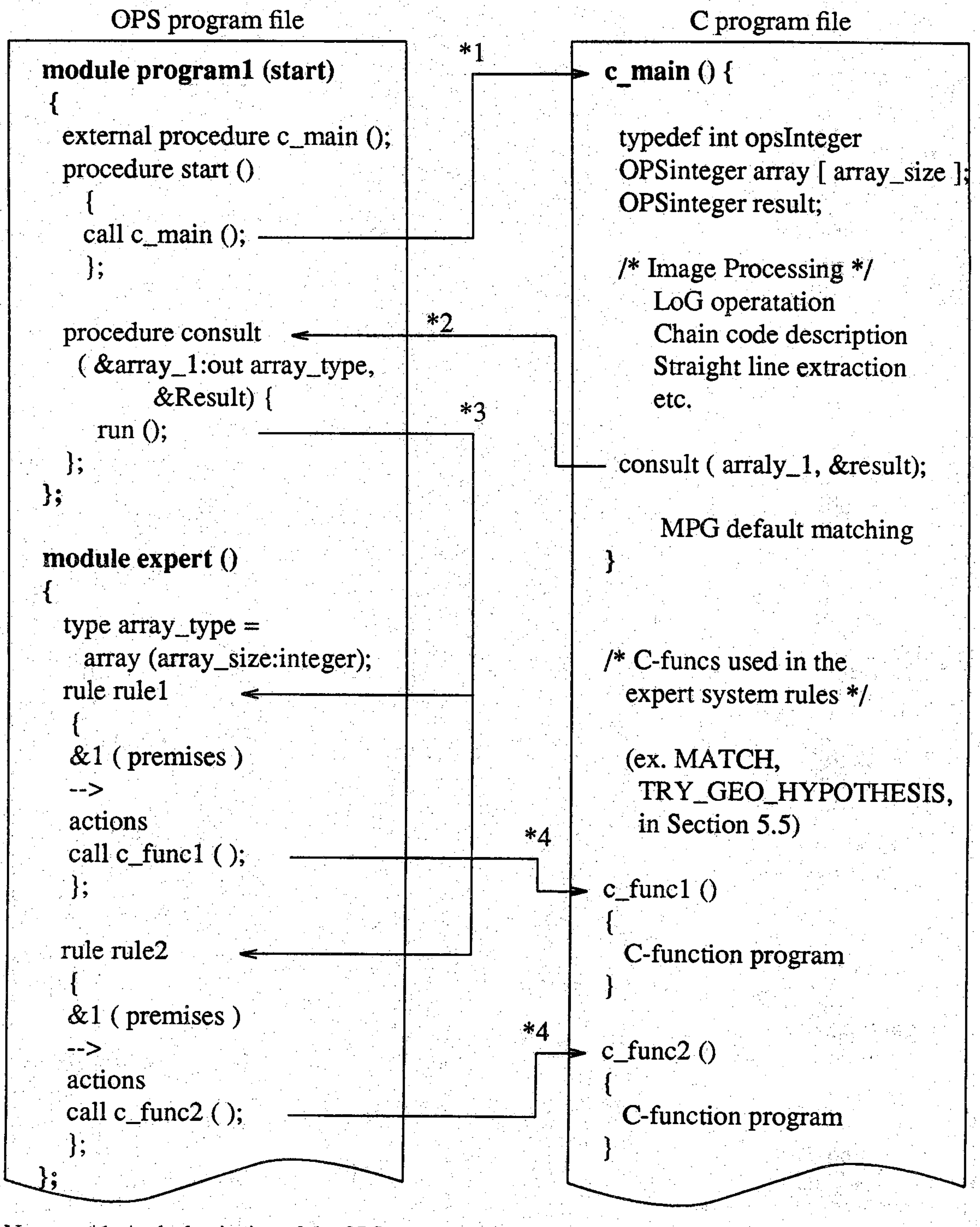
This chapter will briefly address some of the issues dealing with the interfacing of the rule-based part in OPS83 [33] and the rest of the software in the C language.

Our original desire was to set up the rule based program for strategy selection as a consultant to the main stereo processing software in C. In other words, we wanted the OPS83 part to serve as a subroutine to the rest of the system. However, OPS83 does not permit that, since in our lab it could only be initiated by the UNIX operating system. We, therefore, ended up setting a top level program in OPS83 which called the rest of the stereo matching software in C and also called the strategy selection parts written using the OPS83 rule writing facilities. The organization of the whole program is shown in Figure 52.

6.2. Calling the Rule-Based Part from the C Program

When rule-based decision making is necessary in the C program, OPS83 rules are invoked. The procedure for this part itself is very simple in principle. In the C program the name of the main procedure (or function) of the expert system is simply called just like a library function. The following presents the structure of the program.

```
/* main program written in C */ c_main () {
 typedef int opsInteger
 OPSinteger array [ array_size ];
 OPSinteger result;
 •
 •
 consult ( array_1 , &result ) }
```



```
module program1 (start)
{
  external procedure c_main ();
  procedure start ()
    {
    call c_main ();
    };

  procedure consult
   ( &array_1:out array_type,
          &Result) {
     run ();
  };
};

module expert ()
{
  type array_type =
    array (array_size:integer);
  rule rule1
   {
   &1 ( premises )
   -->
   actions
   call c_func1 ( );
   };

  rule rule2
   {
   &1 ( premises )
   -->
   actions
   call c_func2 ( );
   };
};
```

```
c_main () {

  typedef int opsInteger
  OPSinteger array [ array_size ];
  OPSinteger result;

  /* Image Processing */
     LoG operatation
     Chain code description
     Straight line extraction
     etc.

  consult ( arraly_1, &result);

      MPG default matching
}


/* C-funcs used in the
  expert system rules */

  (ex. MATCH,
   TRY_GEO_HYPOTHESIS,
   in Section 5.5)

c_func1 ()
{
  C-function program
}

c_func2 ()
{
  C-function program
}
```

Note : *1. At the beginning of the OPS program the control of program is transferred to C program.
*2. In the strategy selection part of the C program, the expert syetem "consult" is called.
*3. Rules are fired by the function "run".
*4. The subordinate C-functions can be called in any part of the OPS prgram.

