**Carnegie Mellon University**
**Research Showcase**

Computer Science Department | School of Computer Science

1-1-1983

# Stereo by intra- and inter-scanline search using dynamic programming

Ohta
*Carnegie Mellon University*

Takeo Kanade

Follow this and additional works at: http://repository.cmu.edu/compsci

Recommended Citation
Ohta and Kanade, Takeo, "Stereo by intra- and inter-scanline search using dynamic programming" (1983). *Computer Science Department.* Paper 1473.
http://repository.cmu.edu/compsci/1473

This Technical Report is brought to you for free and open access by the School of Computer Science at Research Showcase. It has been accepted for inclusion in Computer Science Department by an authorized administrator of Research Showcase. For more information, please contact research-showcase@andrew.cmu.edu.

**NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:**
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

CMU-CS-83-162

# Stereo by Intra- and Inter-scanline Search Using Dynamic Programming

Yuichi Ohta
Takeo Kanade

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

October 1983

## Abstract

This paper presents a stereo algorithm for obtaining an optimal matching surface in a three dimensional search space. Our approach is purely computational. When a pair of stereo images is rectified so that the epipolar lines are horizontal scan lines, we can search for a pair of corresponding points in right and left images within the same scan lines. We call this search *intra-scanline* search. This intra-scanline search can be treated as the problem of finding a matching path on a two dimensional (2D) search plane whose axes are the right and left scanlines. The intra-scanline search alone, however, does not take into account mutual dependency between scanlines in an image. *Inter-scanline* search is necessary to find the consistency among scanlines.

Therefore, the problem of finding a correspondence between pair of stereo images can be cast as that of finding a matching surface (i.e., a set of matching paths) in a three dimensional search space, which is a stack of the 2D search planes. Vertically connected edges provide the consistency constraint across the 2D search plane. Thus, stereo involves two searches: one is inter-scanline search for possible correspondence of connected edges and the other is intra-scanline search for correspondence of every edge under the constraint given by connected edges. We utilize dynamic programming for both searches and they proceed simultaneously.

The algorithm has been tested with images including urban aerial images, synthesized images, and block scenes.

This research was sponsored by the Defense Advanced R

# 1. Introduction

Stereo is a useful method of obtaining depth information without using active sensors. Many researchers are currently studying edge-based stereo techniques [Grimson and Marr 79] [Baker and Binford 81]. Edge-based stereo is usually classified as a feature-based technique, but it can produce a relatively dense depth map as compared to typical feature-based techniques [Moravec 79] [Herman, Kanade, and Kuroe 83]. Edges usually have a higher density than the feature points by the Moravec's interest operator or the junctions at the corner of man-made objects. Feature points with high density produces a dense depth map, but on the other hand, they cause ambiguities in finding the correspondences in the right and the left images.

Therefore, one of the key problems in edge-based stereo is to find the correspondence of edges. Basically, the problem of finding the correspondence of edges in the right and the left images involves the search in the two dimensional image. We refer to this as the image-to-image correspondence problem. When we know the camera model, we can simplify the image-to-image correspondence problem into a set of scanline-to-scanline problems. That is, when a pair of stereo images is rectified so that the epipolar lines are horizontal scanlines, we can search for a pair of corresponding edges in right and left images within a pair of scanlines [Barnard and Fischler 82]. We call this search *intra-scanline* search. This intra-scanline search can be treated as the problem of finding a matching path on a two-dimensional search plane whose vertical and horizontal axes are the right and left scanlines. A dynamic programming technique can handle this search efficiently [Baker and Binford 81]. The intra-scanline search alone, however, does not take into account mutual dependency between scanlines in an image. *Inter-scanline* search is necessary to find the consistency among scanlines.

By considering both intra- and inter-scanline searches, the problem of finding a correspondence between pair of stereo images can be cast as that of finding a matching surface (*i.e.*, a set of matching paths) in a three dimensional search space, which is a stack of the 2D search planes and whose axes are left-image column position, right-image column position, and the row position of image. The cost of the matching surface is defined as the sum of the costs of the intra-scanline matches on the 2D search planes. Vertically connected edges provide the consistency constraint across the 2D search plane (inter-scanline constraint). Our stereo algorithm tries to find an optimal set of paths which minimizes the matching cost under the inter-scanline constraint. It involves two searches: one is inter-scanline search for possible correspondence of connected edges and the other is intra-scanline search for correspondence of edges under the constraint given by connected edges. Dynamic programming is used for both searches and they proceed simultaneously. It reduces the computation to a feasible order.

Our main task domain is urban aerial photographs which contain tall buildings, roads, and trees. Images in

other domains are also used to show the performance of our stereo technique.

## 2. The Problem

In an edge-based stereo system, vertically connected edges across the scanline offer the consistency constraint among scanlines. Suppose a connected edge $u$ in the right image matches a connected edge $v$ in the left image on scanline $t$ as shown in figure 2-1. Then they should also match on other scanlines. If they do not match on scanline $t$, they should not match on others, either. This constraint helps to get correct matches near the corners where many edges come together as on the scanline $t+1$ in the figure. It is also effective in distinguishing noisy edges from real ones as illustrated on scanline $t-1$.

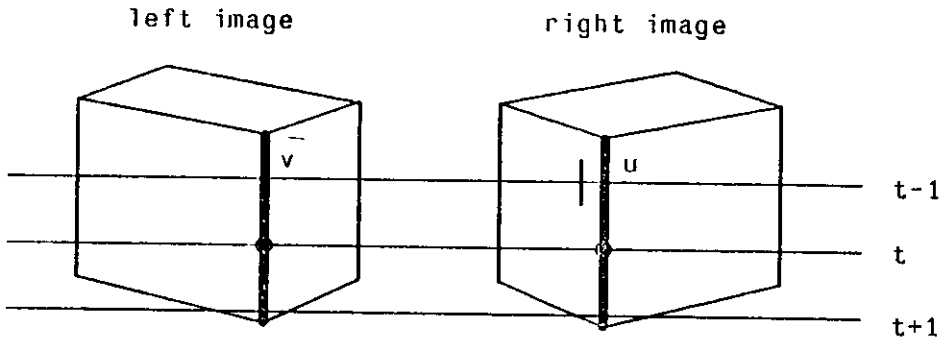



**Figure 2-1:** Constraint provided by vertically connected edges

The problem is, then, how to combine the inter-scanline consistency constraint with the intra-scanline search scheme. Henderson [Henderson, et al. 79] sequentially processed each pair of scanlines and used the results obtained to guide searches in succeeding scanlines. However, in this scheme the matching process at any given scanline is significantly affected by the errors in the preceding scanlines. This is more serious when the scanlines are processed in top-to-bottom or in bottom-to-top order, because in many cases we should decide the correspondence of connected edges near the ends where other connected edges may appear closely, as in figure 2-1, and where the ambiguity is larger than in other areas.

Baker [Baker 82] first processed each pair of scanlines independently. After all the intra-scanline matching was done, he used a cooperative process to detect and correct the matching results which violate the consistency constraint. The problem of this method is that the inter-scanline constraint is not used in the search. Thus, the result from the cooperative process is not guaranteed to be optimal. Baker suggested the necessity of a search which finds an optimal result satisfying the consistency constraint in a 3D search space. But a feasible method has been an open problem.

A straightforward way to achieve a matching which satisfies the edge continuity constraint is to consider all

matching between connected edges in the right and left images. However, since the typical number of connected edges is several hundred in each image, a brute force method is infeasible.

Dynamic programming is one method for solving this problem. It solves an $N$ stage decision process as $N$ single stage ones. This reduces the computational complexity to the logarithm of the original one. In order to apply dynamic programming to a decision process, the process should satisfy the following two requirements. First, the decision stages must be ordered so that every stage whose results are needed at a given stage is processed before the given one. Second, the decision process should be *Markovian*: at any stage the behavior of the process depends solely on the current state and does not depend on the previous history. These are not at all obvious in the correspondence problem of connected edges, but we clarify them in the following sections.

Our search space is a 3D space which is a stack of 2D search planes for intra-scanline matching. In the 3D search space we obtain a set of matching paths which gives the optimal correspondence of edges under the inter-scanline consistency constraint. Actually the scheme involves two searches as shown in figure 2-2. One is for the correspondence of all connected edges in the right and left images, and the other is for the correspondence of edges on the right and left scanlines under the constraint given by the former. The score (cost) of the connected edge correspondence is supplied by the latter one. We employed dynamic programming for both searches and they proceed simultaneously.

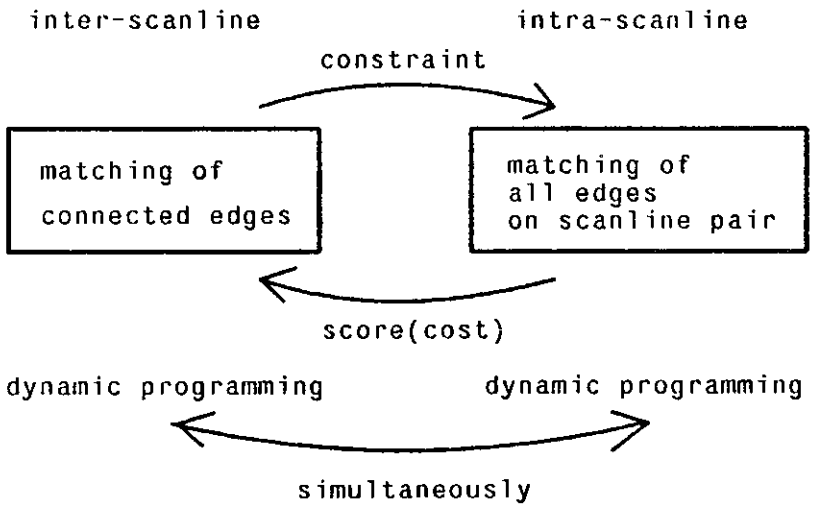



**Figure 2-2:** Two searches involved in stereo matching

# 3. Correspondence Search Using Dynamic Programming

## 3.1. Intra-scanline search on 2D plane

The problem of obtaining a correspondence between edges on the right and left scanlines can be solved as a path finding problem on a 2D plane. Figure 3-1 illustrates this 2D search plane. The vertical lines show the positions of edges on the left scanline and the horizontal ones show those on the right scanline. We refer to the intersections of those lines as nodes. Nodes in this plane correspond to the stages in dynamic programming where a decision should be made to select an optimal path to that node. In the intra-scanline search, the stages must be ordered as follows: *When we examine the correspondence of two edges, one on the right and one on the left scanline, the edges which are on the left of these edges on each scanline must already be processed.* For this purpose, we give indices for edges in left-to-right order on each scanline: $[0{:}M]$ on the right and $[0{:}N]$ on the left. Both ends of a scanline are also treated as edges for convenience. It is obvious that the condition above is satisfied if we process the nodes with smaller indices first. The path is a consecutive set of straight line segments from node (0,0) to node $(M,N)$ on a 2D array $[0{:}M,0{:}N]$. It goes from the upper left to the lower right corners monotonically, if we assume the no-reversal constraint in edge correspondence: that is, the order of matched edges has to be preserved in the right and left scanlines. A path has a vertex at node $\boldsymbol{m}=(m,n)$ when right edge $m$ and left edge $n$ are matched.

The cost of the path is defined by equation (3-1). Let $D(\boldsymbol{m},\boldsymbol{k})$ be the minimal cost of the partial path from node $\boldsymbol{k}$ to node $\boldsymbol{m}$. We denote $D(\boldsymbol{m},\boldsymbol{k})$ as $D(\boldsymbol{m})$ when $\boldsymbol{k}$ is (0,0). $D(\boldsymbol{m})$ is the cost of the optimal path to node $\boldsymbol{m}$ from the origin (0,0). A primitive path is a partial path which contains no vertices and it is represented by a straight line segment as shown on figure 3-1. The cost of a path is the sum of those of its primitive paths. Let $d(\boldsymbol{m},\boldsymbol{k})$ be the cost of the primitive path from node $\boldsymbol{k}$ to node $\boldsymbol{m}$. Obviously, $d(\boldsymbol{m},\boldsymbol{k}) \geq D(\boldsymbol{m},\boldsymbol{k})$ and on an optimal path $d(\boldsymbol{m},\boldsymbol{k}) \equiv D(\boldsymbol{m},\boldsymbol{k})$. Our definition of $d(\boldsymbol{m},\boldsymbol{k})$ will be given in section 4.2.

$D(\boldsymbol{m})$ can be defined recursively as:

$$D(\boldsymbol{m}) = \min_{\{\boldsymbol{i}\}} \{d(\boldsymbol{m},\boldsymbol{m}-\boldsymbol{i}) + D(\boldsymbol{m}-\boldsymbol{i})\}$$
$$D(\boldsymbol{0}) = 0 \qquad (3-1)$$

here $\boldsymbol{m}=(m,n)$, $\boldsymbol{i}=(i,j)$, $0 \leq i \leq m$, $0 \leq j \leq n$, $i+j \neq 0$.

Vector $\boldsymbol{i}$ represents a primitive path coming to node $\boldsymbol{m}$. When $i=0$, the primitive path is horizontal, such as

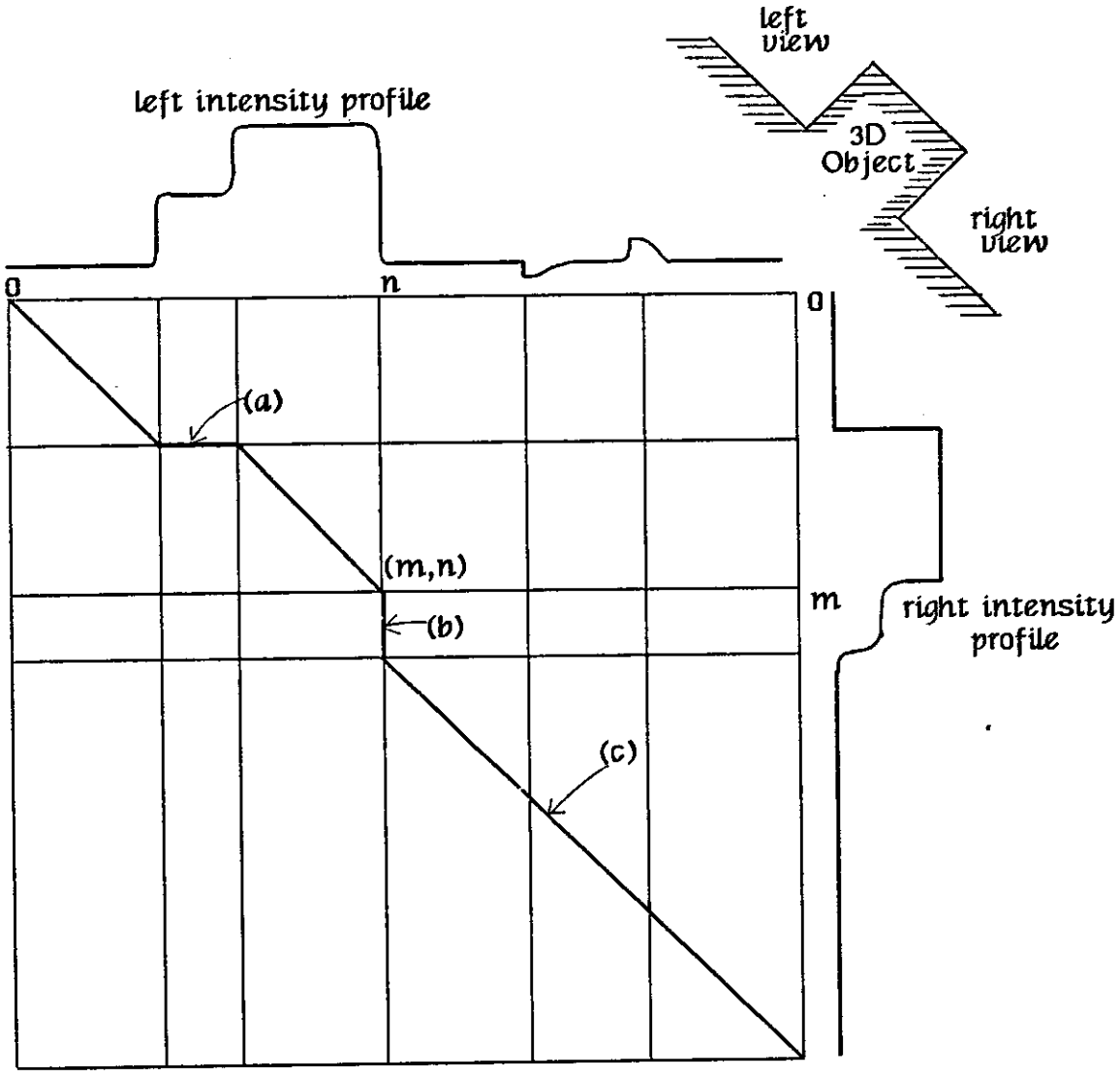



**Figure 3-1:** 2D search plane for intra-scanline search.
*Intensity profiles are shown along each axis.*
*The horizontal axis corresponds to the left scanline*
*and the vertical one corresponds to the right scanline.*
*Vertical and horizontal lines are the edge positions*
*and path selection is done at their intersections.*

is shown at (a) in figure 3-1. It corresponds to the case in which a visible part in the left image is occluded in the right image. When $j=0$, the primitive path is vertical, such as is shown at (b). When $i>1$ and/or $j>1$, the primitive path goes beyond $i-1$ and/or $j-1$ edges on the right and/or left scanlines. This means those $i-1$ and/or $j-1$ edges are to be ignored in the matching, as is shown at (c) in the figure. Such a path corresponds to the case where some edges have no corresponding ones on the other scanline because of noise.

