Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming

YUICHI OHTA, MEMBER, IEEE, AND TAKEO KANADE, MEMBER, IEEE

Abstract—This paper presents a stereo matching algorithm using the dynamic programming technique. The stereo matching problem, that is, obtaining a correspondence between right and left images, can be cast as a search problem. When a pair of stereo images is rectified, pairs of corresponding points can be searched for within the same scanlines. 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. Vertically connected edges in the images provide consistency constraints across the 2D search planes. *Inter-scanline* search in a three-dimensional (3D) search space, which is a stack of the 2D search planes, is needed to utilize this constraint.

Our stereo matching algorithm uses edge-delimited intervals as elements to be matched, and employs the above mentioned two searches: one is inter-scanline search for possible correspondences of connected edges in right and left images and the other is intra-scanline search for correspondences of edge-delimited intervals on each scanline pair. Dynamic programming is used for both searches which proceed simultaneously: the former supplies the consistency constraint to the latter while the latter supplies the matching score to the former. An interval-based similarity metric is used to compute the score.

The algorithm has been tested with different types of images including urban aerial images, synthesized images, and block scenes, and its computational requirement has been discussed.

Index Terms—Aerial photo analysis, corresponding problem, depth map, dynamic programming, stereo, three-dimensional computer vision.

I. Introduction

STEREO is a useful method of obtaining depth information. The key problem in stereo is a search problem which finds the correspondence points between the left and right images, so that, given the camera model (i.e., the relationship between the right and left cameras of the stereo pair), the depth can be computed by triangulation. In edge-based stereo techniques, edges in the images are used as the elements whose correspondences are to be found [2]-[4], [8]. Even though a general problem of finding correspondences between images involves the search within the whole image, the knowledge of the camera model simplifies this image-to-image correspondence into a set of scanline-to-scanline correspondence problems. That is, once a pair of stereo images is rectified so that the epipolar lines are horizontal scanlines, a pair of corresponding edges in the right and left images should be searched for only within the same horizontal scanlines. 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 vertical and horizontal axes are the right and left scanlines. A dynamic programming technique can handle this search efficiently [2], [3].

However, if there is an edge extending across scanlines, the correspondences in one scanline have strong dependency on the correspondences in the neighboring scanlines, because if two points are on a vertically connected edge in the left image, their corresponding points should, most likely, lie on a vertically connected edge in the right image. The intra-scanline search alone does not take into account this mutual dependency between scanlines. Therefore, another search is necessary which tries to find the consistency among the scanlines, which we call *inter-scanline* search.

By considering both intra- and inter-scanline searches, the correspondence problem in stereo can be cast as that of finding in a three-dimensional (3D) search space an optimal matching surface that most satisfies the intra-scanline matches and inter-scanline consistency. Here, a matching surface is defined by stacking 2D matching paths, where the 2D matching paths are found in a 2D search plane whose axes are left-image column position and right-image column position, and the stacking is done in the direction of the row (scanline) number of the images. The cost of the matching surface is defined as the sum of the costs of the intra-scanline matches on the 2D search planes, while vertically connected edges provide the consistency constraint across the 2D search planes and thus penalize those intra-scanline matches which are not consistent across the scanlines. Our stereo matching uses dynamic programming for performing both the intra-scanline and the inter-scanline searches, and both searches proceed simultaneously. This method reduces the computation to a feasible amount.

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 algorithm.

II. Use of Inter-Scanline Constraints

As mentioned above, for a pair of rectified stereo images, matching edges within the same scanline (i.e., the intra-scanline search) should be sufficient in principle. However, in

Manuscript received December 15, 1983; revised October 29, 1984. This research was sponsored in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539, and in part by the Air Force Office of Scientific Research under Contract F49620-83-C-0100.