Figure 52 Interactions between the OPS program and C programs.

```
module expert ()
 {
 type array_type = array ( array_size : integer );
 -- rules of the expert system
 rule rule1
  {
  &1
  -->
  -- actions
  }

 procedure consult ( &array_1 : out array_type, &Return_value : out integer )
  {
  run ();    -- procedure "run " exists in the shell
  };}
```

As this example indicates, when the consultation of a expert system is required, simply the name of the procedure written in OPS83, "consult" in the example above, is called in the C program. The type of parameters must be matched. (As the C compilation and OPS compilation are executed separatedly, the parameter disagreement is not examined by either of the compilers.) In the procedure consult, the "run" command is executed in order to fire rules.

6.3. Example of Rule Representation

In the previous chapter, we showed only the functional forms of the rules. In actuality, each rule must correspond to the format specified by OPS83. The following example illustrates in OPS83 format the rule for choosing the best geometric constraint.

```
rule choose-best-geo-con
{
 &1 (mode task = integrate_match; stage = 2 );
 &2 (control ctl [&X ][&Y] = 2 );
 &3 (hypothesis theta [&X][&Y][1] > 1 );
 -->
  remove &2;
  remove &3;
  call choose_most_likely_theta (&X, &Y,&Hypothesis,&Theta);
```

```
    make stage_2_rule_worked ( status = 'yes' );
}
```

Two lines just above the mark "-->" are the "IF" part of a rule. The lines below the mark "-->" are the "THEN" part of a rule. If scene classification allow the geometrically constrained matching, it is executed using a subprogram written in C. If it is not successful this region is recorded as smoothly continuous and probably the zero-crossing figure matching is invoked by the other program.

CHAPTER VII EXPERIMENTAL RESULTS

We will now show range maps computed using our rule based procedure and compare the results with those obtained with a straightforward application of the MPG algorithm. In addition to the range maps, we will also show the reduction in possible ambiguous matches with our scheme; a match is potentially ambiguous if for a zero-crossing in the left image there is more than one zero-crossing in the search window in the right image.

7.1. Configurations of the Experimental System

Our experimental setup is as shown in Figure 53. Random dot illumination is vertical and aimed straight down on objects placed on a table. The cameras are arranged in such a manner that their optic axes are nearly parallel, which eliminates the need for image rectification to satisfy the epipolar constraints. Any residual misalignments between the cameras are removed by hand adjusting their aiming angles until the images produced by them both are row-aligned. The random dot illumination is produced by a slide projector using a computer generated 1024×1024 pixels random-dot pattern. For regular illumination, the same slide projector with a blank slide is used. The cameras are placed about 99 inches from the center of the scene and the camera aiming angle is about 2.3 degrees from the perpendicular to the work table, making for an angle of 4.6 degrees between the optic axes. The images were digitized over 512×480 matrices with 8 bits for representing each gray level. After LoG filtering, 256×256 pixel pictures were supplied to the pattern matching algorithm.

7.2. Stereo Images and Depth Maps

The images observed by the left and the right cameras for a two box scene lit by regular light are shown in Figure 54 (a) (b). This pair of images is used for the dominant feature matching. The finest LoG operator is applied to these images. Then, as