The iteration starts at $\boldsymbol{m}=(0,0)$ and computes $D(\boldsymbol{m})$ for each node $\boldsymbol{m}$ in ascending order of $\boldsymbol{m}$. At each node the primitive path $\boldsymbol{i}$ that gives the minimum is recorded. The sequence of primitive paths which gives $D(\boldsymbol{M})$ at node $\boldsymbol{M}=(M,N)$ is the optimal path.

### 3.2. Computational cost for intra-scanline search

The number of primitive paths which should be examined on a 2D search plane to compute $D(\boldsymbol{M})$ by equation (3-1) is $O(M^2N^2)$: the number of nodes in the search plane is $O(MN)$ and at each node we should examine $O(MN)$ primitive paths. Actually, we can limit the maximum disparity allowed in the matching. When the range is $d\times10^2$% of the width of the image, the number of nodes to be examined becomes $d\times M\times N$. Furthermore, we can limit the number of edges which can be ignored by a primitive path; it is unusual to skip many edges at a time. When this limit is $I$, the number of primitive paths examined at each node is about $I^2$. Thus, the number of primitive paths to be examined is $d\times M\times N\times I^2$. In the stereo images we have used in the experiments, $d$ is about 0.05~0.2, $M$ and $N$ are about 5~90 in average, and $I$ is set to 5. Thus, the number of primitive paths examined on each scanline ranges from about 70 for our simplest images to $7\times10^3$ for our most complicated ones.

As we can see in figure 3-1, our matching scheme is based on the intervals between edges rather than on the edges themselves as is Baker's [Baker 82]. This difference is mainly reflected in the difference of the definition of the cost function $\boldsymbol{d}()$. As will be described in section 4.2, our cost is based on the similarity of the intensities on right and left intervals, while Baker's is defined based on the similarity of edges such as contrast and orientation. As far as we discuss on the ability of the representation scheme for the search plane, there are no essential differences between the two; both can handle the cases for occlusions and/or noise edges. However, by using intervals as the matching unit, it is not necessary to treat an edge as a doublet to cope with occlusions; an interval is equivalent to a facing pair of half edges. When there are $N$ edges, there are only $N-1$ intervals while doublets give $2\times N$ half edges. This simplifies the representation of the search plane and has an advantage when dealing with complex images.

### 3.3. Inter-scanline search in 3D space

The problem of obtaining a correspondence between edges under the inter-scanline consistency constraint can be viewed as the problem of finding a set of paths in a 3D space which is a stack of 2D planes for intra-scanline search. Figure 3-2 illustrates this 3D space. The side faces of this space correspond to the right and left stereo images. We obtain an optimal set of paths satisfying the inter-scanline constraint. An optimal set is the one with minimal cost which is the sum of the costs of the individual paths in the set. A pair of connected edges in the right and left images make a set of 2D nodes in the 3D space when they share scanline pairs. We refer to these 2D nodes as a single 3D node. The optimal path on a 2D plane is obtained by iterating the selection of an optimal path at each 2D node. Similarly, the optimal set of paths in a 3D space is obtained by iterating the selection of an optimal set of paths at each 3D node. Connected edges, 3D nodes, and sets of paths between 3D nodes are illustrated in figure 3-2.

As described in section 2, the decision stages must be ordered in dynamic programming. In the intra-scanline search, their ordering was straightforward; it was done by ordering edges from left to right on each scanline. A similar consideration must be given to the inter-scanline search in 3D space. The decision stages are the 3D nodes. A 3D node is actually a set of 2D nodes, and the cost at a 3D node is computed based on the cost obtained by intra-scanline search on each 2D search plane. This leads to the following condition: *When we examine the correspondence of two connected edges, one in the right and one in the left image, the connected edges which are on the left of these connected edges in each image must already be processed.* A connected edge $u_1$ is said to be on the left of $u_2$, if all the edges in $u_1$ on the scanlines which $u_1$ and $u_2$ share are on the left of those in $u_2$. The "left-of" relationship is transitive; if there is a connected edge $u_3$ and $u_1$ is on the left of $u_3$ and $u_3$ is on the left of $u_2$, then $u_1$ is on the left of $u_2$. Figure 3-3 illustrates this concept. The order of two connected edges which do not satisfy both the relations in figure 3-3 may be arbitrarily specified. We assign an ordering index from left to right for every connected edge in an image. This ordering is possible without contradiction when a connected edge never crosses a scanline more than once and when two connected edges never intersect each other. The edge linking process must be performed, therefore, so that it does not make such connected edges.

Suppose we assign indices $[0:U]$ to connected edges according to the ordering in the right image, and $[0:V]$ in the left. The left and right ends of an image are treated as connected edges for convenience: the left ends are assigned index 0's.

Let $\boldsymbol{u}=(u,v)$ be a 3D node made by a connected edge $u$ in the right image and a connected edge $v$ in the left image. Let $C(\boldsymbol{u})$ be the cost of the optimal set of paths which come up to the 3D node $\boldsymbol{u}$. $C(\boldsymbol{u})$ is computed as follows:

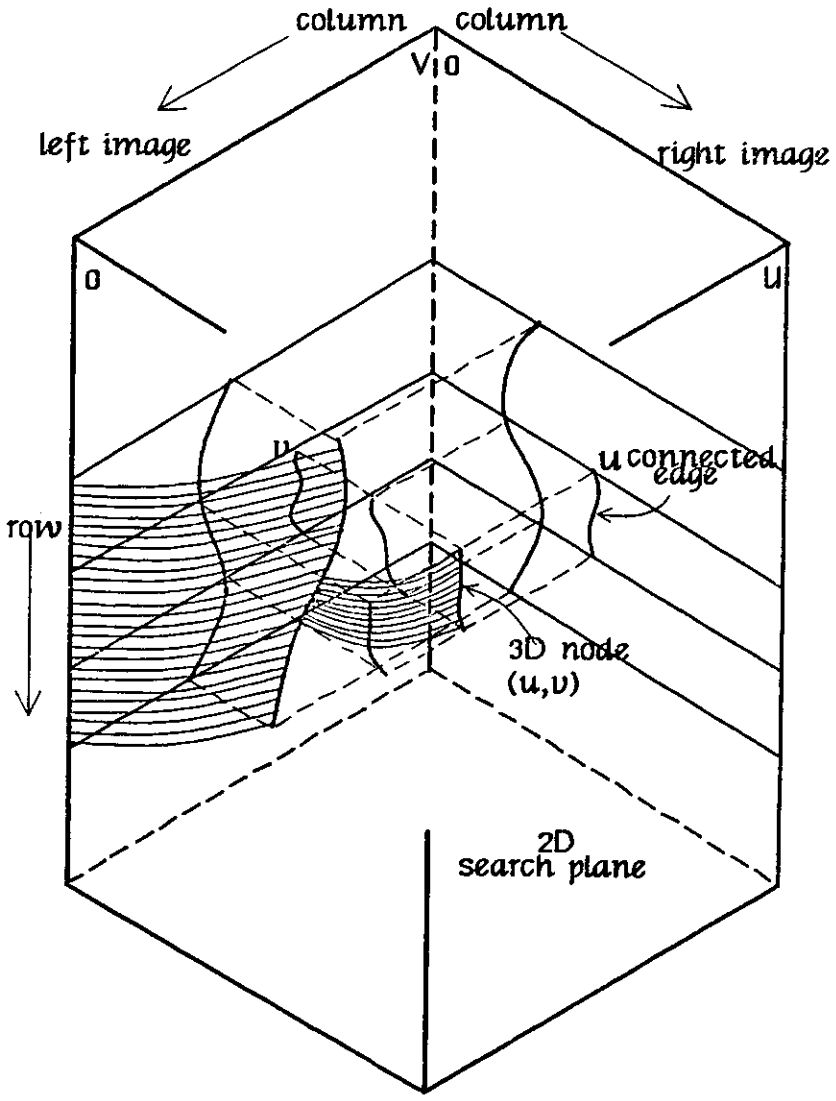



**Figure 3-2:** 3D search space for intra- and inter-scanline search.
*This may be viewed as a rectangular solid seen from above.*
*The side faces correspond to the right and left stereo images.*
*Connected edges in each image form sets of intersections (nodes)*
*in this space. Each set is called a 3D node.*
*Selection of a set of paths is done at every 3D node.*

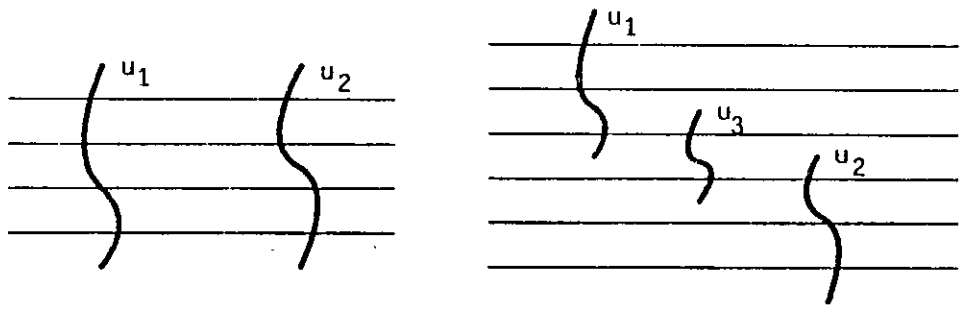



Figure 3-3: Connected edge $u_1$ is on the left of $u_2$.

$$C(\boldsymbol{u}) = \min_{\{i\}} \sum_{t=s(\boldsymbol{u})}^{e(\boldsymbol{u})} \{D(l(\boldsymbol{u};t), l(\boldsymbol{u}-i(t);t)+C(\boldsymbol{u}-i(t);t)\} \qquad (3-2)$$
$$C(\mathbf{0})=0, \quad \text{i.e. } C(\mathbf{0};t)=0 \text{ for all } t$$

here $\boldsymbol{u}=(u,v)$, $i(t)=(i(t),j(t))$, $0\le i(t)\le u$, $0\le j(t)\le v$, $i(t)+j(t)\neq 0$.

$C(\boldsymbol{u};t)$ is the cost of the path on scanline $t$ in the optimal set; that is, $C(\boldsymbol{u})$ is the sum of $C(\boldsymbol{u};t)$: $C(\boldsymbol{u})=\sum_{t=s(\boldsymbol{u})}^{e(\boldsymbol{u})} C(\boldsymbol{u};t)$. $D(\boldsymbol{m},\boldsymbol{k};t)$ is the cost of the optimal 3D primitive path from node $\boldsymbol{k}$ to node $\boldsymbol{m}$ on the 2D plane for scanline $t$. A 3D primitive path is a partial path between two 3D nodes on a 2D search plane and it has no vertices at the nodes belonging to a 3D node. So a 3D primitive path is a chain of 2D primitive paths and an intra-scanline search is necessary to obtain the optimal 3D primitive path on a 2D plane between given two 3D nodes. The function $l(\boldsymbol{u};t)$ gives the index of a node belonging to the 3D node $\boldsymbol{u}$ on the 2D plane for scanline $t$. The numbers $s(\boldsymbol{u})$ and $e(\boldsymbol{u})$ specify respectively the starting and ending scanlines between which the 3D node $\boldsymbol{u}$ exists. The cost $C(\boldsymbol{u})$ is minimized on the function $i(t)$. A 3D node $\boldsymbol{u}-i(t)$ gives the start node of the 3D primitive path on scanline $t$. The inter-scanline constraint is represented by $i(t)$. For example, if $i(t)$ is independent of $i(t-1)$, there are no constraints between scanlines and the search represented by equation (3-2) becomes equivalent to a set of intra-scanline searches which are performed independently on each scanline. Intuitively, $i(t)$ must be equal to $i(t-1)$ in order to keep the consistency constraint.

The iteration starts at $\boldsymbol{u}=(0,0)$ and computes $C(\boldsymbol{u})$ for each 3D node $\boldsymbol{u}$ in ascending order of $\boldsymbol{u}$. At each 3D

node the $i(t)$'s which give the minimum are recorded. The sequence of (2D) primitive paths which forms the 3D primitive path is also recorded on each scanline. The set of paths which gives $C(U)$ at the 3D node $U=(U,V)$ is obtained as the optimal set. $U$ is the 3D node formed by the right ends of stereo images. It should be noted that when there are no connected edges except for the right and left sides of the images, the algorithm (3-2) works as a set of intra-scanline searches repeated on each scanline independently. In this sense, the 3D algorithm completely contains the 2D one.

### 3.4. Computational cost for inter-scanline search

The number of 2D primitive paths which should be examined in the 3D search space to compute $C(U)$ by equation (3-2) is $O(TU^2V^2M^2N^2)$: the number of 3D nodes in the search space is $O(UV)$, at each 3D node we should examine $O(UV)$ sets of 3D primitive paths, each set has $O(T)$ paths, and each 3D primitive path requires $O(M^2N^2)$ computation for intra-scanline search. $T$ is the number of scanlines in the image. This means the search in 3D space requires $U^2\times V^2$ times the computation of the 2D search. Although this still seems to be a prohibitive amount of computation, the situation is much better for four main reasons.

First, a connected edge is much shorter than $T$ for most cases. Let the average length be $a\times T$, then the average number of connected edges crossing a scanline is $a\times U$ in the right image and $a\times V$ in the left image. The number of 2D nodes which are made by those edges is $a^2\times U\times V$. Because we have $T$ scanlines, the total number of 2D nodes which belong to a 3D node is $a^2\times U\times V\times T$. Second, we can set the disparity range allowable in the matching as in the intra-scanline search. This reduces the number of nodes to $d\times a^2\times U\times V\times T$ where $d$ is the fraction of disparity range to the width of the image. Third, in order to find the best $i(t)$, we can use beam search [Lowerre 76]; On the first scanline, *i.e.* scanline $s(n)$, we should examine every 3D primitive path and select $W$ paths from the best. On many of the succeeding scanlines we need to examine only $W$ paths because $i(t)$ is usually equal to $i(t-1)$, and the average will come to $W$. The total number of 3D primitive paths examined in the 3D search space is $d\times a^2\times U\times V\times T\times W$.

The final reason is that the search plane for each 3D primitive path is usually much smaller than the whole 2D plane for intra-scanline matching. When there are $M$ (or $N$) edges on a scanline and $a\times U$ (or $a\times V$) of them belong to connected edges, the average number of edges between two neighboring connected edges is $M/aU$ (or $N/aV$). Most 3D primitive paths are searched on this small area. The number of 2D primitive paths examined in it is $(M\times N\times I^2)/(a^2\times U\times V)$, where I is the limit for edge ignorance. Thus, the total number of 2D primitive paths to be examined is estimated as $d\times T\times W\times M\times N\times I^2$. This is only $W$ times that of the 2D search shown in the previous section and $W$ is typically set to 5. We always took the lower bounds in the estimation above and the actual value will be higher. But, the estimation suggests the search can be

performed with a feasible amount of computation even in the 3D search algorithm.

### 3.5. Consistency constraints in inter-scanline

Using the term 3D node defined in the previous section, we can describe the inter-scanline consistency constraints as follows: *For any 3D node, either all corresponding 2D nodes are the vertices on the set of paths in the 3D search space or all are not the vertices on the set of paths.* We need to represent this constraint as the relation between $i(t)$ and $i(t-1)$ in equation (3-2). Consider the example in figure 3-4. Suppose we are trying to obtain a set of 3D primitive paths which come up to node $u$. In order to satisfy the consistency constraints above, all the starting points of these paths should be the same 3D node; that is $i(t)=i(t-1)$. The cases when the starting point may be a different 3D node are shown as case2 and case3 in the figure. In case2, a new 3D node appears at scanline $t$ and the starting point changes to the new one. Of course, it is possible that the starting point does not change to the new 3D node. This will happen if the cost of the paths having vertices on the 3D node is higher than the cost of the paths not having vertices on it. In case3, the 3D node $u-i(t-1)$ disappears on scanline $t$ and the starting point is forced to move elsewhere.