Y. Ohta is with the Institute of Information Sciences and Electronics, University of Tsukuba, Ibaraki, 305, Japan.

T. Kanade is with the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213.

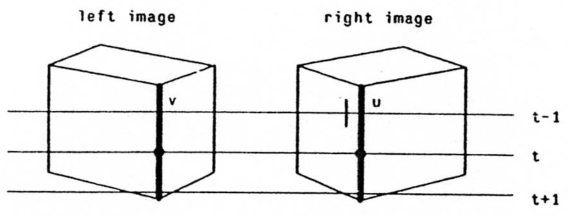


Fig. 1. Constraint provided by vertically connected edges.

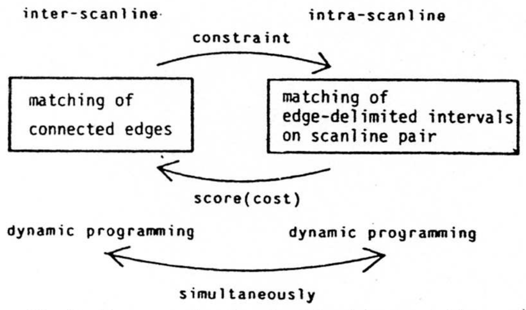

inter-scanline constraint
intra-scanline
matching of connected edges
matching of edge-delimited intervals on scanline pair
score(cost)
dynamic programming
dynamic programming
simultaneously

Fig. 2. Two searches involved in stereo matching.

practice, there is much ambiguity in finding correspondences solely by the intra-scanline search. To resolve the ambiguity, we can exploit the consistency constraints that vertically connected edges across the scanlines provide. Suppose a point on a connected edge *u* in the right image matches with a point on a connected edge *v* in the left image on scanline *t* as shown in Fig. 1. Then, other points on these edges should also match on other scanlines. If edges *u* and *v* do not match on scanline *t*, they should not match on other scanlines, either. We call this property inter-scanline consistency constraint. Thus, our problem is to search for a set of matching paths which gives the optimal correspondence of edges within scanlines under the inter-scanline consistency constraints. Our search space is a 3D space which is a stack of 2D search planes for intra-scanline matching, and two searches are involved as shown in Fig. 2. One is for the correspondence of all connected edges in right and left images, and the other is for the correspondence of edges (actually, intervals delimited by edges) on right and left scanlines under the constraint given by the former.

A few methods have been used to combine the inter-scanline search with the intra-scanline search. Henderson [9] sequentially processed each pair of scanlines and used the result of one scanline to guide the search in the next scanline. However, this method suffers in that the errors made in the earlier scanlines significantly affect the total results.

Baker [2] 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 constraints. Since this method, however, does not use the inter-scanline constraints directly in the search, 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 constraints in a 3D search space, but a feasible method was left as an open problem.

A straightforward way to achieve a matching which satisfies the inter-scanline constraints is to consider all matchings

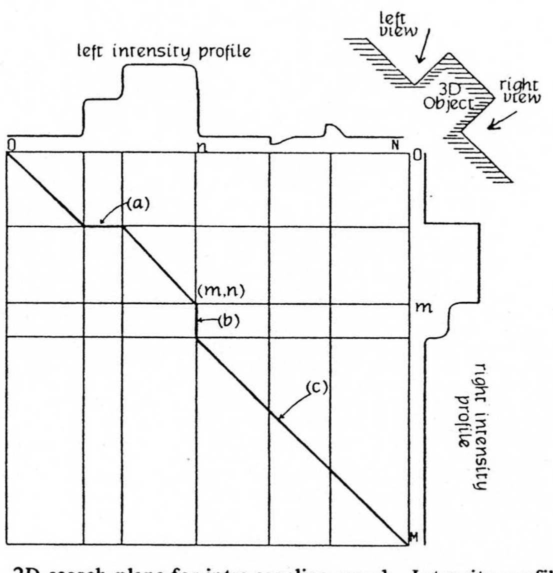


Fig. 3. 2D search plane for intra-scanline search. Intensity profiles are shown along each axis. The horizontal axis corresponds to the left scanline and the vertical one corresponds to the right scanline. Vertical and horizontal lines are the edge positions, and path selection is done at their intersections.