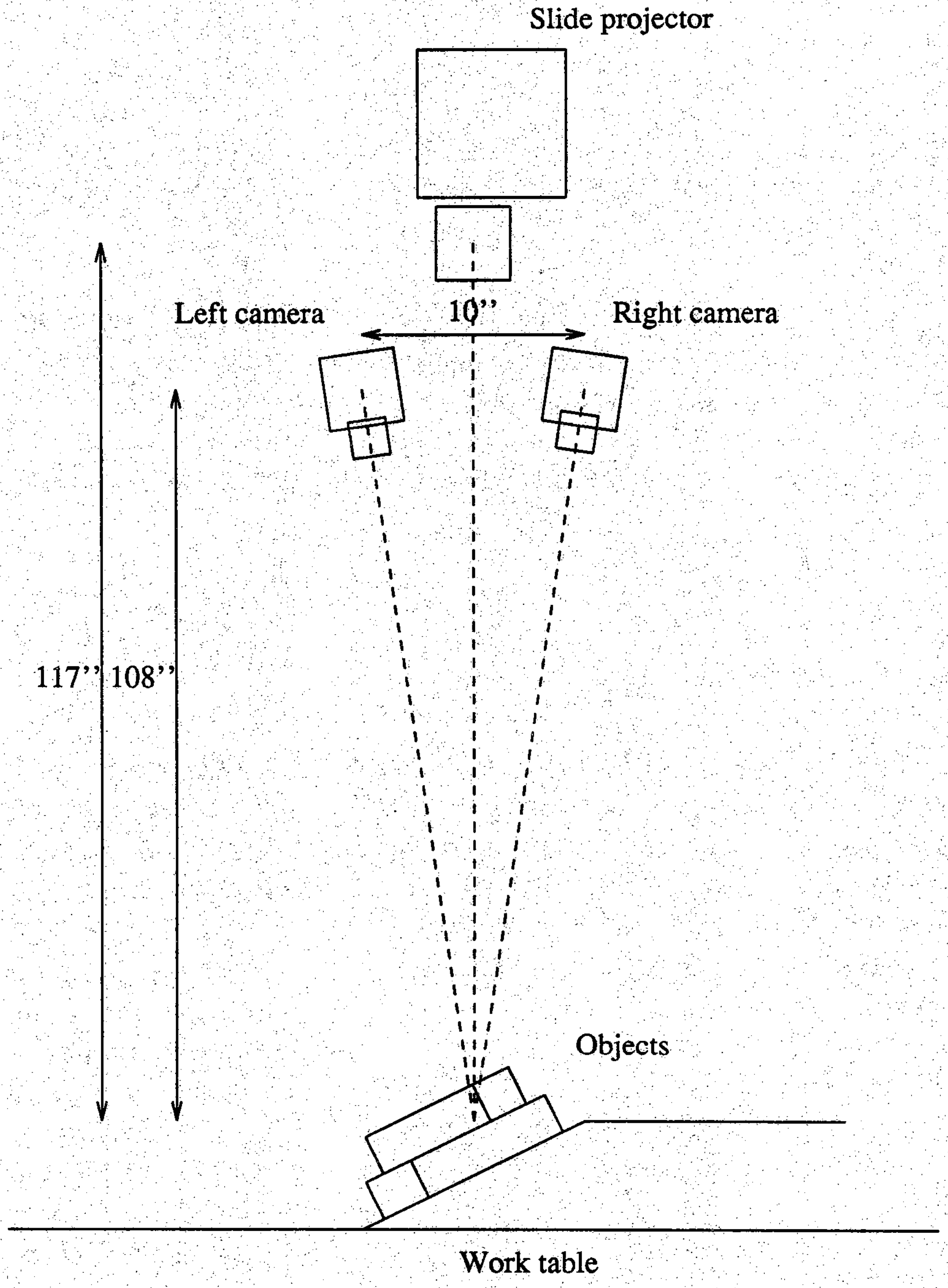
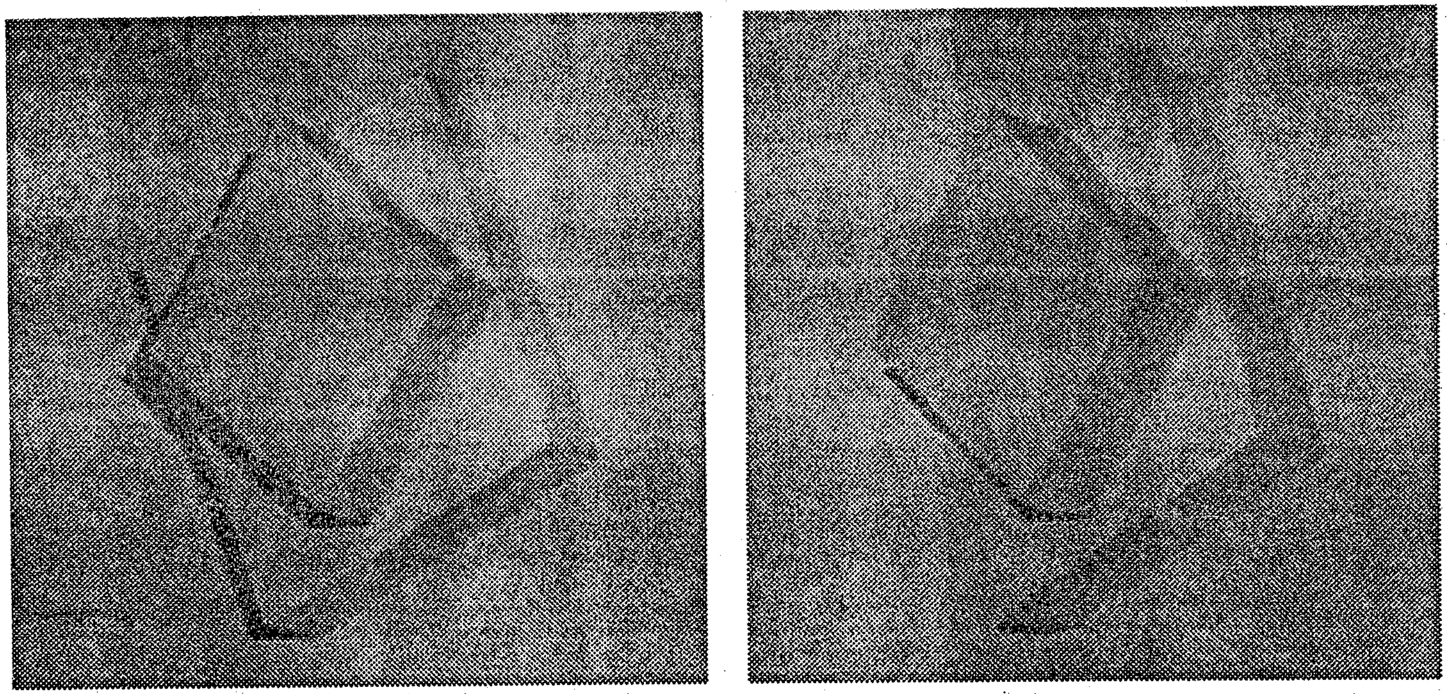


Figure 53 Geometry of experimental system.



(a) Left image.

(b) Right image.

Figure 54 A stereo scene (regular lighting).

the dominant feature, long straight lines are extracted. The zero-crossing contours which are extracted through the finest LoG operator ($w_{2D}=4$), are shown in Figure 55 (a) and (b). The result of the dominant feature extraction, or straight line extraction, is shown in Figure 56 (a) and (b).

For the geometrically constrained matching, zero-crossing figure matching and default matching the same scenes lit by the unstructured (random-dot) lighting is utilized. Three different types of feature extracting LoG-operators, $w_{2D} = 16$, $w_{2D} = 8$, and $w_{2D} = 4$, are employed. After applying these operators, the locally steepest changes of the gray level were extracted as zero-crossing contours. For the illustration a pair of zero-crossing contours with $w_{2D} = 8$ is shown in Figure 57.