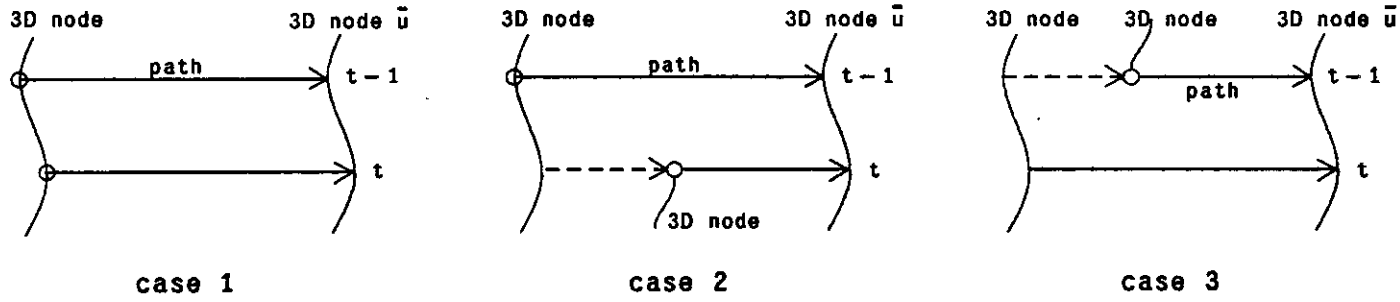



**Figure 3-4:** Three cases for consistency constraint.

Let us denote the 3D node $u-i(t)$, which is the starting point of the 3D primitive path coming up to 3D node $u$ on scanline $t$, as $frm(u;t)$. Then the following rules should be satisfied in each case.

$$\begin{aligned} &\text{case1: } frm(u;t)=frm(u;t-1) \\ &\text{case2: } frm(frm(u;t);t)=frm(u;t-1) \\ &\text{case3: } frm(u;t)=frm(frm(u;t-1);t-1) \end{aligned} \tag{3-3}$$

The rules in case2 and case3 require that the decision at 3D node $\boldsymbol{u}$ depend on decisions at preceding 3D nodes. Unfortunately, a decision system with such a property is not *Markovian* as described in section 2, and therefore there is no guarantee of obtaining an optimal solution by using dynamic programming. This means if we search for a solution using dynamic programming with those rules, the result might be poorer than that of the 2D algorithm.

In order to assure optimality in dynamic programming, the rules are modified as follows.

$$\begin{aligned} &\text{case1: } frm(\boldsymbol{u};t)=frm(\boldsymbol{u};t-1) \\ &\text{case2: } frm(\boldsymbol{u};t)\geq frm(\boldsymbol{u};t-1) \\ &\text{case3: } frm(\boldsymbol{u};k)\leq frm(\boldsymbol{u};t-1) \\ &\qquad k\leq t \;\&\; k\in\{\textit{scanlines on 3D node } frm(\boldsymbol{u};t)\}. \end{aligned} \tag{3-4}$$

The new rule for case2 requires the new 3D node on scanline $t$ should be on the right of the 3D node that is the starting point on scanline $t-1$. For case3, the starting 3D node on scanline $t$ should be on the left of that on scanline $t-1$, and all starting points of the 3D node $\boldsymbol{u}$ should never be on the left of it. The new rules are always satisfied when the rules in equation (3-3) are satisfied. But the converse is not true! Thus under the new rules, the consistency constraint might not be satisfied at all places. In other words, the constraints represented by the rules in equation (3-4) are weaker than those of equation (3-3). However, since we can expect to obtain an optimal solution in dynamic programming, we can expect better results by the 3D search algorithm than by the 2D search algorithm.

# 4. Implementation

## 4.1. Detection, linking, and ordering of edges

The edge based stereo algorithm requires that edge detection and linkage be performed on the images beforehand. We will give a brief outline of the algorithm we adopted for completeness of the description, but the algorithm may be replaced with another one.

Edges running across the scanlines are useful for obtaining correspondence in stereo. The positions of edges are detected by using an intensity profile along a scanline, whose first derivatives are computed. The peaks and valleys in it whose absolute values exceed a threshold are extracted as edge locations. We use several operators with different sizes to compute the first derivatives, as shown in figure 4-1, and the results obtained by these operators are combined. Because the smaller operators can locate edges more accurately than larger ones, edge positions located by smaller operators are given priority. That is, edge positions extracted by a larger operator are adopted only when no edges are extracted by smaller ones within the range covered by the operator. This prevents an edge from being detected more than once at slightly different positions by operators of different sizes.

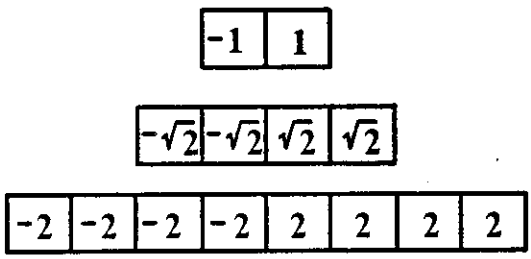
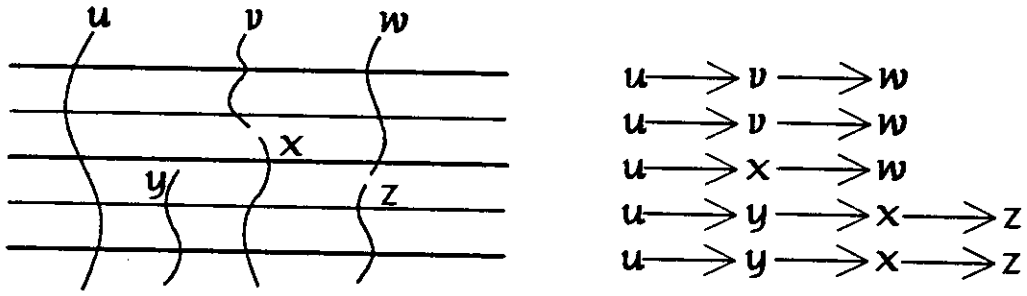


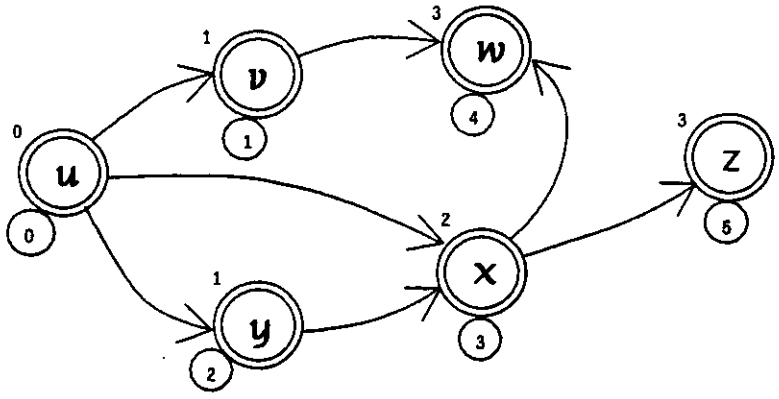

**Figure 4-1:** Operators for edge detection.
*Results of different size operators are combined.* ... i path.

The linking process links the edge positions into connected edges. However, an edge running nearly horizontally is detected as a set of positions which are apart on consecutive scanlines. This is inconvenient for linking them into a connected edge. Thus we detect edge positions by using operators rotated 90 degrees from those in figure 4-1 and the linking process uses both results to obtain connected edges. We adopt only the connected edges which are longer than a threshold, while the others are kept as isolated ones. Both are used in the stereo matching.

Ordering of connected edges is done by the following four processes. First, connected edges which run across the same scanline are locally ordered from left to right. This is done independently on every scanline. Figure 4-2 (a) illustrates this ordering. Second, a graph representing this local order is generated as shown in (b). Nodes in the graph are the connected edges and directed arcs show the local ordering between them. Third, for each node, the maximum number of arcs from the leftmost node to that node is assigned. The numbers are shown on the left shoulder of each node in the figure. If a connected edge crosses a scanline more than once or if two connected edges cross each other, loops are formed in the graph and the maximum goes to infinity. Our linking process was designed, therefore, not to make such connected edges. Finally, the maximum numbers are used to impose a global ordering on the connected edges. The ordering among the connected edges which are given the same number is arbitrary, and we have assigned smaller indices to those which are found earlier when scanning the image from top to bottom. The ordering indices assigned in this way are shown with circled numbers in figure 4-2 (b).




(a) connected edges and their local orders




(b) global ordering

**Figure 4-2:** Ordering of connected edges

### 4.2. Metrics for similarity measure

The computation of cost in our search algorithm is based on the cost of a primitive path on the 2D search plane. We define the cost of a 2D primitive path as the similarity of a right and a left interval on a scanline pair. Let $a_1 \ldots a_k$ and $b_1 \ldots b_l$ be the intensity values of the pixels which comprise the two intervals. Then the mean and variance of all pixels in the two intervals are computed as:

$$m = \frac{1}{2} \times \{ \frac{1}{k} \sum_{i=1}^{k} a_i + \frac{1}{l} \sum_{j=1}^{l} b_j \} \qquad (4-1)$$

$$\sigma^2 = \frac{1}{2} \times \{ \frac{1}{k} \sum_{i=1}^{k} (a_i - m)^2 + \frac{1}{l} \sum_{j=1}^{l} (b_j - m)^2 \}$$

In the definition above, both intervals give the same contribution to the mean $m$ and variance $\sigma^2$ even when their lengths are different. The cost of the primitive path which matches those intervals is defined as follows:

$$\text{cost} = \sigma^2 \times (k^2 + l^2)^{1/2} \qquad (4-2)$$

Intuitively, the meaning of this cost definition can be explained as follows: The pixels in two intervals which matched each other are assumed to have come from a homogeneous surface in the 3D scene and must have similar intensities. That is, their variance should be small. If we consider those pixels forming a cluster in a feature space, the variance may be called *within-class variance.* As described in section 3.1, some edges may be ignored in the matching process. Ignoring an edge means that two intervals divided by that edge are merged. So the intra-scanline search simultaneously tries to find the best segmentation of a scanline pair and the best matching between those segments to form an optimal set of clusters which minimizes the sum of within-class variances.

The definition in equation (4-1) cannot be applied to horizontal or vertical primitive paths which correspond to occlusions. As illustrated in figure 4-3 a horizontal or vertical path means to leave the interval marked with × unmatched and it can be considered as the consequence of avoiding the two paths drawn with dotted lines. Therefore its cost should be defined as a function of the costs of those two paths. Let $\sigma_1^2$ and $\sigma_2^2$ be the variances defined in equation (4-1) for those paths. The cost of a vertical (horizontal) primitive path is defined as follows:

$$\text{cost} = k \times f((\sigma_1^2 + \sigma_2^2)/2;\ th) \qquad (4-3)$$

Here $k$ is the length of the primitive path, $th$ is a threshold, and $f$ is a function as shown in figure 4-4. When $(\sigma_1^2 + \sigma_2^2)/2$ is small, the function $f$ gives a high cost and when $(\sigma_1^2 + \sigma_2^2)/2$ is large, the function $f$ gives a low cost. The minimum limit of $f$ is determined by $th$.

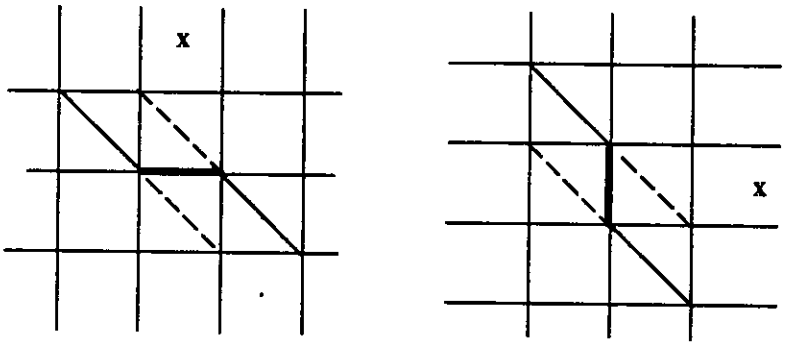



**Figure 4-3:** Primitive paths for occlusion.
*Cost of horizontal/vertical path is defined based on those of dotted paths.*

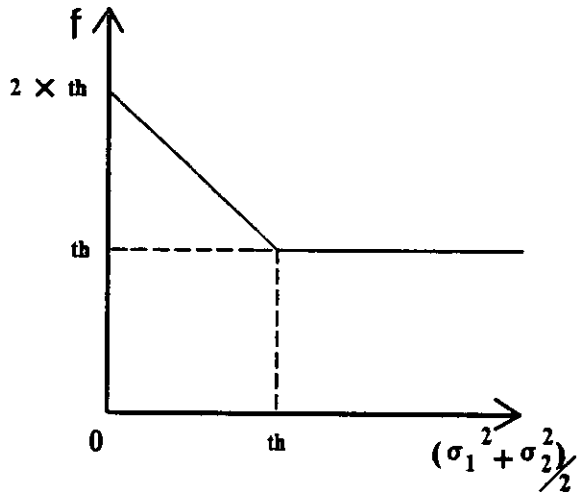



**Figure 4-4:** Mapping function for the cost of horizontal/vertical path.

# 5. Experimental Results

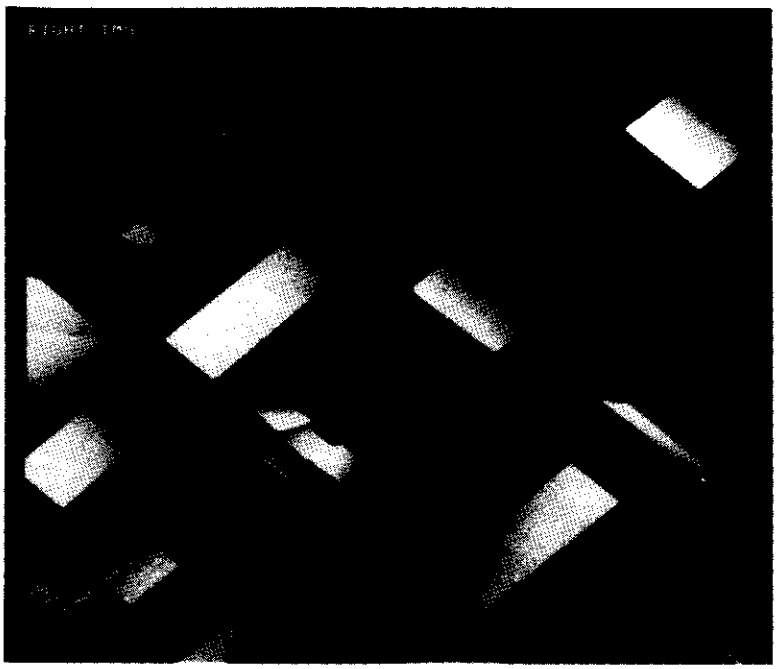
We have applied our stereo algorithm to images from various domains including synthesized images, urban aerial images, and block scenes.

## 5.1. Synthesized images

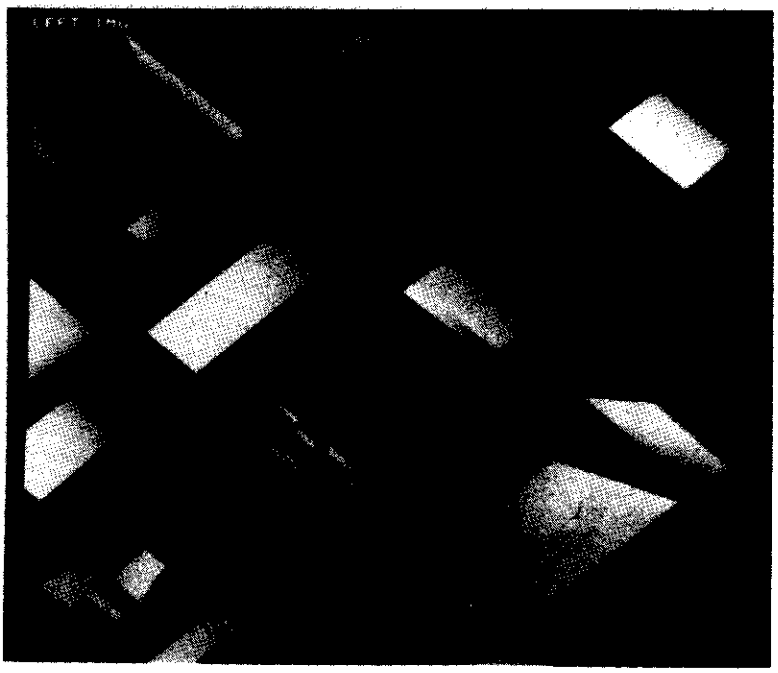
We first applied our stereo algorithm to the synthesized stereo image pair shown in figure 5-1 which is from Control Data Corporation, and which has been used by Baker [Baker 82]. The image size is 256×206 pixels. We extracted every position where the intensity changes as edges. Figure 5-2 shows the edges thus extracted. Some edges in this image have very weak contrast; that is, the difference of intensity is only one grey level, and it is impossible to distinguish them from pseudo edges which arise from digitization. Our stereo algorithm can ignore the pseudo edges when they do not correspond into anything in the other image. Actually, however, we found that almost all pseudo edges in the right and left images do match because the images are synthetic. Figure 5-3 displays the connected edges obtained from figure 5-2. The number attached to each connected edge indicates its ordering index.

Figure 5-4 illustrates a typical matching path obtained on a 2D search plane for scanline 207. The representation scheme of the plane is almost the same as in figure 3-1. The positions of connected edges are indicated by thicker lines and their indices are attached. The thinner lines indicate isolated edges. The path shown by dotted lines is obtained when we do not use the inter-scanline constraint. Note that using the constraint in the 3D search space results in vertices at the two nodes belonging respectively to the 3D nodes (41,42) and (50,44).

Figure 5-5 is a perspective view of edges which are matched in the 3D search space. Figure 5-6 shows the disparity map. The map is registered in the coordinates of the right image; that is, each pixel position in the right image is assigned its disparity value. The higher the elevation, the darker the tone is shown in the disparity map. The black mat shows regions where the disparity cannot be obtained because of occlusion. For those points which do not correspond to edges, the disparity is assigned by interpolation. The following simple interpolation scheme is used here. On each scanline, a linear interpolation is done between neighboring edge positions where the disparity is obtained. That is, the linear primitive paths which run from corner to corner on the 2D search plane shown in figure 3-1 or figure 5-4 illustrate the interpolation scheme. It should be noted that we did not apply any smoothing operation to the disparity maps which are displayed in this paper. Figure 5-7 is an isometric plot of the disparity map.