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

We propose to use dynamic programming, which is used for the intra-scanline search, also for the inter-scanline search. Dynamic programming [1] solves an N-stage decision process as N single-stage processes. This reduces the computational complexity to the logarithm of the original combinatorial one. In order to apply dynamic programming, however, the original decision process must satisfy the following two requirements. First, the decision stages must be ordered so that all the stages whose results are needed at a given stage have been processed before then. Second, the decision process should be *Markovian:* that is, at any stage the behavior of the process depends solely on the current state and does not depend on the previous history. It is not obvious whether these properties exist in the problem of finding correspondences between connected edges in stereo images, but we clarify them in the following sections.

III. Correspondence Search Using Dynamic Programming

A. Intra-Scanline Search on 2D Plane

The problem of obtaining correspondences between edges on right and left epipolar scanlines can be solved as a path finding problem on a 2D plane. Fig. 3 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 indexes to 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 indexes first. Legal paths which must be considered are sequences of straight line segments from node $(0, 0)$ at the upper left corner to node (M, N) at the lower right corner on a 2D array $[0: M, 0: N]$. They must go from the upper left to the lower right corners monotonically due to the above-mentioned condition on ordering. This is equivalent to the nonreversal constraint in edge correspondence: that is, the order of matched edges has to be preserved in the right and left scanlines. This constraint excludes from analysis thin objects such as wires and poles which may result in positional reversals in the images. A path has a vertex at node $\boldsymbol{m} = (m, n)$ when right edge m and left edge n are matched.

The cost of a path is defined as follows. 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 $(0, 0)$. The cost of a path is the sum of those of its primitive paths. A primitive path is a partial path which contains no vertices and is represented by a straight line segment as shown in Fig. 3. It should be noted that a primitive path actually corresponds to matching of the intervals delimited by edges at the start and end nodes rather than a matching of the edges themselves. Let $d(\boldsymbol{m}, \boldsymbol{k})$ be the cost of the primitive path from node $\boldsymbol{k}$ to node $\boldsymbol{m}$. (Our actual definition of $d(\boldsymbol{m}, \boldsymbol{k})$ will be given in Section IV-B.) Obviously, $d(\boldsymbol{m}, \boldsymbol{k}) \geqslant D(\boldsymbol{m}, \boldsymbol{k})$ and on an optimal path $d(\boldsymbol{m}, \boldsymbol{k}) \equiv D(\boldsymbol{m}, \boldsymbol{k})$.

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

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

$$D(O) = 0 \qquad (1)$$

$$\text{where } \boldsymbol{m} = (m, n), \boldsymbol{i} = (i, j), 0 \leqslant i \leqslant m,$$
$$0 \leqslant j \leqslant n, i + j \neq 0, \boldsymbol{o} = (0, 0).$$

Vector $\boldsymbol{i} = (i, j)$ represents a primitive path coming to node $\boldsymbol{m}$. When $i = 0$, the primitive path is horizontal, as shown at (a) in Fig. 3. 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, as shown at (b). When $i > 1$ and/or $j > 1$, the primitive path skips or ignores $i - 1$ and/or $j - 1$ edges on the right and/or left scanlines as 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.

B. 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 (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 ratio of the maximum disparity to the width of the image is $d(<1)$, the number of nodes to be examined becomes $d \times M \times N$. Furthermore, we can limit the number of edges which can be skipped 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 7×10^3 for our most complicated ones.

As seen in Fig. 3, the elements that are actually matched in our algorithm are the intervals between edges rather than the edges themselves as in Baker's [2]. This difference is mainly reflected in the difference of the definition of the cost function $d(\ )$. 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 to the representation of 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 pair of half edges which face each other. When there are N edges, there are $N - 1$ 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.