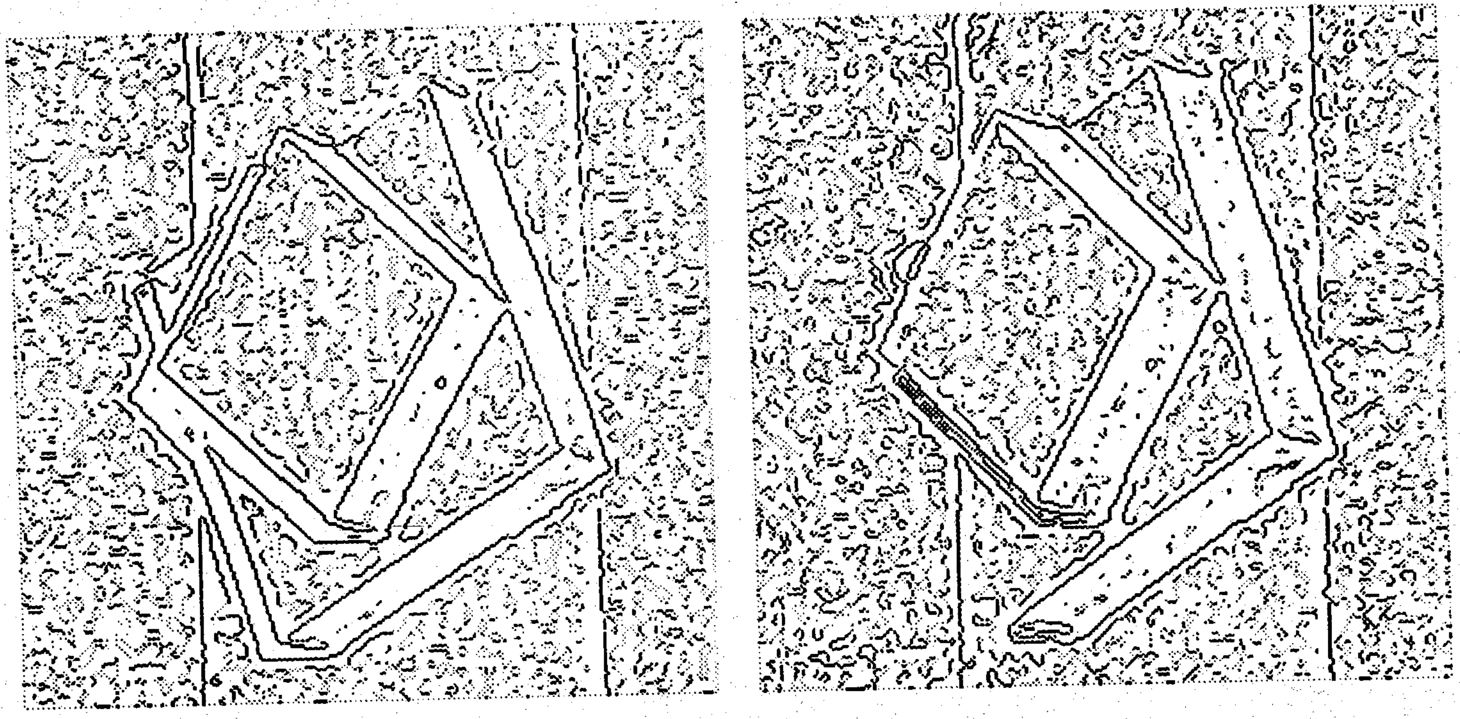
As described in the previous chapter, invocation of different matching procedures is controlled by the entries in the control matrix. For the scene under discussion, the control matrix as it exists just before the default matcher is invoked is shown in Figure 58. In the figure the control matrix entries are displayed by using different gray levels. The darkest grey level corresponds to the positions where the dominant feature matching is successful. The second darkest grey level represents the position where the geometrically constrained matching is successful. The third darkest gray levels correspond to the zero-crossing figure match. And, finally, the lightest grey level denotes the positions to which default matching is assigned. As was mentioned previously, the contents of the control matrix are not static and change during processing.

By invoking the default matching where dictated by the control matrix and computing the final disparities, the resulting depth map is shown in Figure 59. The depth is generated by using a zeroth order interpolation on the computed disparities.

7.3. Comparison with the MPG Algorithm

We will now illustrate a few comparative results obtained with our method and with the MPG algorithm. The depth map obtained with the MPG algorithm for the stereo pair of Figure 54 is shown in Figure 60. The same zeroth order interpolation is used for the MPG case also.

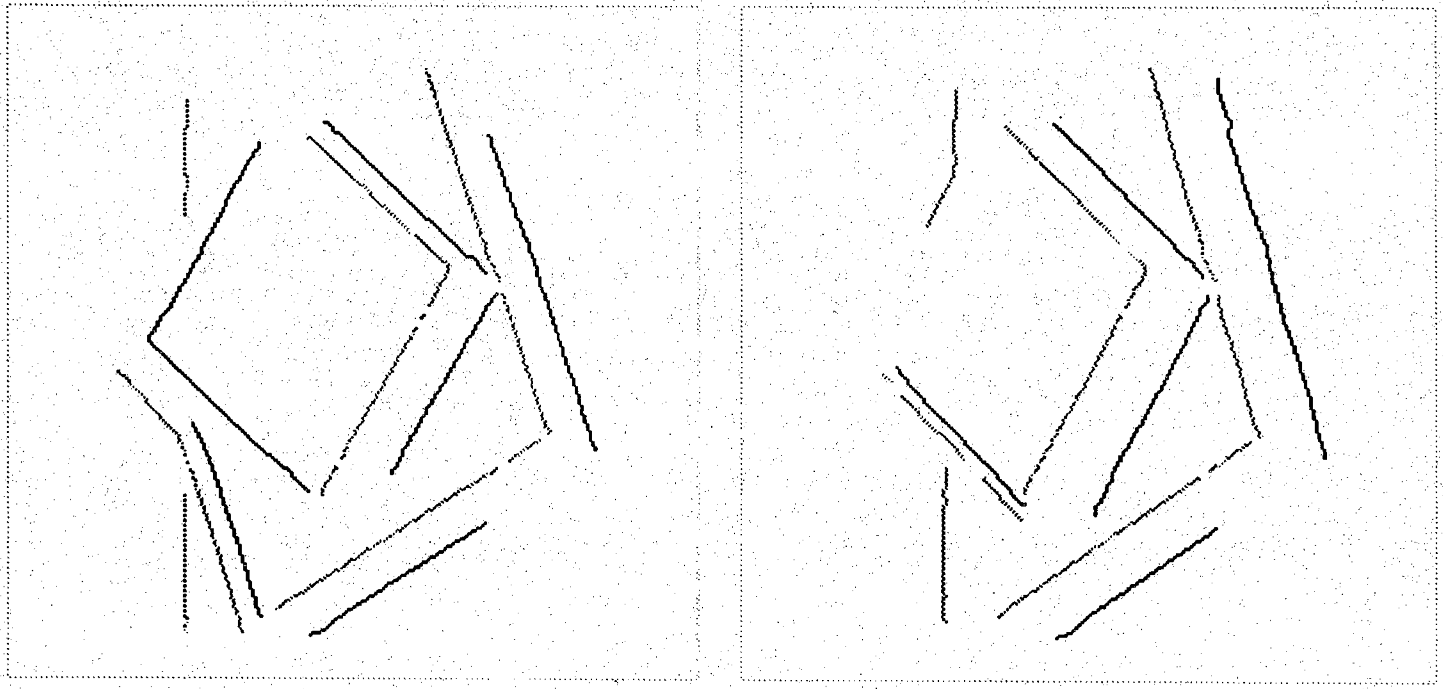
While the figures 59 and 60 provide a qualitative comparison of the results obtained with our and the MPG methods, it is also possible to perform a quantitative comparison by keeping track of the number of left-image zero-crossings that have more than one possible zero-crossing in the right image for a potential match and for which one of these zero-crossings cannot be selected by the disambiguation process. If this number of zero-crossings is divided by the total number of zero-crossings available, we get a measure of robustness of the matching process.