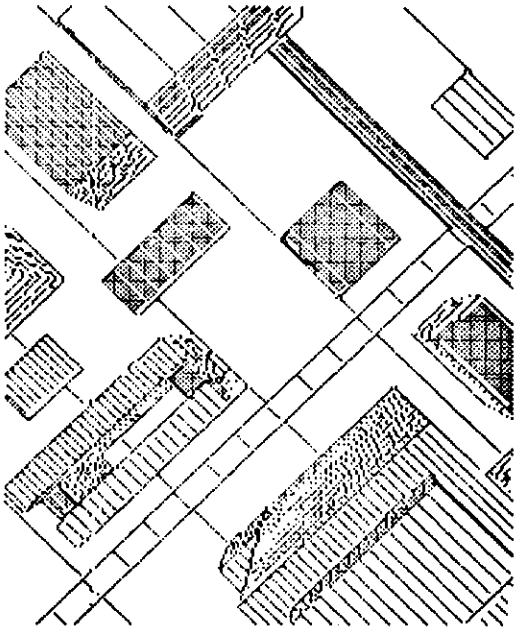


right image

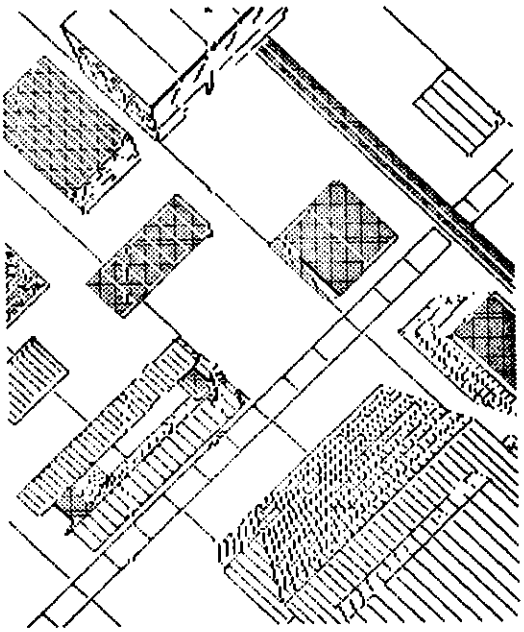


left image

**Figure 5-1:** The "cdc" synthesized stereo image pair.

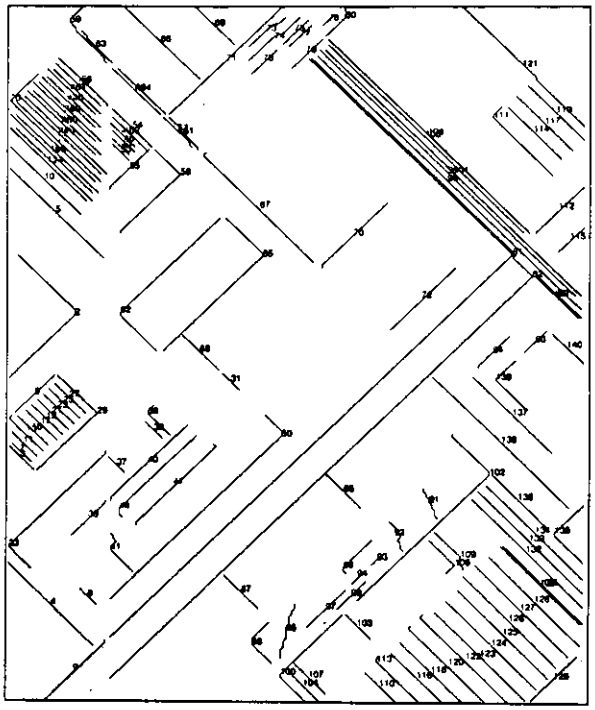


right image

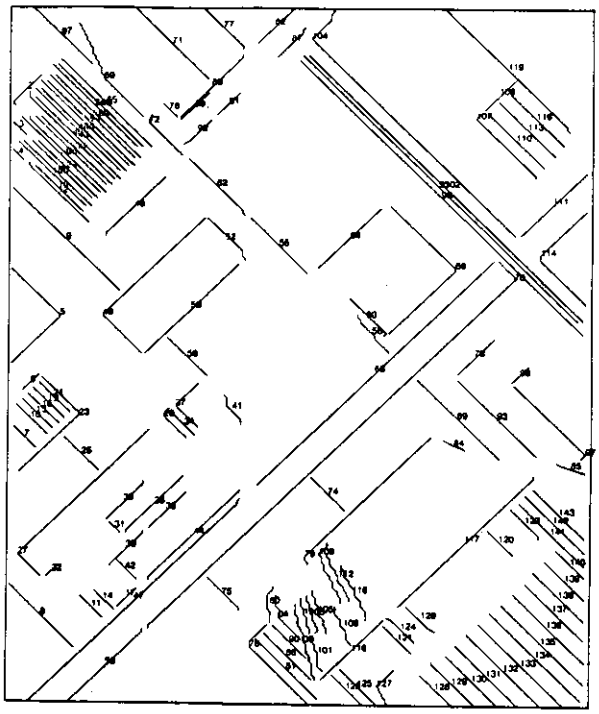


left image

**Figure 5-2:** Edges extracted on images in figure 5-1.
*Only the edges extracted by horizontal operators are displayed.*



right image



left image

**Figure 5-3:** Connected edges obtained from figure 5-2.
*The numbers attached are ordering indices.*

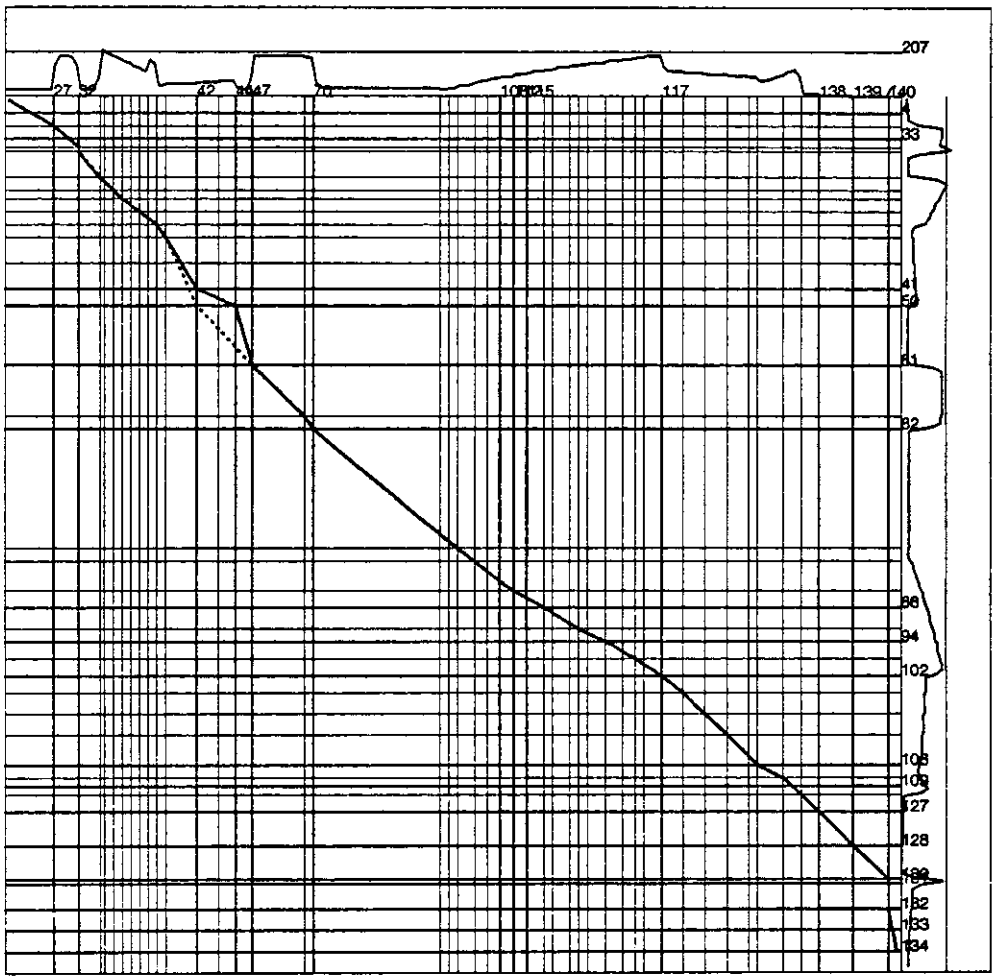



**Figure 5-4:** A typical matching path on a 2D search plane.
*Scanline 207 on images in figure 5-1.*

### 5.2. Urban aerial images

The stereo pairs used here are aerial images of the Washington, D.C. area. The first one is "pentagon" and the second one is "white house". They are rectified based on camera models computed by Gennery's program [Gennery 79] using manually selected point pairs.

Figures 5-8, 5-9, and 5-10 are the original stereo pair, edges, and connected edges, respectively. The image size is 512×512 pixels and the intensity resolution is 8 bits. The number of edges extracted is about 40,000 in each image. The number of connected edges is about 400 in each image. Figure 5-11 (a) and (b) show the disparity maps obtained by 2D search and 3D search, respectively. These maps are registered in the left image coordinates. We can see that the detailed structures of the roof of the building and the bridge over the highway are clearly extracted. Figure 5-12 displays an isometric plot of the disparity map.

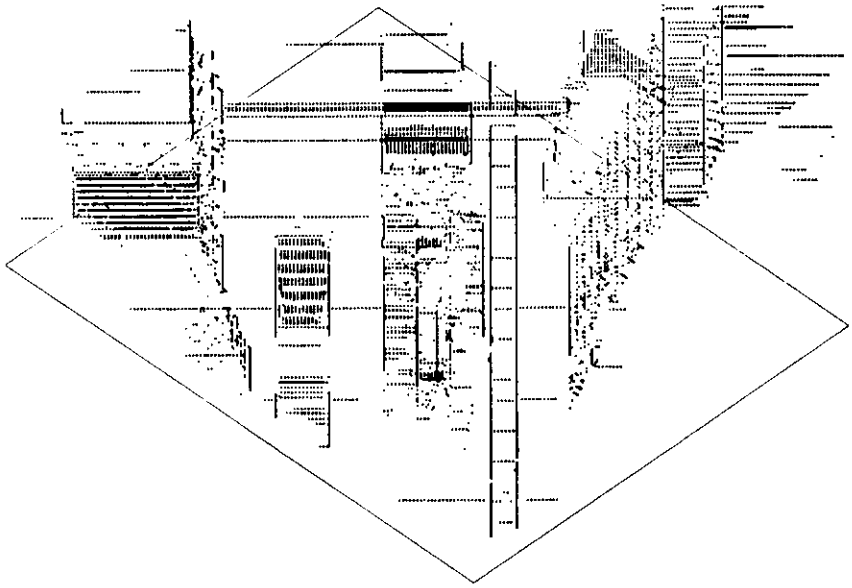


**Figure 5-5:** A perspective view of matched edges obtained from figure 5-2.
*Viewed from lower left corner.*

Figure 5-13 emphasizes the difference between the two disparity maps obtained by 2D search and 3D search. We can see that many local mismatches on the roof of the building in figure 5-11 (a) are corrected in (b). We also counted the number of positions where the consistency constraint, described in equation (3-4), is not satisfied. It is 372 in the 2D search and 27 in the 3D search. These numbers quantitatively show a significant improvement achieved by the 3D search algorithm. The reason why the inconsistency is not zero in the 3D case is that we used "weaker" rules for the constraint.

The image in figure 5-14 is the "white house" stereo pair. This example is interesting because it includes both buildings and highly textured trees. Figures 5-15 and 5-16 show the edges and connected edges, respectively. As we may expect, many connected edges are obtained around the building while few are obtained in the textural part. The disparity maps obtained by the 2D and 3D search algorithms are shown in figure 5-17. The maps are in the right image coordinates. Therefore, the disparity values for pixels on the right wall of the central building, which is visible in the right image but occluded in the left, are undetermined. Figure 5-18 shows the differences between the two maps. Considerable improvements can be observed at the boundaries of buildings. In the textural part, the two algorithms provide approximately the same results. The number of inconsistencies in the result of the 3D search is 32 while that in the 2D search is 436. Figure 5-19 shows an isometric plot of the disparity map in which the buildings and the trees are

**Figure 5-6:** Disparity map for figure 5-1.
*This map is registered in the right image coordinates.*
*Higher elevation is displayed darker.*
*For the black mat area disparity was not obtained.*

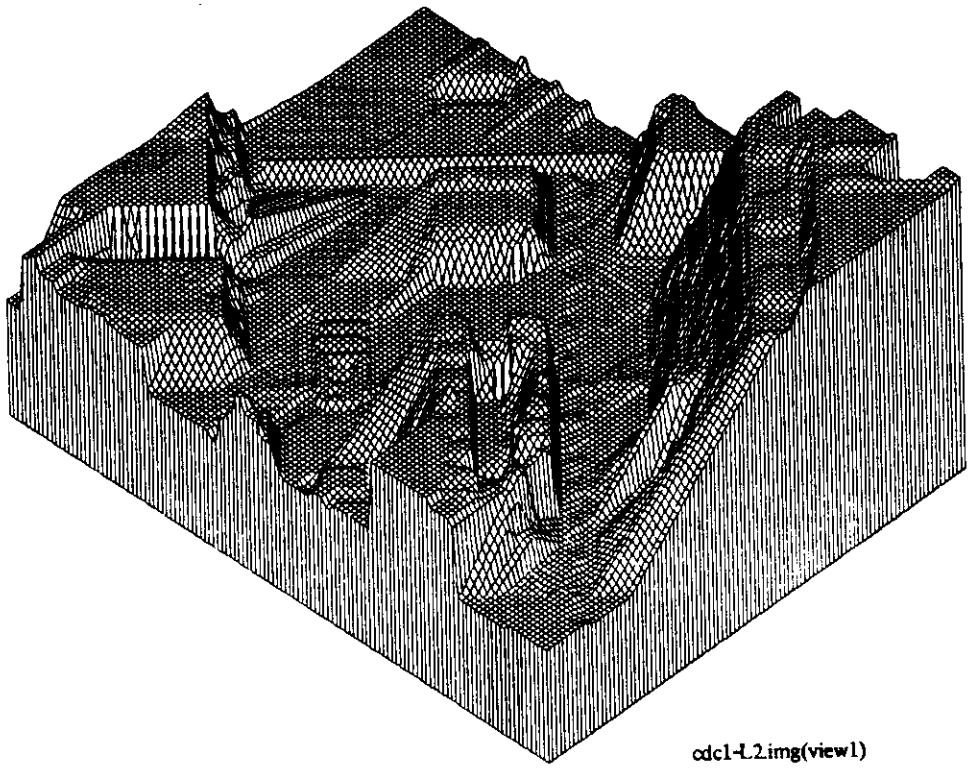
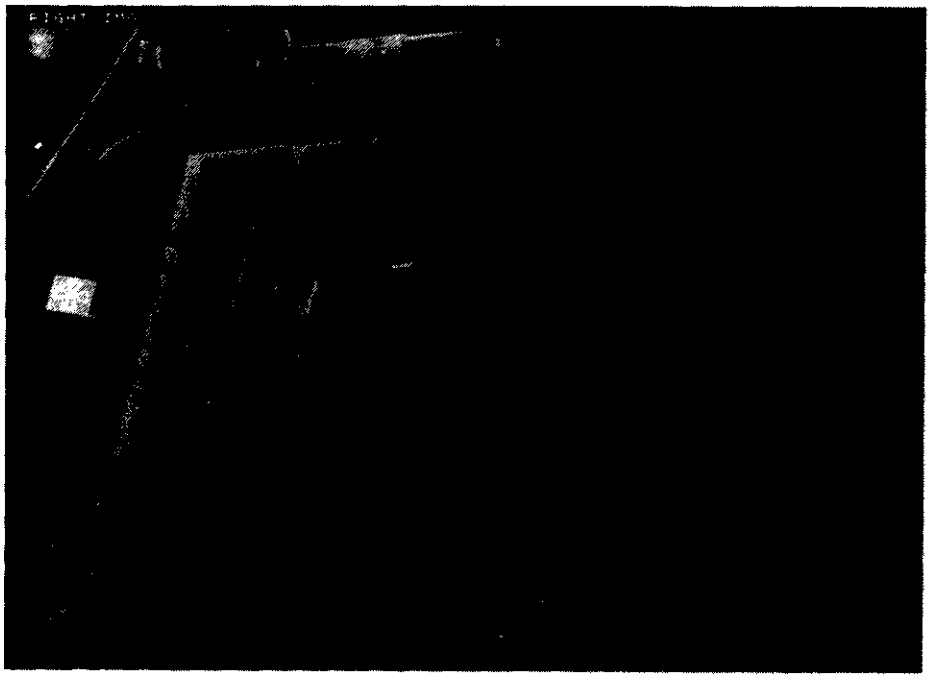
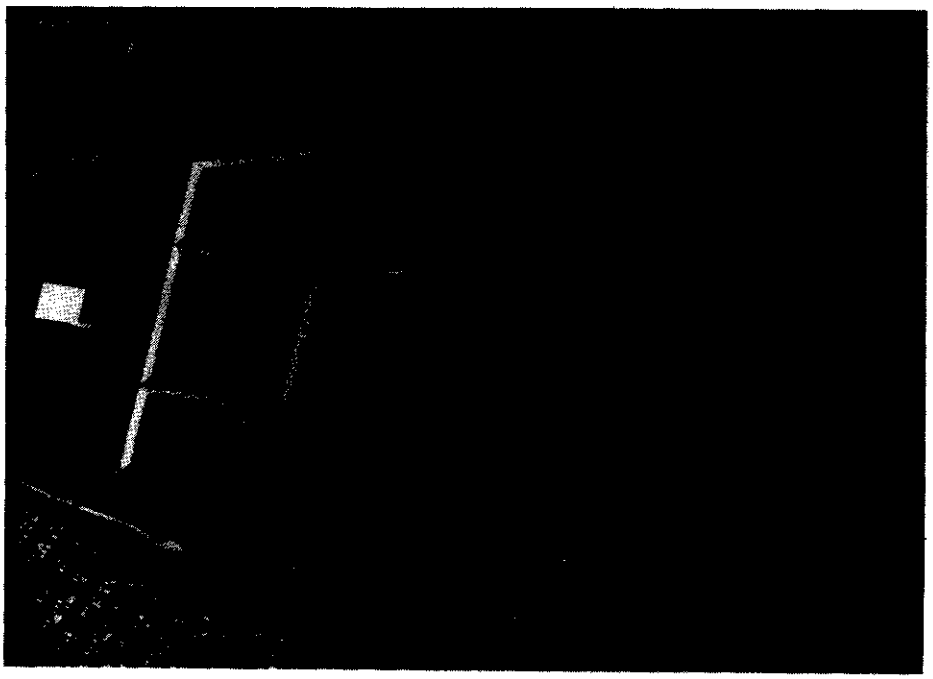