C. Inter-Scanline Search in 3D Space

The problem of obtaining a correspondence between edges under the inter-scanline consistency constraints 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. Fig. 4 illustrates this 3D space. The side faces of this space correspond to the right and left images of a stereo pair. The cost of a set of paths is defined as the sum of the costs of the individual paths in the set. We want to obtain an optimal (i.e., the minimal cost) set of paths satisfying the inter-scanline constraints. 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 this set of 2D nodes as a single 3D node. The optimal path on the 2D plane is obtained by iterating the selection of an optimal path at each 2D node. Similarly, the optimal set of paths in the 3D space is obtained by iterating the selection of an optimal set of paths at each 3D node. Connected edges, 3D notes, and sets of paths between 3D nodes are illustrated in Fig. 4.

As described in Section II, 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 where 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 com-

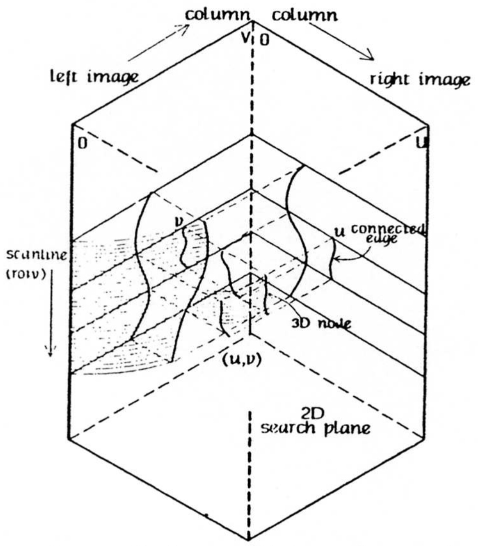


Fig. 4. 3D search space for intra- and inter-scanline search. This may be viewed as a rectangular solid seen from above. The side faces correspond to the right and left stereo images. Connected edges in each image form sets of intersections (nodes) in this space. Each set is called a 3D node. Selection of a set of paths is done at every 3D node.

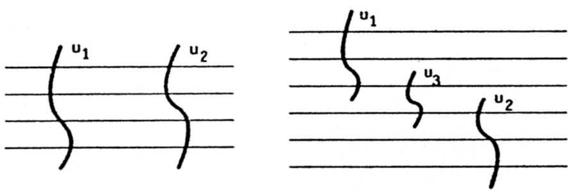


Fig. 5. Connected edge u_1 is on the left of u_2.

puted based on the cost obtained by the intra-scanline search on each 2D search plane. This leads to the following condition: *When we examine the correspondence of two 3D nodes (i.e., two connected edges), one in the right and one in the left image, the connected edges which are on the left of these two 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 are on the left of those in u_2 on the scanlines which u_1 and u_2 share. 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 (if u_1 and u_2 share any scanlines). Fig. 5 illustrates this concept. The order of two connected edges which do not satisfy both the relations in Fig. 5 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. Our edge-linking process which will be explained in Section IV is devised so that it does not produce such cases.

Now we will present how the cost of a 3D path is defined. Suppose we assign indexes [0: U] to connected edges 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. 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 reach to the 3D node $\boldsymbol{u}$. The cost $C(\boldsymbol{u})$ is computed as follows:

$$C(\boldsymbol{u}) = \min_{\{\boldsymbol{i}\}} \sum_{t=s(\boldsymbol{u})}^{e(\boldsymbol{u})} \{D(\boldsymbol{I}(\boldsymbol{u}; t), \boldsymbol{I}(\boldsymbol{u} - \boldsymbol{i}(t); t); t) + C(\boldsymbol{u} - \boldsymbol{i}(t); t)\}$$

$$C(\boldsymbol{O}) = 0, \text{ i.e., } C(\boldsymbol{O}; t) = 0 \quad \text{for all} \quad t$$

$$\text{where} \quad \boldsymbol{u} = (u, v), \boldsymbol{i}(t) = (i(t), j(t)),$$
$$0 \leqslant i(t) \leqslant u, 0 \leqslant j(t) \leqslant v,$$
$$i(t) + j(t) \neq 0. \tag{2}$$