(a) Left image.

(b) Right image.

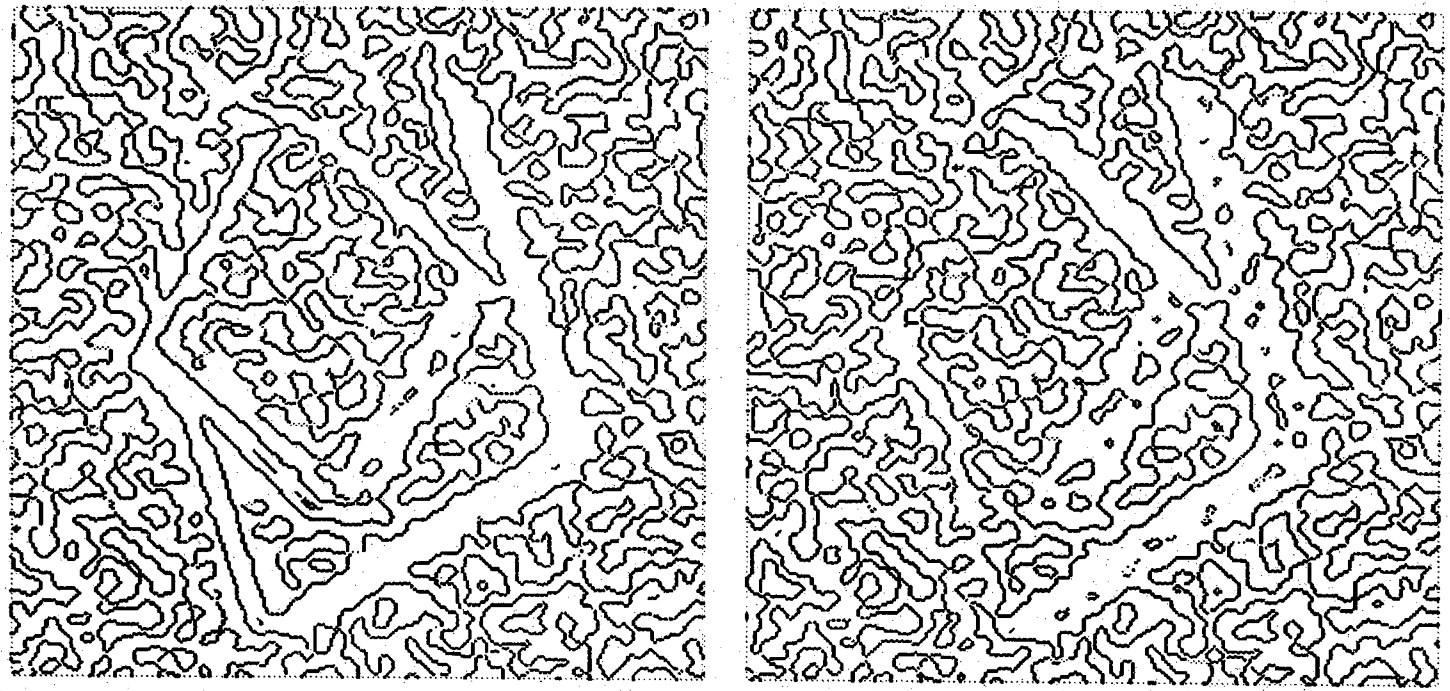
Figure 55. Zero-crossing contours ($w_{2D} = 4$, regular lighting).



(a) Left Image

(b) Right Image

Figure 56 Extracted dominant features.



(a) Left image.

(b) Right image.

Figure 57 Zero-crossing contours ($w_{2D} = 8$, unstructured lighting).

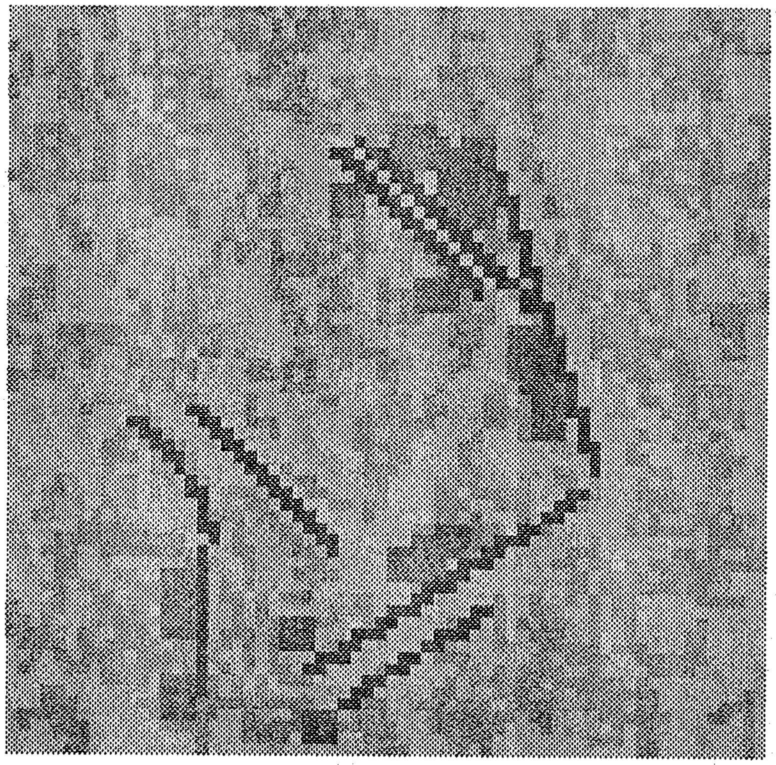


Figure 58 Control matrix.