**Figure 5-7:** An isometric plot of the disparity map.



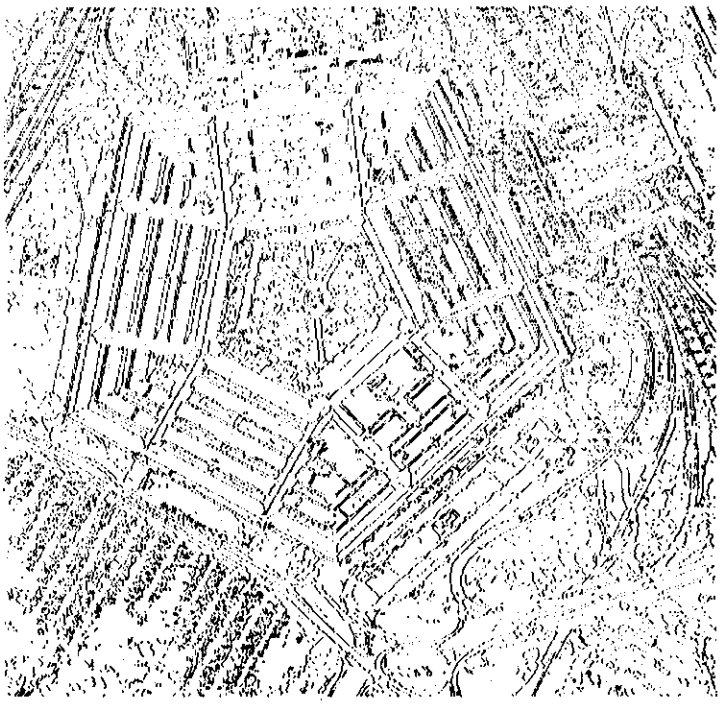
right image



left image

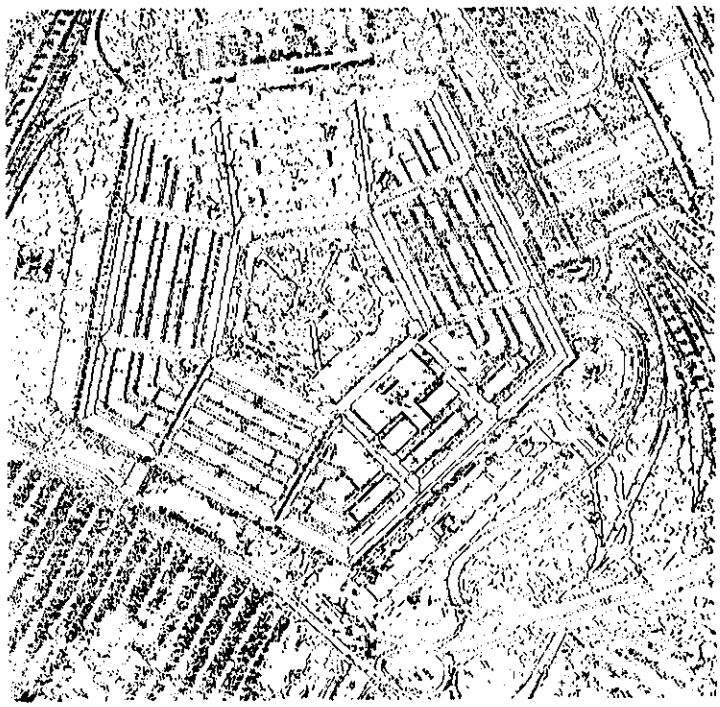
**Figure 5-8:** The "pentagon" stereo pair of urban aerial images.

/visb/yxo/38011-2/pentagon/right.imgE



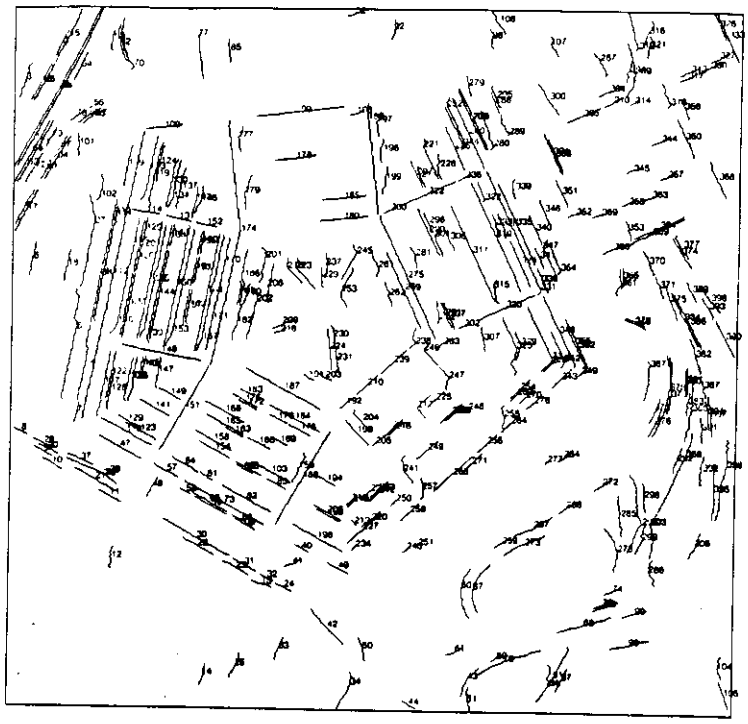
right image

/visb/yxo/38011-2/pentagon/left.imgE

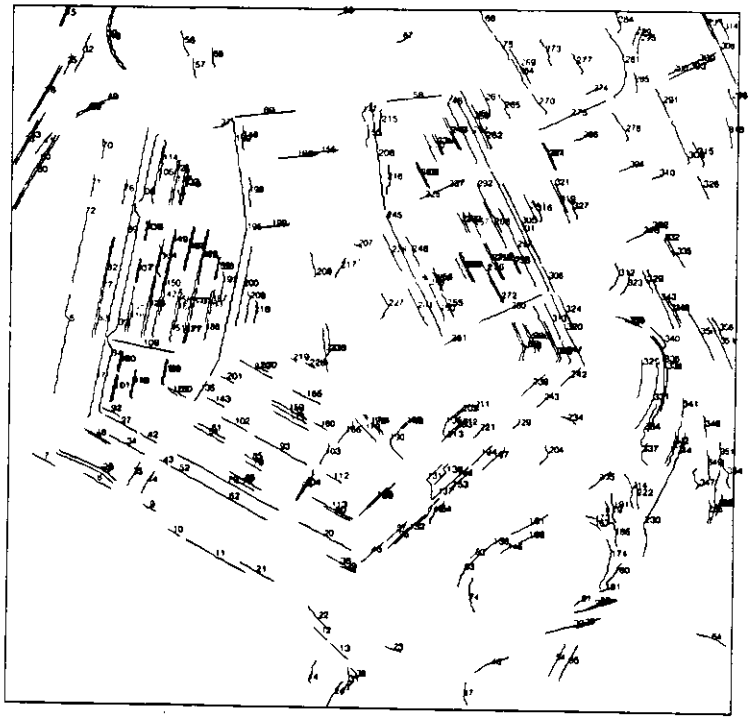


left image

**Figure 5-9:** Edges extracted on images in figure 5-8.

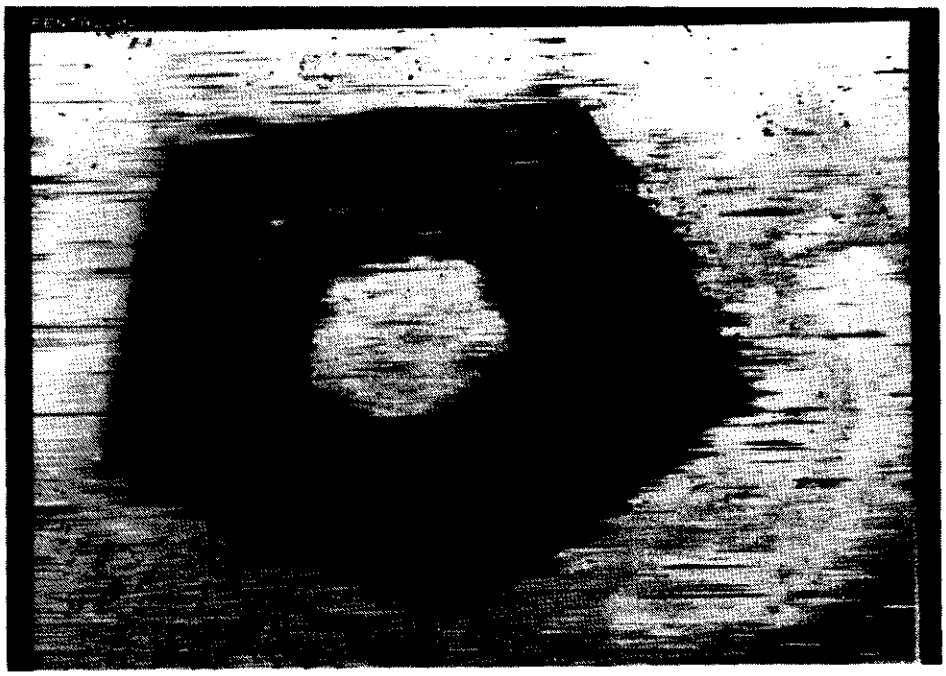


right image

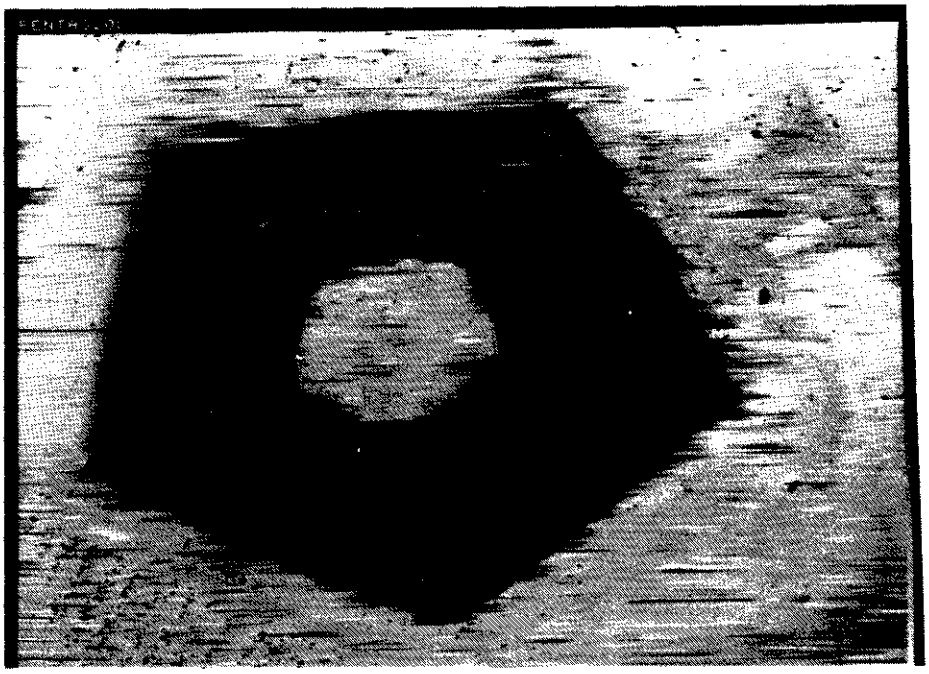


left image

**Figure 5-10:** Connected edges obtained from figure 5-9.



(a) result of 2D search



(b) result of 3D search

**Figure 5-11:** Disparity map obtained for figure 5-8.
*Both are in the left image coordinates.*
*Notice the detailed structures of the roof of the building and the bridge over the highway (upper left corner).*

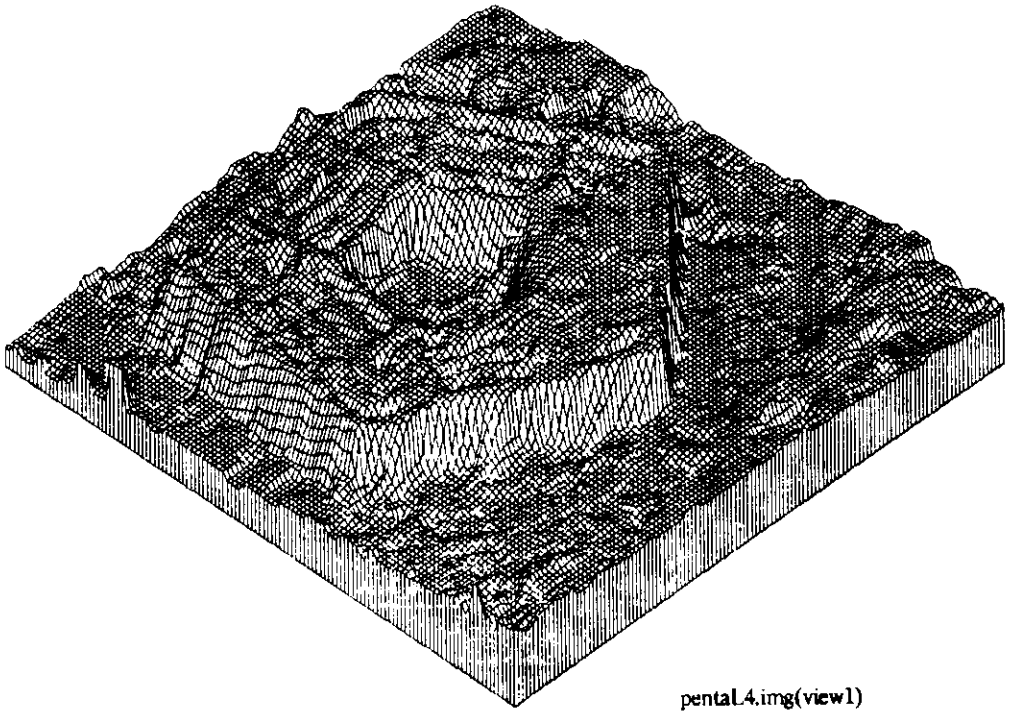



**Figure 5-12:** An isometric plot of the disparity map.
*Viewed from lower left corner.*

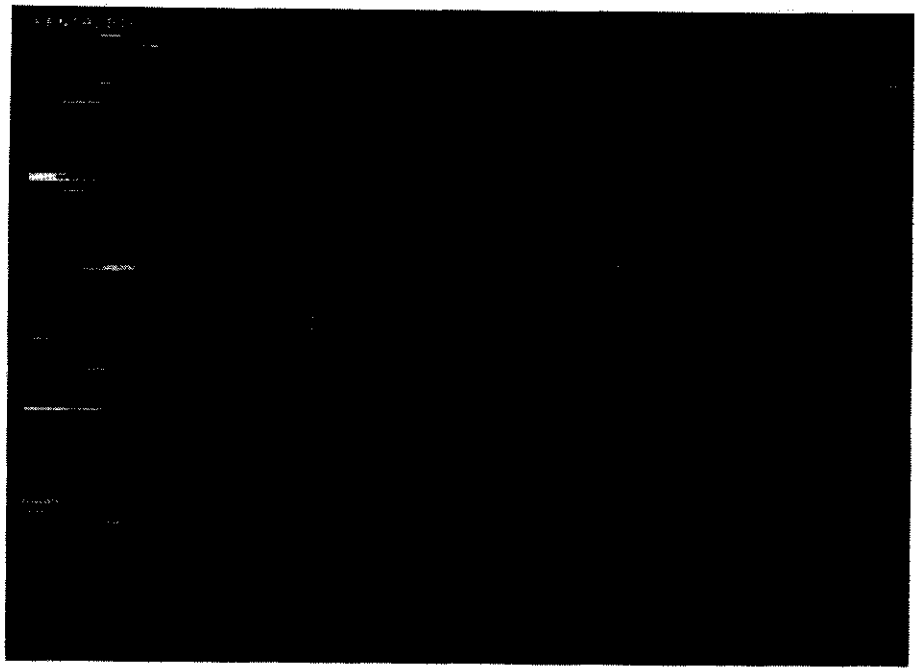


**Figure 5-13:** Difference of figure 5-11 (a) and (b).

extracted well.