Here, $C(\boldsymbol{u}; t)$ is the cost of the path on scanline t in the optimal set; that is, $C(\boldsymbol{u}) = \Sigma_{t=s(\boldsymbol{u})}^{e(\boldsymbol{u})} C(\boldsymbol{u}; t)$, and $D(\boldsymbol{m}, \boldsymbol{k}; t)$ 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 two given 3D nodes. The function $\boldsymbol{I}(\boldsymbol{u}; t)$ gives the index of a 2D node belonging to the 3D node $\boldsymbol{u}$ 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 $\boldsymbol{i}(t)$. A 3D node $u - \boldsymbol{i}(t)$ gives the start node of the 3D primitive path on scanline t. The inter-scanline constraint is represented by $\boldsymbol{i}(t)$. For example, if $\boldsymbol{i}(t)$ is independent of $\boldsymbol{i}(t-1)$, there are no constraints between scanlines and the search represented by (2) becomes equivalent to a set of intra-scanline searches which are performed independently on each scanline. Intuitively, $\boldsymbol{i}(t)$ must be equal to $\boldsymbol{i}(t-1)$ 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 $\boldsymbol{i}(t)$'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 $C(\boldsymbol{U})$ at the 3D node $\boldsymbol{U} = (U, V)$ (which is the 3D node formed by the right ends of stereo images) is obtained as the optimal set.

It should be noted that when there are no connected edges except for the right and left sides of the images, the algorithm (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.

D. Computational Cost for Inter-Scanline Search

The number of 2D primitive paths which should be examined in the 3D search space to compute $C(\boldsymbol{U})$ by (2) can be estimated to be $O(TU^2V^2M^2N^2)$, where T stands for the number of scanlines in the images, U and V are for the numbers of connected edges in the right and left images, and M and N for the (average) numbers of edge points in one scanline in the right and left images, respectively. This estimate is obtained as follows: 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

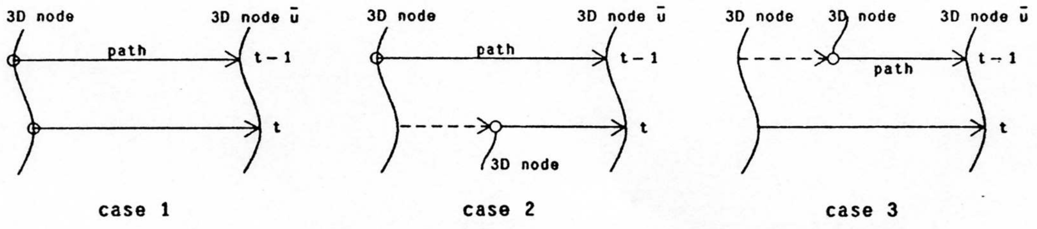


Fig. 6. Three cases for consistency constraint.

primitive path requires $O(M^2N^2)$ computation for intra-scanline search. Therefore, compared with the case of 2D search, the search in 3D space requires $U^2 \times V^2$ times more computation. Although this may seem 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. If we let the average length be $a \times T (a < 1)$, 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 there are 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 limit the allowable disparity range in the matching as we did in the intra-scanline search. This reduces the number of nodes to $d \times a^2 \times U \times V \times T$ where $d (<1)$ is the fraction of the maximum disparity to the width of the image. Third, in order to find the best $\boldsymbol{i}(t)$, we can use the beam search technique [10]; on the first scanline $s(\boldsymbol{n})$, we should examine every 3D primitive path and select W paths from the best. On many of the succeeding scanlines it is necessary to examine only W paths because $\boldsymbol{i}(t)$ is usually equal to $\mathrm{i}(t-1)$, and the average will come to W. The total number of 3D primitive paths examined in the 3D search is now $d \times a^2 \times U \times V \times T \times W$.