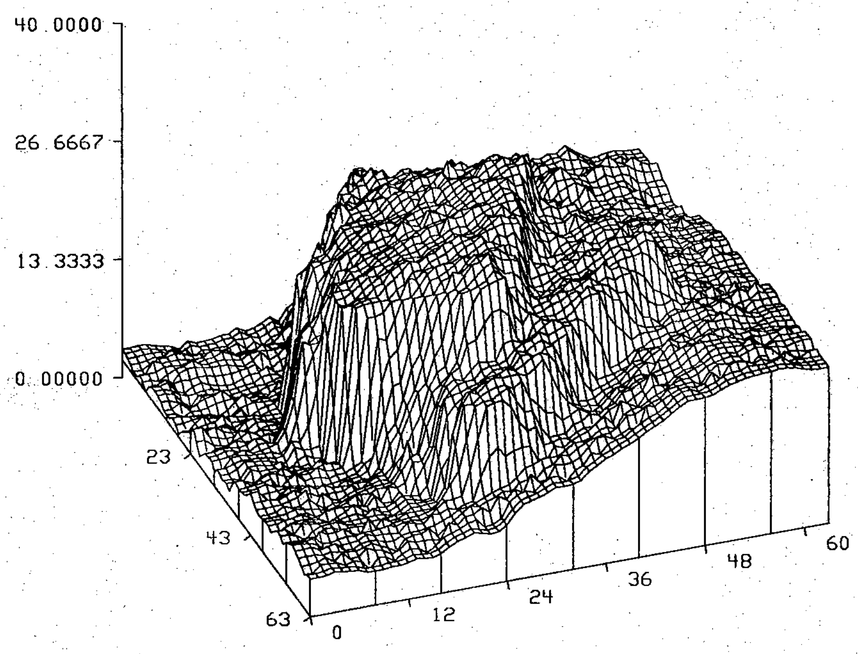


Figure 59 Range map of a two box scene (by our algorithm).

As presented in the previous chapter, our method uses a narrow width disambiguation window. Therefore, the disparity value on some occasion, may nullify the values given by the matching. This nullification occurs in two cases : if the range provided by the coarser channel is entirely wrong, or if the value given by the pulling effect is inconsistent with the currently measured disparity value. The latter case is likely to occur in both methods, the usual MPG and rule-based method, with almost the same rate. The difference of the rate of the occurrence of the nullification mostly reflects the amount of the misleading information by the coarser channel.

Table 7 shows the counts of several aspects in the disambiguation process regarding four scenes shown in Figure 55, 61, (a), (b), and (c). The computation was executed in both the usual MPG method and the integrated rule-based approach. For each method, three numbers are counted: total number of image points which have any zero-crossing match (A), the number of image points where the matching was entirely nullified by the disambiguation (B), and the number of image points where a better range value was chosen among the candidates by the disambiguation (C). The upper number of each box of the table is for the usual MPG approach. The lower number in each box of the table is for our rule-based matching model. As the total number of matches for both methods is different, the ratios B/C, and C/A are also computed and shown in terms of percent in Table 7. In the case of the usual model, all matching operations are taken into account. In the case of the rule-based model, the number represents the quantity of matchings in the default matching. The algorithms used in both methods are exactly the same, for matching in all aspects such as the matching criteria, disambiguation processing, search space size etc.

The numbers for both methods appearing in the column C/A are approximately the same. This is reasonable because the number of the feature points in a certain search space is completely determined by the relation between the LoG operator size and the search space size as shown by the Marr-Poggio theory. Looking closely at the table, one may find a weak tendency that the the lower number is larger than upper number. One possible explanation for this is that the ambiguous occasion will occur more likely when the zero-crossing figure match is not applicable.

We assumed that the matching nullification occurs more frequently if the range measurement in the coarser channel was less accurate. If our claim is true, the difference of the number in the column B/A (expressed as percent) in the box corresponds to W_{2D}=8. This indicates the difference of the frequency of the misleading information caused by the coarser channel, W_{2D}=16. This value in the case of the usual method is 24.5 percent on the average of four images. The same value in the case of our model is 16.8 percent. Similarly, for the boxes corresponding to W_{2D}=4 the values are 13.3 and 8.5 respectively. This sort of difference is not observed in the coarsest channel fusion. (The values are 15.0 and 15.0 respectively.) This is because there exists no coarser channel and the same fixation points (zero-disparity) are employed.

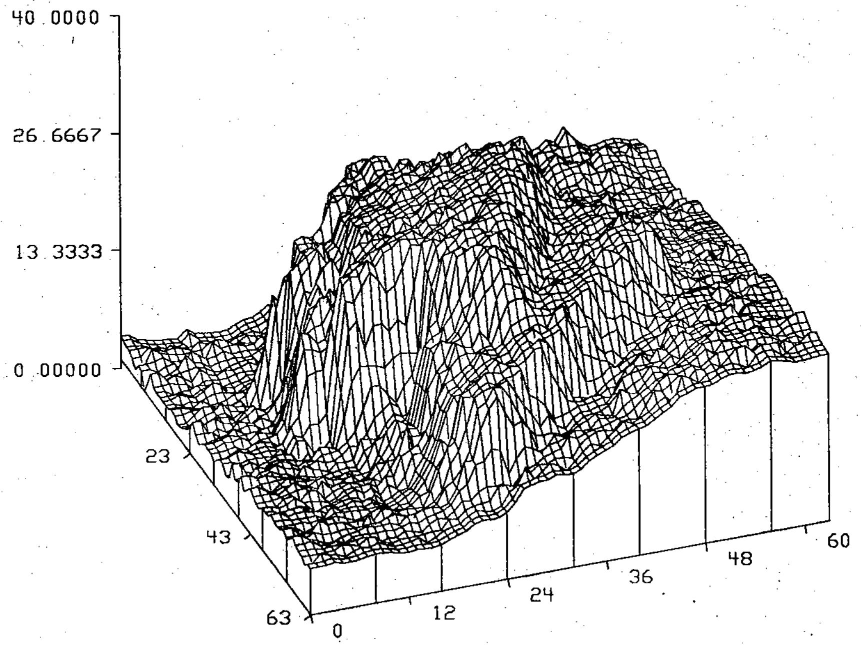
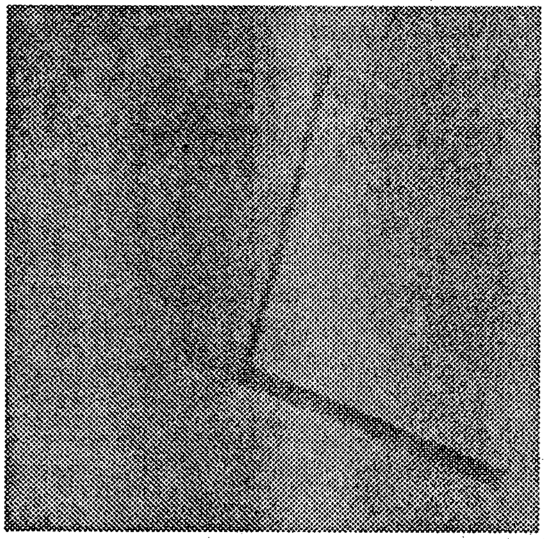
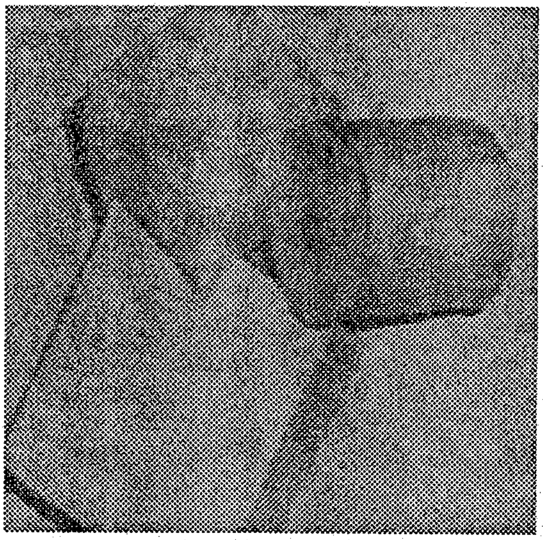