### 5.3. Block scenes

We also applied our program to block scenes (obtained by courtesy of the University of Southern California). Actually, these images are not exactly rectified; there are discrepancies of a few scanlines between corresponding point pairs, but we ignored them in the following experiments.

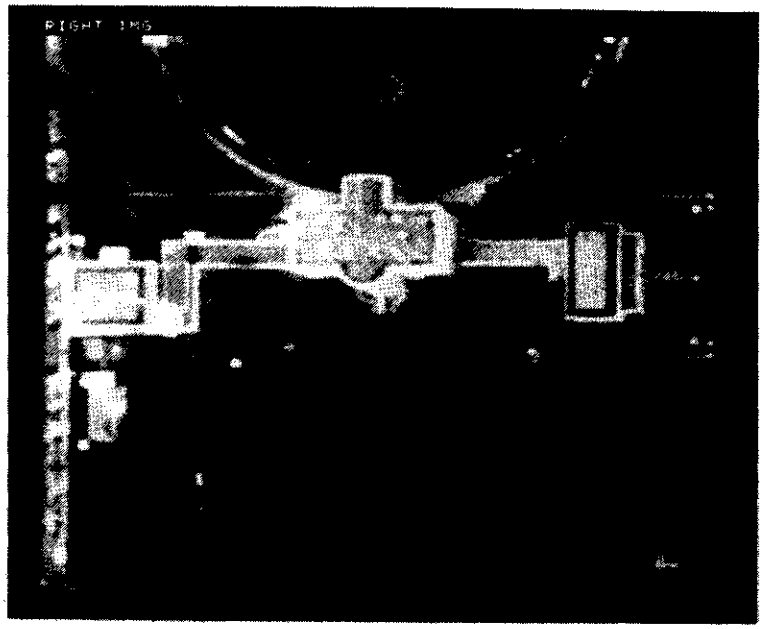
Figures 5-20 through 5-22 show an original image pair, their edges, and connected edges. The image size is 512×512 pixels. Figure 5-23 displays the perspective view of the matched edges. The disparities of the edges on the sphere in front of the blocks and on the small block behind the arch are correctly obtained.

Figures 5-24 through 5-26 show another block scene whose image size is also 512×512 pixels. The disparity range for this image is about 20% of the image width. The numbers of edges and connected edges are respectively about 5,000 and 90 in each image. The numbers are much smaller than in the aerial images. However, the number of inconsistencies in the result of the 2D search is 269 which is almost the same as that in the aerial images. Most inconsistencies occurred at the Rubik cube where repetitive patterns cause many ambiguities. In the 3D search, the inconsistencies are reduced to 36. Figure 5-27 displays the perspective view of the matched edges.

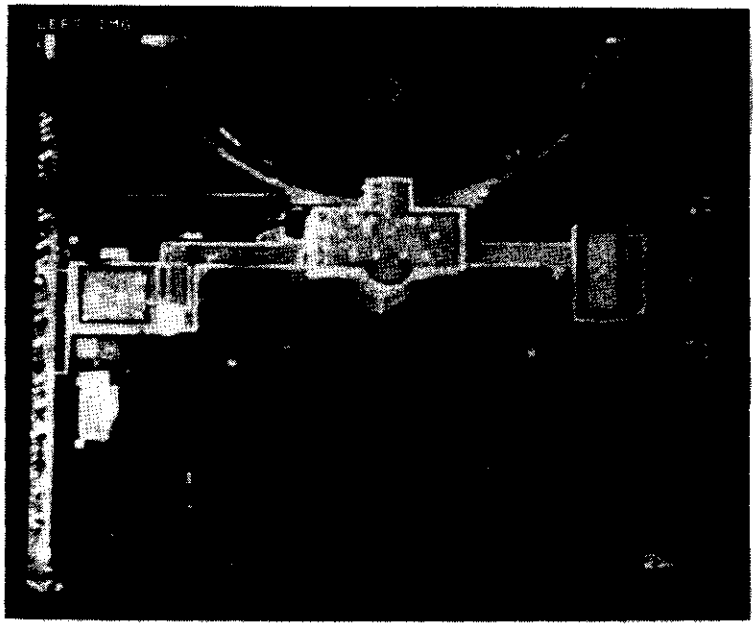
### 5.4. Summary of the experiments

Table 5-1 summarizes the stereo images used in the experiments. It shows the image size, the number of edges extracted in each image, the number of connected edges obtained in each image, the disparity range used in the search, the number of inconsistencies that occurred in the 2D and 3D search, and the CPU time of VAX11/780 required to obtain the whole disparity map.

The CPU time varies from one image to another. Perhaps the most complicated image pair is the "pentagon", where left image has an average of 90 edges on each scanline. It takes 52 min for the 2D search algorithm and 858 min for the 3D search algorithm to obtain a disparity map. The "rubik" image pair has the largest disparity range. It is about 20% of the width of the images. The simplest image pair "arch", which is still fairly complicated, requires only 2 min for the 2D search or 7 min for the 3D search.