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 of the number of edges which can be skipped. 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 five. In the discussion above, we always took the lower bounds in our estimation, and therefore the actual value can be somewhat higher. However, the estimation suggests the search can be performed with a feasible amount of computation even by the 3D search algorithm.

E. 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 none of them are the vertices on the set of paths.* We need to represent this constraint as the relation between $\boldsymbol{i}(t)$ and $\boldsymbol{i}(t-1)$ in (2). To do this, let us consider the example in Fig. 6. Suppose we are trying to obtain a set of 3D primitive paths which reach to node $\boldsymbol{u}$. In order to satisfy the consistency constraints above, all the starting points of these paths should be the same 3D node; i.e., $\boldsymbol{i}(t) = \boldsymbol{i}(t-1)$ (case 1). The cases when the starting point is a different 3D node are shown as case 2 and case 3 in the figure. In case 2, 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 case 3, the 3D node $\boldsymbol{u} - \boldsymbol{i}(t-1)$ disappears on scanline t and the starting point is forced to move elsewhere.

Let us denote the 3D node $\boldsymbol{u} - \boldsymbol{i}(t)$, from which the 3D primitive path starts and reaches to the 3D node $\boldsymbol{u}$ on scanline t, by $frm\,(\boldsymbol{u}; t)$. Then the following rules should be satisfied in each case.

$$\begin{aligned} &\text{case 1:} \quad frm\,(\boldsymbol{u}; t) = frm\,(\boldsymbol{u}; t-1) \\ &\text{case 2:} \quad frm(frm(\boldsymbol{u}; t); t) = frm(\boldsymbol{u}; t-1) \\ &\text{case 3:} \quad frm(\boldsymbol{u}; t) = frm(frm(\boldsymbol{u}; t-1); t-1). \end{aligned} \tag{3}$$

The rules in case 2 and case 3 require that the decision at 3D node $\boldsymbol{u}$ depends on decisions at preceding 3D nodes. Unfortunately, a decision system with such a property is not *Markovian* as described in Section II, and therefore there is no guarantee of obtaining an optimal solution by using dynamic programming. This means that 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, we modify the rules in (3) as follows.

$$\begin{aligned} &\text{case 1;} \quad frm\,(\boldsymbol{u}; t) = frm\,(\boldsymbol{u}; t-1) \\ &\text{case 2:} \quad frm\,(\boldsymbol{u}; t) \geqslant frm\,(\boldsymbol{u}; t-1) \\ &\text{case 3:} \quad frm\,(\boldsymbol{u}; t) \leqslant frm\,(\boldsymbol{u}; t-1). \end{aligned} \tag{4}$$

The new rule for case 2 requires that the new 3D node on scanline t be on the right of the 3D node that is the starting point on scanline $t-1$. For a case 3, the new starting node on scanline t should be on the left of that on scanline $t-1$. It should be noted that though the new rules are always satisfied when the rules in (3) are satisfied, 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

$$\begin{array}{|c|c|}\hline -1 & 1 \\ \hline\end{array}$$

$$\begin{array}{|c|c|c|c|}\hline -\sqrt{2} & -\sqrt{2} & \sqrt{2} & \sqrt{2} \\ \hline\end{array}$$

$$\begin{array}{|c|c|c|c|c|c|c|c|}\hline -2 & -2 & -2 & -2 & 2 & 2 & 2 & 2 \\ \hline\end{array}$$

Fig. 7. Operators for edge detection. Results from operators of different sizes are combined.

represented by the rules in (4) are weaker than those of (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.

IV. Implementation

Implementation of the stereo algorithm which has been presented requires a method of detecting edges and linking them, and a definition of similarity measures for edge-delimited intervals. This section briefly describes the method and the definition which we actually used in our implementation and testing.

A. Detection, Linking, and Ordering of Edges

In stereo after rectification has been done, only edges running across the scanlines are useful for obtaining correspondence. We detect the positions of edges by differentiating an intensity profile along a scanline. The peaks and valleys in the first derivatives 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 Fig. 7, 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 larger operator. This prevents an edge from being detected more than once at slightly different positions by operators of different sizes.