Figure 60 Range map of a two box scene (by MPG algorithm).


(a) One Box (Image 2).


(b) Three objects (Image 3).

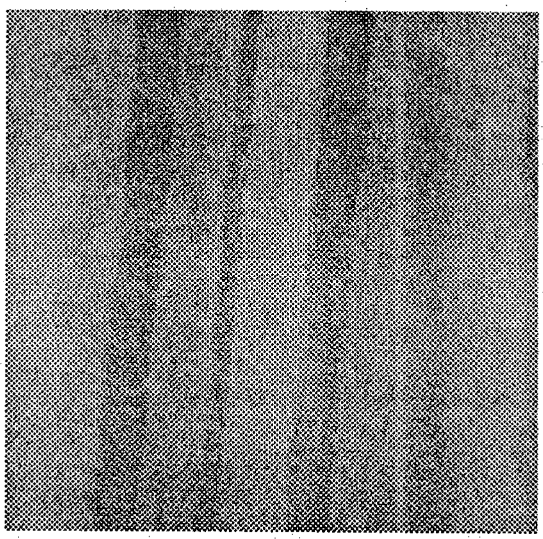

(c) Steps (Image 4).

Figure 61 Experiment scenes (regular light).

Table 7 Comparison between the MPG and our method.

Image No.	W_{2D}	Total No. Matching A	No. Nullified by Disamb. B	(Ratio) $\frac{B}{A}\times 100$	No. Disamb. Affected C	(Ratio) $\frac{C}{A}\times 100$
1	16	3010 1562	366 206	12 13	865 457	28 29
	8	5638 2823	1408 484	24 17	2134 927	37 32
	4	7184 6419	946 555	13 8	1538 1168	21 18
2	16	2554 1482	367 209	14 14	834 463	32 31
	8	5789 3492	1574 648	27 18	2557 1243	44 35
	4	6424 5470	988 510	15 9	1885 1191	29 21
3	16	3044 1378	511 244	16 17	928 466	30 33
	8	5787 2765	1391 476	24 17	2270 1060	39 38
	4	7356 6326	1004 590	13 9	1711 1223	23 19
4	16	3577 920	655 143	18 15	926 258	25 28
	8	6163 1722	1444 281	23 16	1769 496	28 28
	4	7207 5754	898 490	12 8	1229 805	17 13

Upper numbers are for the usual MPG model
Lower numbers are for the integrated rule-based model

It can be claimed that the misleading information by the coarser channel occurs more frequently in the usual method than our integrated rule-based method. This is because, for the fixation point determination prior to the medium channel ranging, the ranges determined by the geometrically constrained matching and zero-crossing figure match are taken into account in the rule-based model.

Another point to be discussed is the total number of the matches. As some part of the matching process in the MPG method is replaced by other methods, the number of point-wise matches normally appearing in the MPG method is significantly reduced. As this method uses the local orientation, and the sign of the zero-crossings as the attributes, the possibility of the erroneous correspondence in the usual MPG is occasionally higher than it is in other higher level matching. The reduction of the number of point-wise matches may lead to the reduction of the number of erroneous matching. The column of A in Table 7 indicates the total number of the point-wise matches. In the case where W_{2D} is 16 or 8, the reduction is about 56 percent. In the case where W_{2D} is 4, the reduction is about 14 percent.

CHAPTER VIII CONCLUSIONS

In this report we have shown that the MPG procedure, although a landmark development in the evolving science of stereo vision, suffers from major deficiencies on account of its being a purely bottom-up approach. We pointed out that this procedure will generate disparity values at all those points which exhibit photometric differences. Since such points in a scene are a result of a complex interaction between illumination and the surfaces in the scene, their locations cannot be predicted in advance. Therefore, in most cases, especially if the scene surfaces are smooth, the pixels at which the disparities get calculated can be more or less randomly located, making difficult the process of interpolation for the purpose of computing continues looking depth maps.

In our work, our position has been that the problem with the MPG algorithm can be alleviated by injecting high level object knowledge into the matching process. In this report, we took a first step in that direction by assuming that all the object surfaces were planar and then this knowledge was directly used in the matching process.

Although, our use of object surface constraints by itself cannot be considered to be new, what is new in our work is the integration of such constraints with other approaches to the matching process -- it seems plausible that such integration is also carried out by the human visual system. Since it is entirely likely that the human visual system latches on to any striking geometrical detail and since such details probably are fused at the outset -- if only to facilitate the fusion of less striking scene details -- in our combination method, we first extract dominant features, these being the straight lines constituted by joining of the planar faces of the scene objects.

In our rule based approach, we showed how the straight line feature matching was followed by the figural continuity matcher. And, the figural continuity matcher was followed by the default matcher, which is full blown implementation of the MPG algorithm. By embedding all these matching algorithms in a rule-based setting, the system became capable of invoking each of the methods on an opportunistic basis, meaning if, say, the figural continuity matcher did not produce acceptable results in a region of the image, the system could then automatically go ahead and invoke the default matcher. The bookkeeping for which matcher to apply where was done with the help of a 64×64 control matrix for 256×256 images.

Clearly, this report is only a first step in our attempt to inject high level knowledge into the process of binocular fusion. We are sure that future attempts would try to deal with higher order surfaces in scenes of greater complexity that what we have shown.