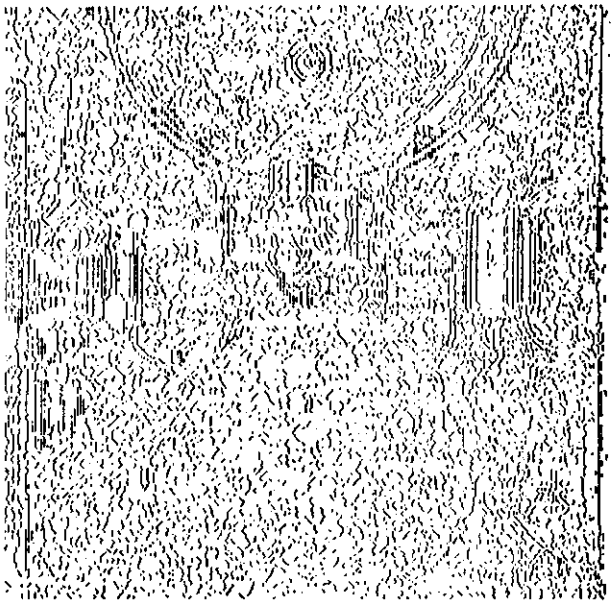


right image

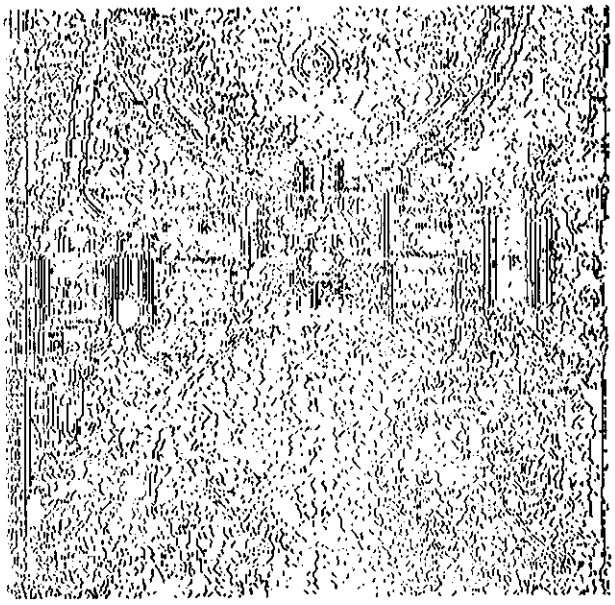


left image

**Figure 5-14:** The "white house" stereo pair of urban aerial images.

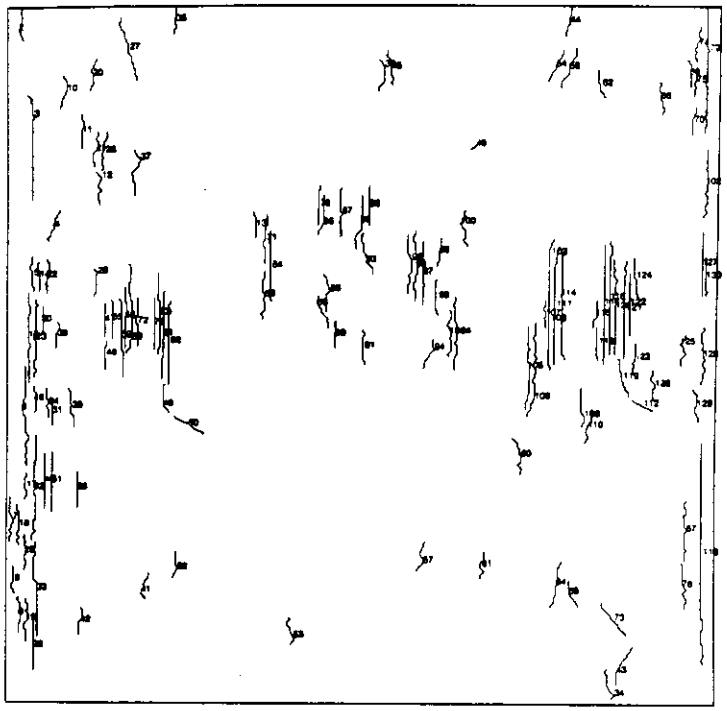


right image

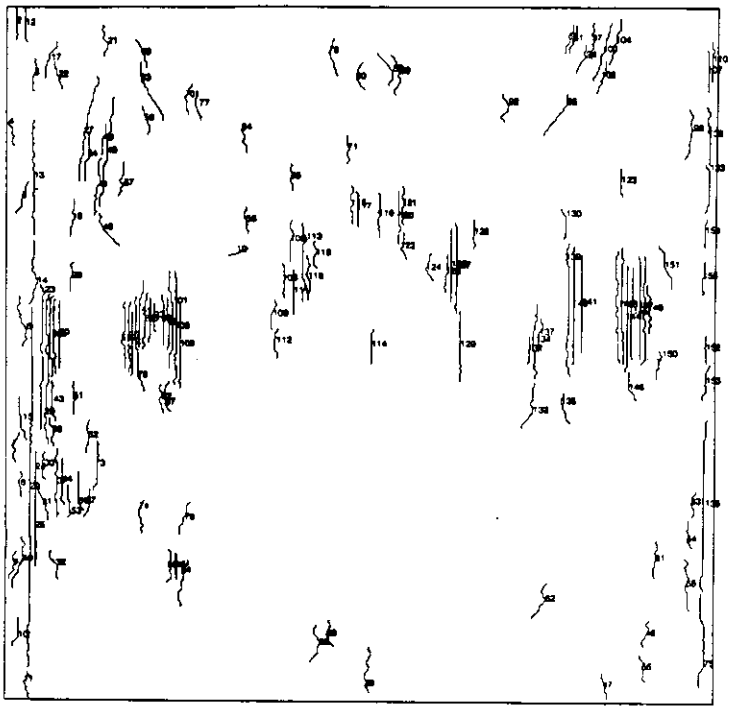


left image

**Figure 5-15:** Edges extracted from the images in figure 5-14.

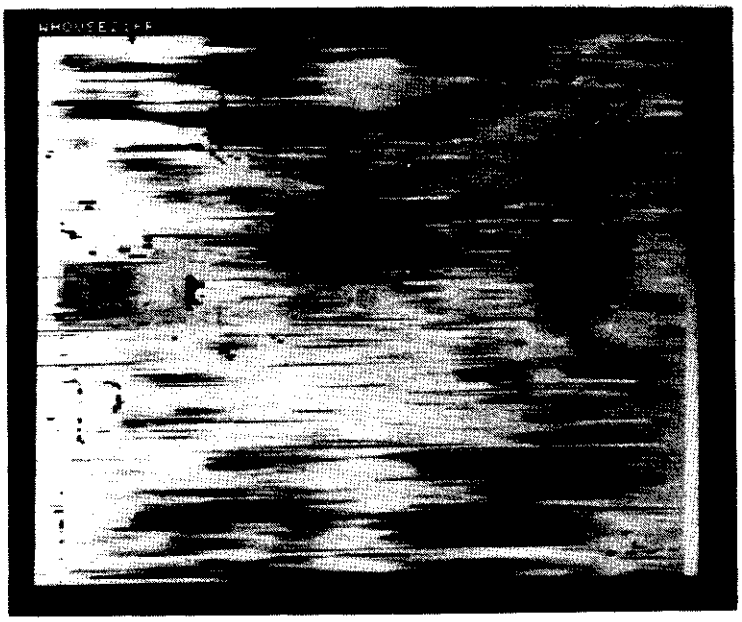


right image

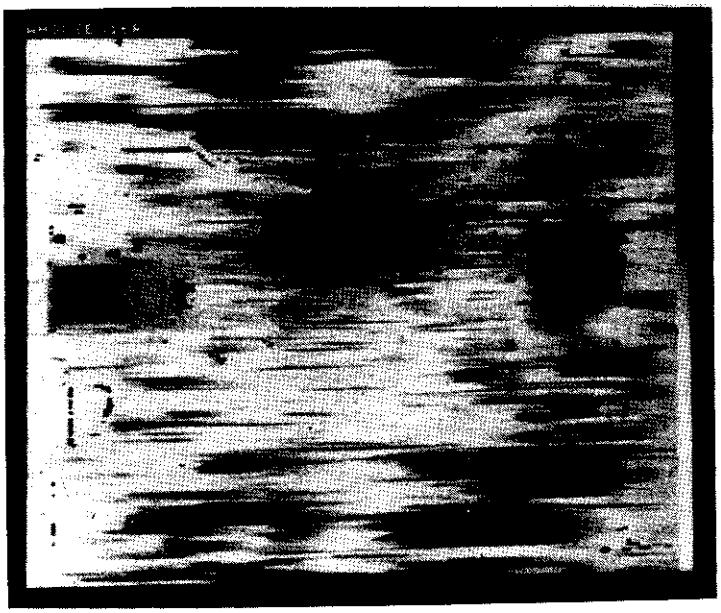


left image

**Figure 5-16:** Connected edges obtained from figure 5-15.



(a) result of 2D search



(b) result of 3D search

**Figure 5-17:** Disparity map obtained for figure 5-14.
*Both are in the right image coordinates.*

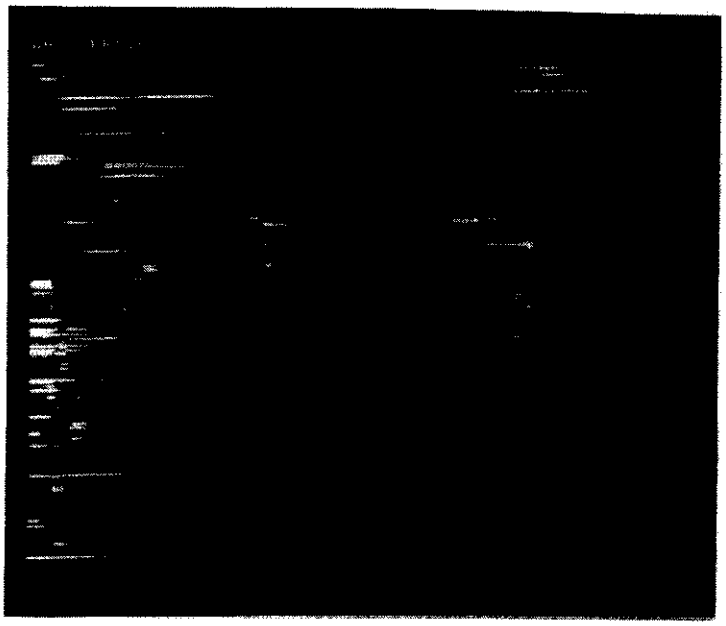


**Figure 5-18:** Difference of figure 5-17 (a) and (b).

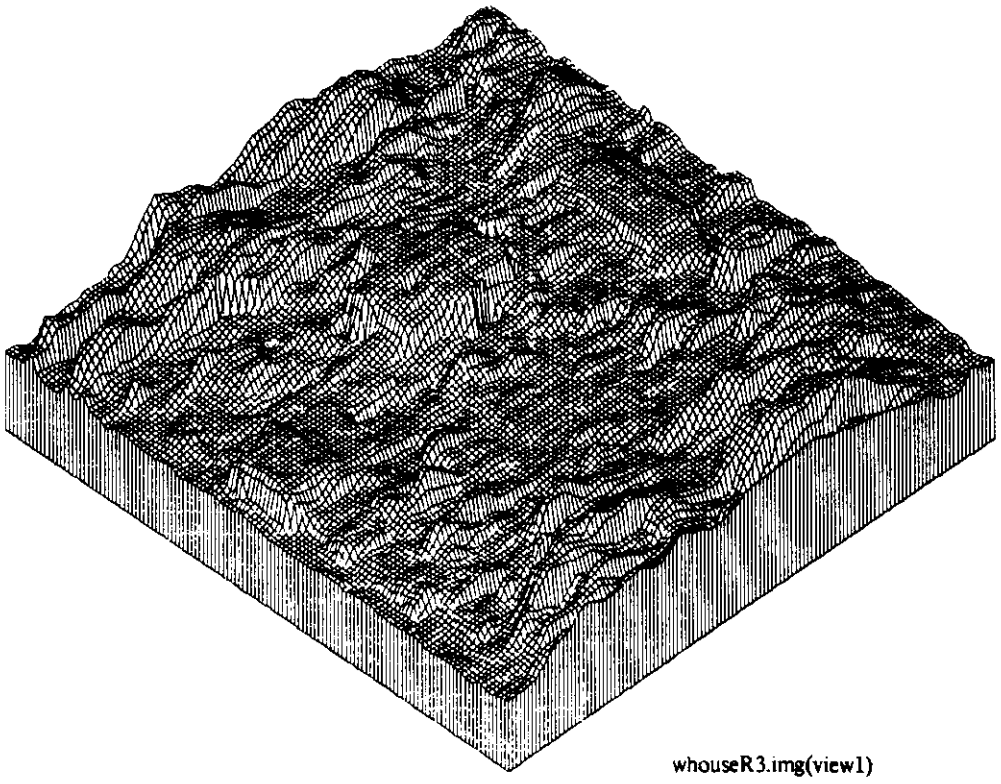
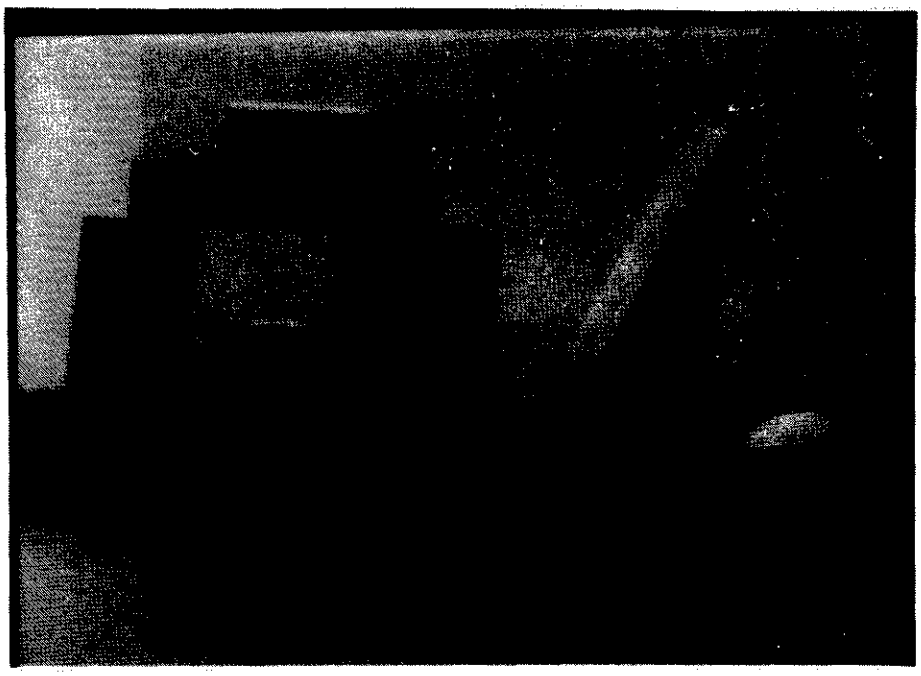
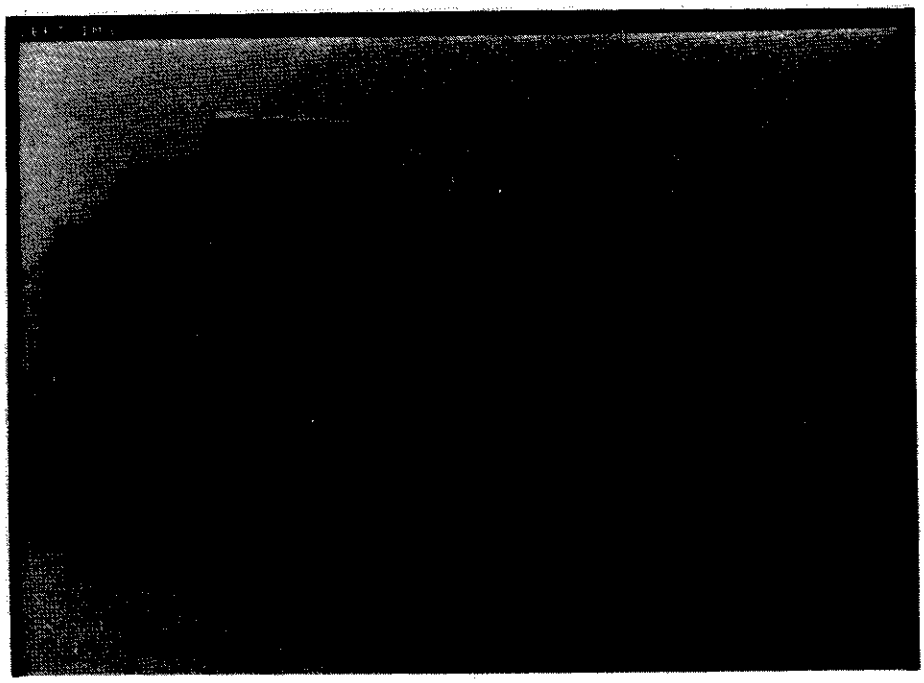



**Figure 5-19:** An isometric plot of disparity map of "white house".
*View from lower left corner.*



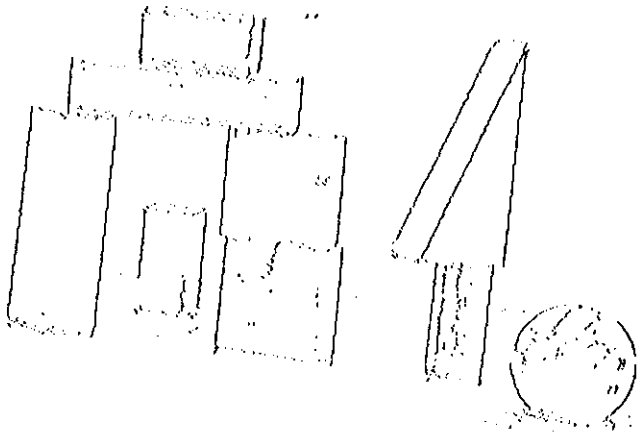
right image



left image

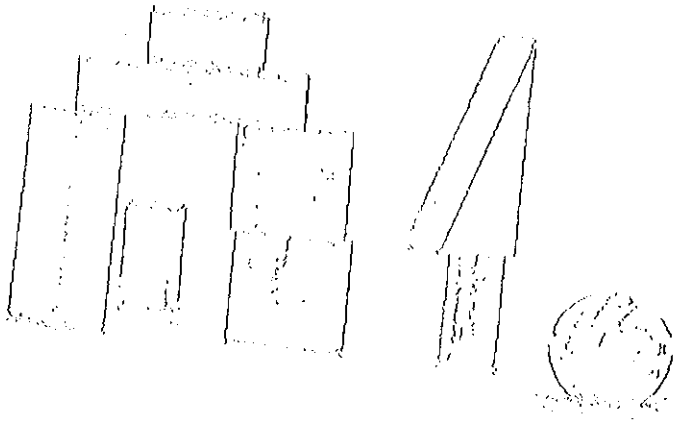
**Figure 5-20:** The "arch" stereo pair of block scene.