BIBLIOGRAPHY

[1] Julesz, B., "Binocular Depth Perception of Computer-Generated Patterns," *Bell Syst. Tech. J.*, vol. 39, pp. 1125 - 1162, 1960.

[2] Marr, D. and T. Poggio, "A computational Theory of Human Stereo Vision," *Proc. of Royal Society of London, Section B*, vol. 204, pp. 301 - 328, 1979.

[3] Grimson, W. E. L. , "A Computer Implementation of a Theory of Human Stereo Vision," *Philosophical Trans. of the Royal Society of London, Section B*, vol. 292, pp. 217 - 253, May, 1981.

[4] Ayache, N., O. D. Faugeras, B. Faverjon and G. Toscani, "Matching Depth Maps Obtained by Passive Stereo," *Proc. of 3rd Workshop on Computer Vision*, pp. 197 - 204, 1985.

[5] Medioni, G. and R. Nevatia, "Segment-Based Stereo Matching," *Computer Vision, Graphics, and Image Processing*, vol. 31, pp. 2 - 18, 1985.

[6] Eastman, R. D. and A. M. Waxman, "Using Disparity Function for Stereo Correspondence ," *Computer Vision, Graphics, and Image Processing*, vol. 39, pp. 73 - 101, 1987.

[7] Hoff W. and N. Ahuja, "Extracting Surfaces from Stereo Images : An Integrated Approach," *U. of Illinois Technical Report*, vol. UILU-ENG-87-2204, 1987.

[8] Mayhew, J. E. W. and J. P. Frisby, "Psychophysical and Computational Studies towards a Theory of Human Stereopsis," *Artificial Intelligence*, vol. 17, pp. 349 - 385, 1981.

[9] Grimson, W. E. L. , "Computational Experiments with a Feature Based Stereo Algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, No. 1, pp. 17 - 34, January, 1985.

[10] Nishihara, H. K. , "PRISM: A Practical Real-Time Imaging Stereo Matcher," *Optical Eng.*, vol. 23, pp. 536-545, 1984.

[11] Ikeuchi, K. , H. K. Nishihara, B. K. P. Horn, P. Sobalvarro and S.Nagata "Determining Grasp Configurations Using Photometric Stereo and the PRISM Binocular Stereo System," *The International Journal of Robotics Research*, vol. 5 No. 1, pp. 46-65, 1986.

[12] Grimson, W. E. L. , *From Images to Surfaces,* The M. I. T. Press, Cambridge, Massachusetts, 1981.

[13] Kak, A. C. , "Depth Perception for Robots," *Handbook of Industrial Robotics,* pp. 272-319, John Wiley, New York, 1985.

[14] Chen, J. S., A. Huertas and G. Medioni, "Very Fast Convolution with Laplacian-of-Gaussian Mask," *Proc. Computer Vision and Pattern Recognition* , pp. 293 - 298, 1986.

[15] Berzins, V. , "Accuracy of Laplacian Edge Detectors," *Computer Vision, Graphics, and Image Processing,* vol. 27, pp. 195 - 210, 1984.

[16] Witkin, "Scale-Space Filtering," *Proc. 8th Int. Joint Conf. Artif. Intell.,* Karlsruhe, West Germany, pp. 1019 - 1022, 1983.

[17] Horn, B. K. P., "Robot Vision," *MIT Press,* Cambridge,MA, 1986.

[18] Baker H. H. and T. O. Binford, "Depth From Edge and Intensity Based Stereo," *Proc. Seventh Int. Joint Conf. Artif. Intell.,* pp. 631 - 636, 1981.

[19] Ohta Y. and T. Kanade, "Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI-7, No. 2, pp. 139 - 154, March, 1985.

[20] Lloyd S. A. , E. R. Haddow and J. F. Boyce, "A Parallel Binocular Stereo Algorithm Utilizing Dynamic Programming and Relaxation Labelling," *Computer Vision, Graphics, and Image Processing,* vol. 39, pp. 202 - 225, 1987.

[21] Herman M. and T. Kanade, "Incremental Reconstruction of 3D Scenes from Multiple, Complex Images," *Artificial Intelligence,* vol. 30, pp. 289 - 341, 1986.

[22] Nevatia, R., "Depth Measurement by Motion Stereo," *Computer Graphics and Image Processing,* vol. 5, pp. 203-214, 1976.

[23] Moravec, H. P. , "The Stanford Cart and the CMU Rover," *Proceedings of the IEEE,* vol. 71 No. 7, pp. 872-884, 1983

[24] Tsai, R. , "Multiframe Image Point Matching and 3-D Surface Reconstruction," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI-5, No. 2, pp. 159 - 173, March, 1983,

[25] Nevatia, R. and K. R. Babu, "Linear Feature Extraction and Description," *Computer Graphics and Image Processing,* vol. 13, pp. 257 - 269, 1980.

[26] Freeman, H. "Computer Processing of Line-Drawing Images," *Computing Surveys,* vol. 6, pp. 57 - 97, March, 1974.

[27] Rosenfeld, A., "Digital Straight Line Segments," *IEEE Trans. on Computers,* vol. c-23 No. 12, pp. 1264 - 1269, December, 1974.

[28] Suk, M. and H. Kang, "New Measures of Similarity Between Two Contours Based on Optimal Bivariate Transforms," *Computer Vision, Graphics, and Image Processing,* vol. 26, pp. 168 - 182, 1984.

[29] Burr, D. J. , "Elastic Matching of Line Drawings," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI - 3,, pp. 708 - 713, 1981.

[30] Pavlidis, T. and F. Ali, "A Hierarchical Syntactic Shape Analyzer," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI-1, No. 1, pp. 2 - 9, January, 1979.

[31] Pavlidis, T. , "The Use of a Syntactic Shape Analyzer for Contour Matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI-1, No. 3, pp. 307 - 310, July, 1979.

[32] Sze, T. W. and Y. H. Yang, "A Simple Contour Matching Algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI - 3, pp. 676 - 678, 1981.

[33] Forgy C. L. , *The OPS83 User's Manual System Version 2.2,* Production Systems Technologies, Inc., 1986.

[34] K. L. Boyer and A. C. Kak, "Structural Sterepsis in 3-D Vision," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI-10, No. 2, pp. 144 - 166, March, 1988.