/visb/yxo/usc/arch/right.imgE



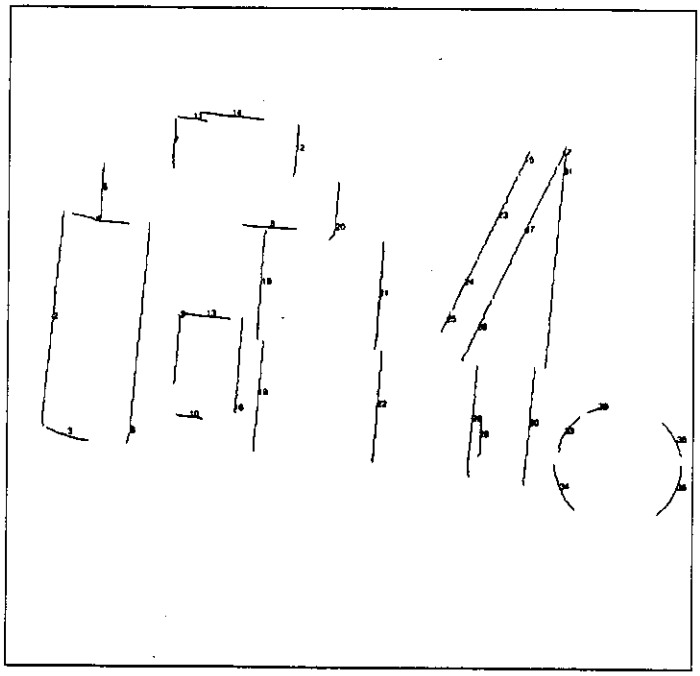
right image

/visb/yxo/usc/arch/left.imgE

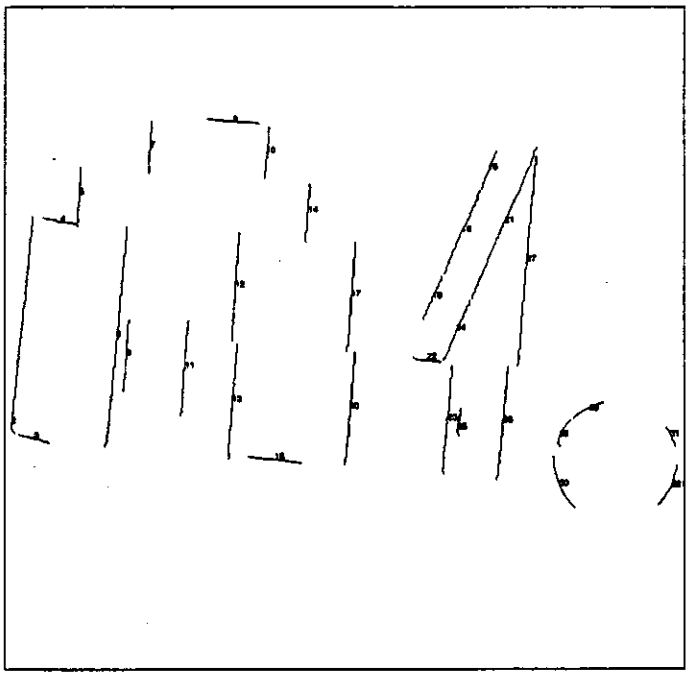


left image

**Figure 5-21:** Edges extracted on images in figure 5-20.

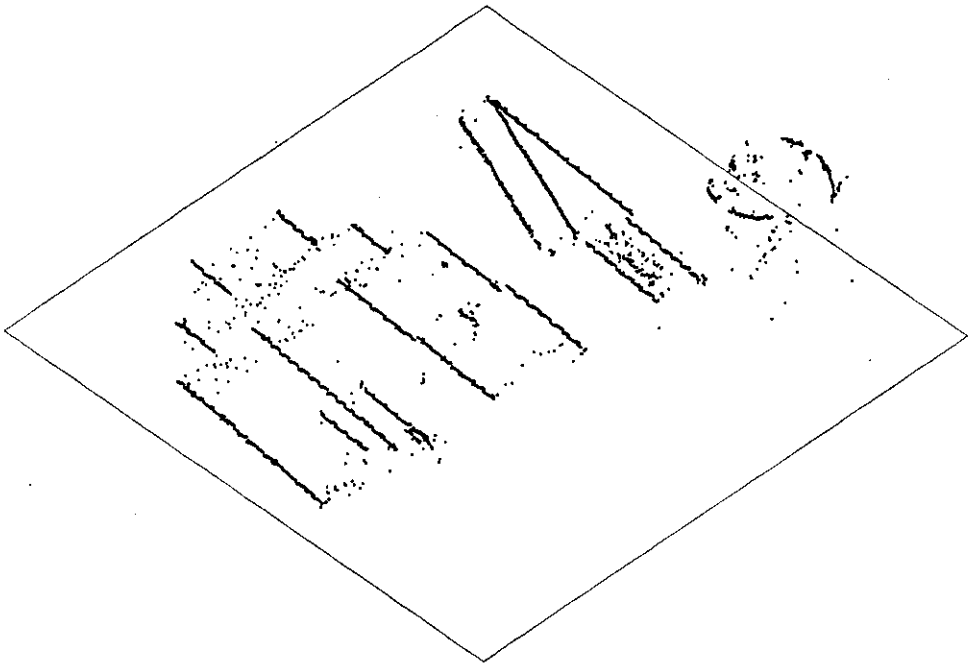


right image

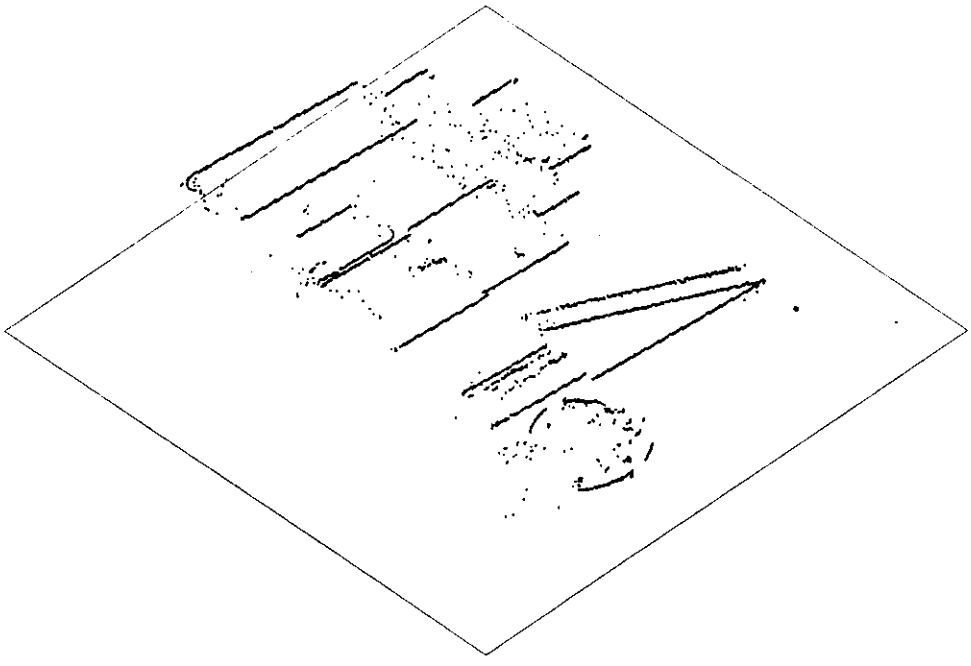


left image

**Figure 5-22:** Connected edges obtained from figure 5-21.

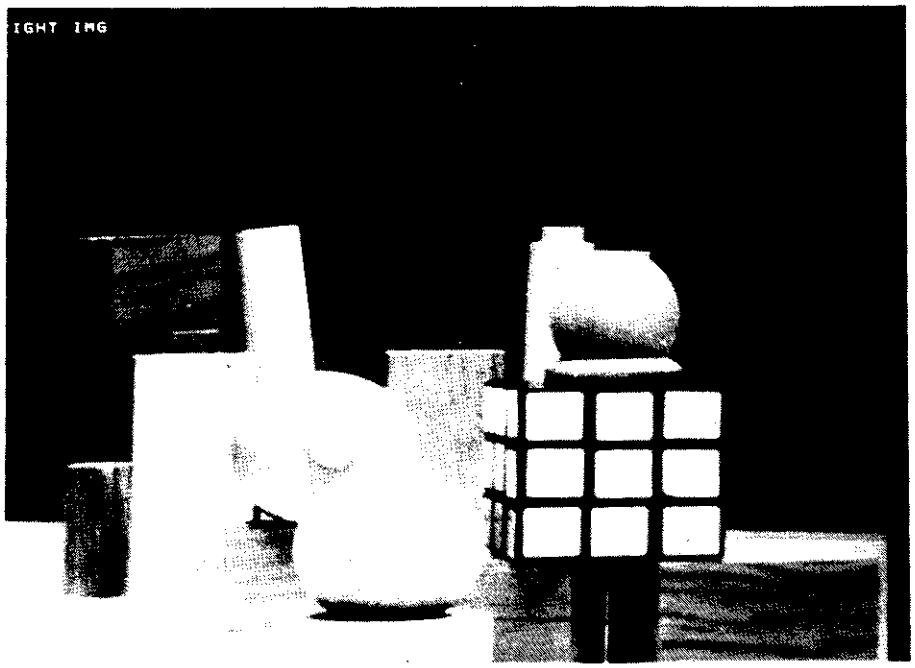


View from lower left corner

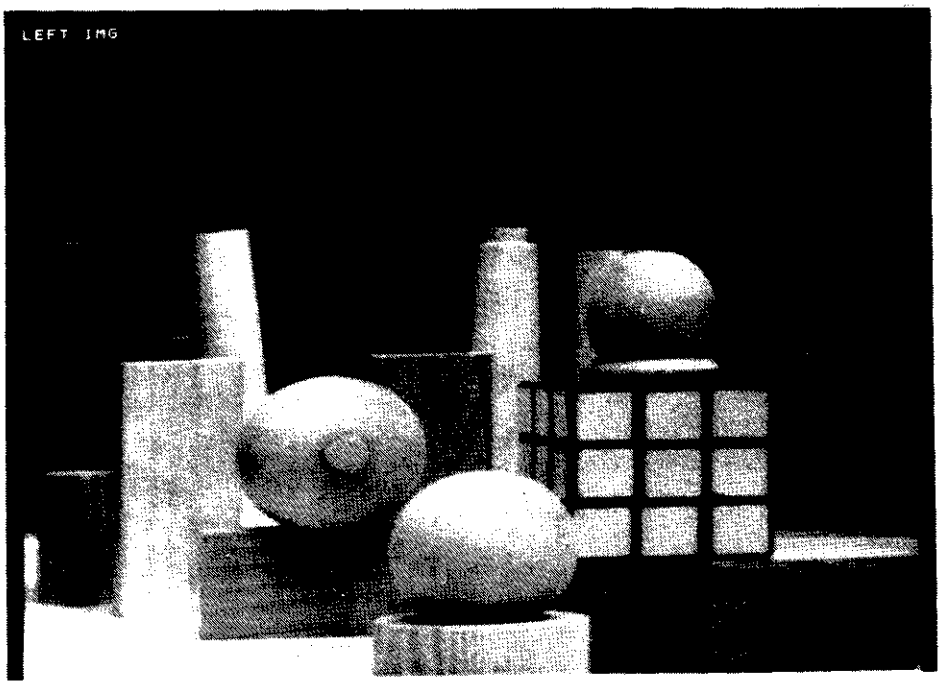


View from lower right corner

**Figure 5-23:** Perspective views of matched edges for figure 5-21.



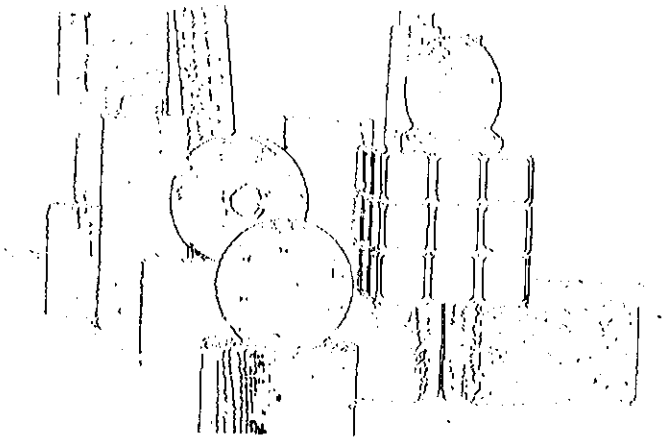

right image




left image

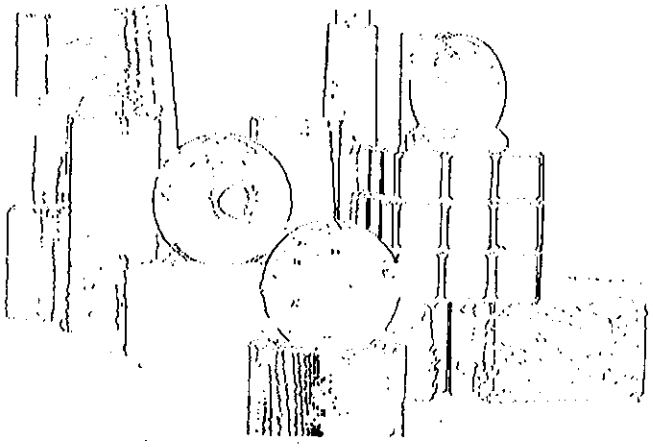
**Figure 5-24:** The "rubik" stereo pair of block scene.

/visb/yxo/uiuc/rubik/right.imgE



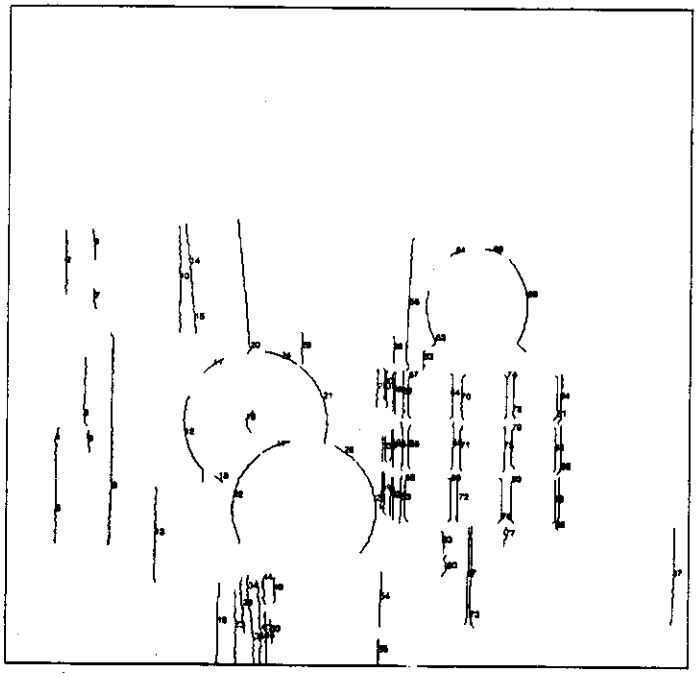
right image

/visb/yxo/uiuc/rubik/left.imgE

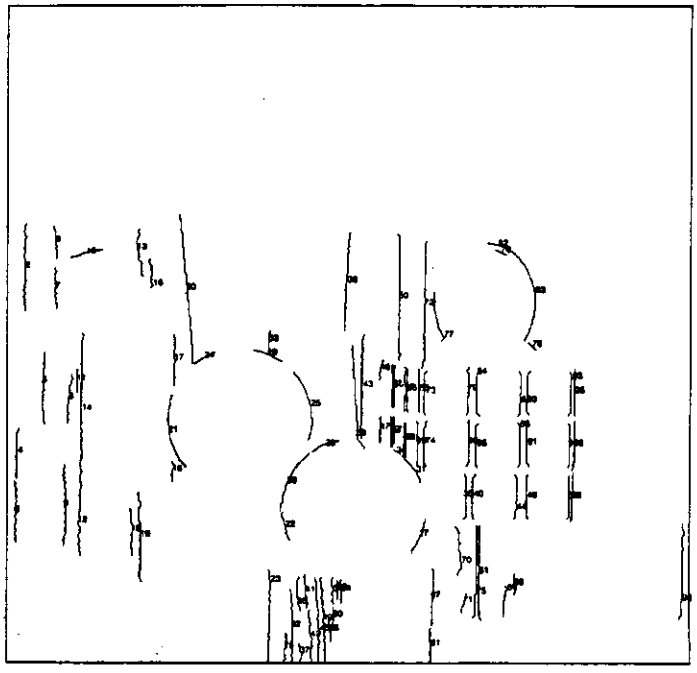


left image

**Figure 5-25:** Edges extracted on images in figure 5-24.

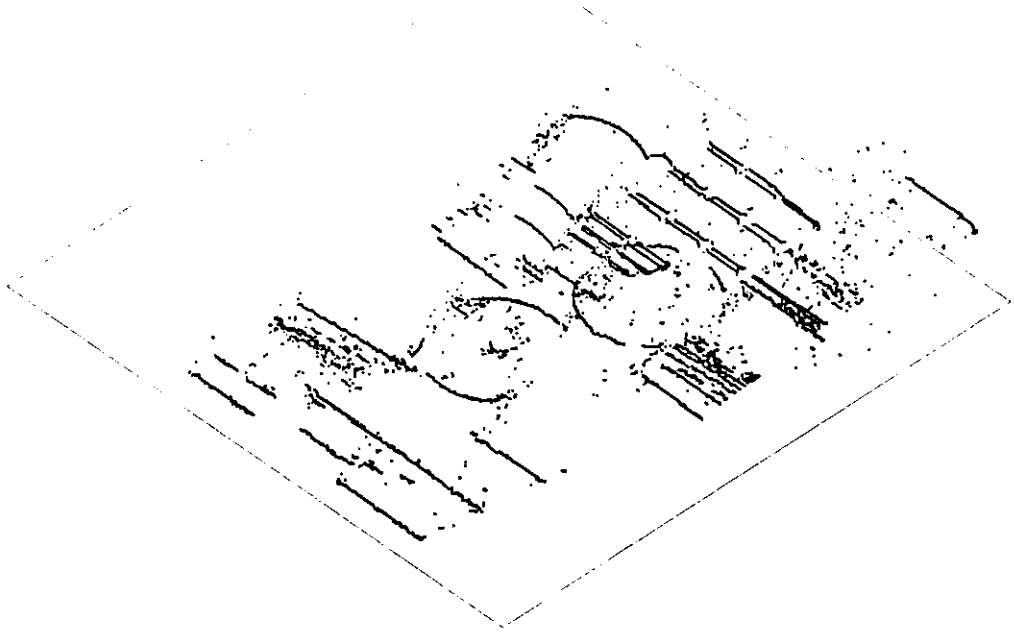


right image

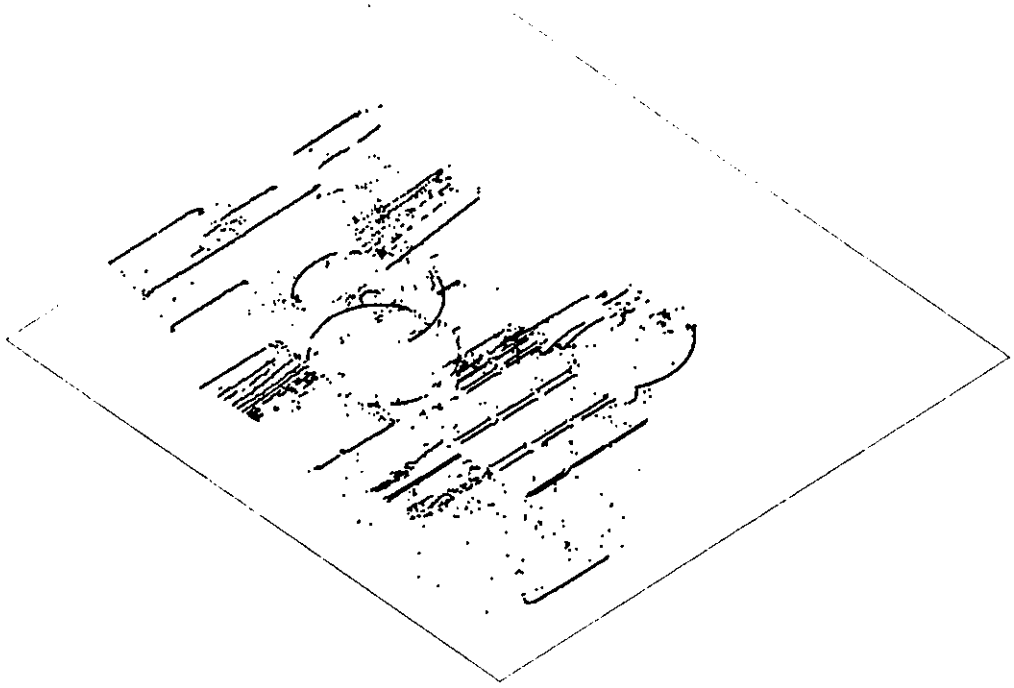


left image

**Figure 5-26:** Connected edges obtained from figure 5-25.



View from lower left corner



View from lower right corner

**Figure 5-27:** Perspective views of matched edges for figure 5-25.

**Table 5-1:** Summary of the stereo images used in the experiments.

| | cdc | pentagon | w-house | arch | rubik |
|---|---|---|---|---|---|
| image size (columns,rows) | 206, 256 | 512, 512 | 388, 388 | 512, 512 | 512, 512 |
| number of edges (right,left) | 7809, 7678 | 39719, 45809 | 21867, 25381 | 2657, 2611 | 5221, 5497 |
| number of connected edges (right,left) | 140, 143 | 398, 356 | 130, 155 | 36, 32 | 87, 96 |
| disparity range (pixels, % of width) | 35 (17%) | 20 (4%) | 35 (9%) | 51 (10%) | 97 (19%) |
| inconsistency (2D search, 3D search) | 86, 20 | 372, 27 | 436, 32 | 14, 2 | 269, 36 |
| CPU time (minutes) (2D search, 3D search) | 19, 321 | 52, 858 | 50, 739 | 2, 7 | 11, 87 |

## 6. Conclusion

In this paper, we have described a stereo algorithm which searches for an optimal solution in a 3D search space using dynamic programming. We have applied the algorithm to various domains including synthesized images, urban aerial images, and block scenes.

Performance of computational stereo should be evaluated by the results, the clarity of the algorithm, and the processing time. We obtained reasonably good results in each domain. Perhaps the major reason our algorithm accepts such a wide domain of images is as follows. For the image portions which contain long connected edges such as linear structures, our 3D search scheme works effectively to keep the consistency constraint. For the portions which do not contain long connected edges, our stereo algorithm provides ordinary 2D search which works effectively for the matching of isolated edges within each scanline pair. In other words, when inter-scanline constraints are available, our algorithm fully utilizes them, but when they are not available, our algorithm works as the 2D search. This feature will not be realized by an algorithm such as the segment-based one [Medioni and Nevatia 83] which depends heavily on the connectivity of edges. For some images containing a large number of edges, our algorithm requires a relatively long processing time. However, the processing time can be drastically reduced by implementing the algorithm in hardware. Hardware implementation is more realistic when the algorithm is simple. Actually, VLSI implementation of the dynamic programming algorithm is an existing technique in speech recognition [Ackland, Weste, and Burr 81]. We have started investigating an application of a systolic array processor made of Programmable Systolic Chips [Fisher et al. 83] for the hardware implementation of our stereo algorithm [Guerra and Kanade 83].

## Acknowledgements

Discussions we had with Jim Crowley, Marty Herman, David McKeown, and Steve Shafer during the course of this research were quite useful as well as their comments on the earlier drafts of this paper: we are grateful. We also thank Neil Swartz and Steven Thomas for their programming help, Cynthia Hibbard for editorial comments, and Kim Faught for drawing figures. Some of the stereo images ("cdc", "arch", and "rubik") were supplied by Gerald Medioni, University of Southern California.

## References

[Ackland, Weste, and Burr 81]
Ackland, B., Weste, N., and Burr, D.J.
An integrated multiprocessing array for time warp pattern matching.
In *Proc. 8th Annual Symposium on Computer Architecture*, pages pp.197-215. 1981.

[Baker 82] Baker, H. H.
*Depth from Edge and Intensity Based Stereo.*
Technical Report AIM-347, Stanford Artificial Intelligence Laboratory, 1982.

[Baker and Binford 81]
Baker, H.H. and Binford, T.O.
Depth from edge and intensity based stereo.
In *Proc. 7th International Joint Conference on Artificial Intelligence*, pages 631-636. Aug., 1981.

[Barnard and Fischler 82]
Barnard, S.T. and Fischler, M.A.
Computational Stereo.
*Computing Surveys* 14(4):553-572, Dec., 1982.

[Fisher et al. 83] Fisher, A.L., Kung, H.T., Monier, L.M. and Dohi, Y.
Architecture of the PSC: A Programmable Systolic Chip.
In *Proceedings of the Tenth International Symposium on Computer Architecture.* June, 1983.

[Gennery 79] Gennery, D.
Stereo-camera calibration.
In *Proceedings of Image Understanding Workshop*, pages 101-107. DARPA, Nov., 1979.

[Grimson and Marr 79]
Grimson, W.E.L. and Marr, D.
A computer implementation of a theory of human stereo vision.
In *Proceedings of Image Understanding Workshop*, pages 41-47. DARPA, Apr., 1979.

[Guerra and Kanade 83]
Guerra, C. and Kanade, T.
*Systolic Implementation of Stereo Algorithm.*
(in preparation), Carnegie-Mellon University, Computer Science Department, Aug., 1983.

[Henderson, et al. 79]
Henderson, R.L., Miller, W.J., and Grosch, C.B.
Automatic stereo reconstruction of man-made targets.
*SPIE* 186(6):240-248, 1979.

[Herman, Kanade, and Kuroe 83]
Herman, M., Kanade, T. and Kuroe, S.
The 3D MOSAIC Scene Understanding System.
In *Proc. 8th International Joint Conference on Artificial Intelligence*, pages pp.1108-1112. Aug., 1983.

[Lowerre 76] Lowerre, B. T.
*The HARPY Speech Recognition System.*
Tech. Rep., Computer Science Department, Carnegie-Mellon Univ., April, 1976.

[Medioni and Nevatia 83]
Medioni, G.G. and Nevatia, R.
Segment-based Stereo Matching.
In *Proceedings of Image Understanding Workshop*, pages 128-136. DARPA, June, 1983.

[Moravec 79] Moravec, H.
Visual mapping by a robot rover.
In *Proc. 6th International Joint Conference on Artificial Intelligence*, pages pp.598-600.
Aug., 1979.