The linking process links the edge positions into connected edges. An edge running nearly horizontally presents a difficulty, because it is detected as a set of positions which are apart on consecutive scanlines and linking them into a connected edge is not easy. Thus, we also detect the vertical edge positions by using operators rotated by 90 degrees from those in Fig. 7 and the linking process uses both horizontal and vertical edge positions to obtain connected edges. We adopt only the connected edges which are longer than a threshold, while shorter ones are kept as isolated edges. 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 each scanline. Fig. 8(a) illustrates this ordering. Second, a graph representing this local order is generated as shown in 8(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. Such 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

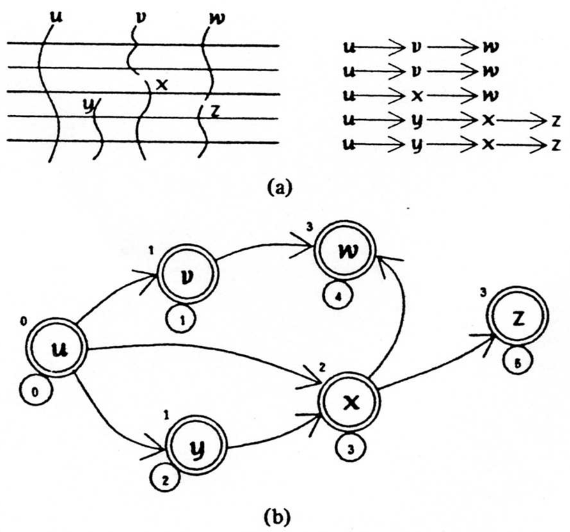


Fig. 8. Ordering of connected edges. (a) Connected edges and their local orders. (b) Global ordering.

connected edges cross each other, loops are formed in the graph and the maximum number goes to infinity. Our linking process was designed, therefore, not to make such connected edges. Finally, we order the connected edges by the ascending order of their maximum numbers of arcs. The ordering among the connected edges which have the same maximum number is arbitrary, and we have assigned smaller indexes to those which are found earlier when scanning the image from top to bottom. The ordering indexes assigned in this way are shown with circled numbers in Fig. 8(b).

B. 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 between intervals delimited by edges in the right and left images on the same scanline. If we let $a_1 \cdots a_k$ and $b_1 \cdots b_l$ 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

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

$$\sigma^2 = \frac{1}{2}\left(\frac{1}{k}\sum_{i=1}^{k} (a_i - m)^2 + \frac{1}{l}\sum_{j=1}^{l} (b_j - m)^2\right). \tag{5}$$

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

$$C_p = \sigma^2 \sqrt{k^2 + l^2}. \tag{6}$$

Intuitively, the meaning of this cost definition can be explained as follows. The pixels in two intervals which are matched to 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 a

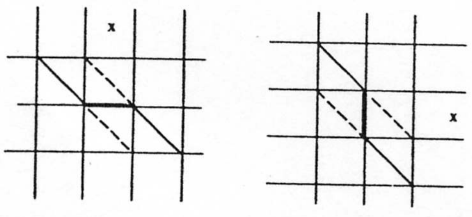


Fig. 9. Primitive paths for occlusion. Cost of a horizontal/vertical path is defined based on those of dotted paths.

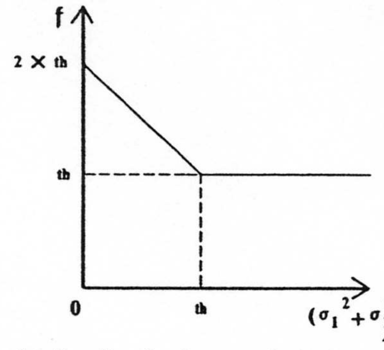


Fig. 10. Mapping function for the cost of a horizontal/vertical path.

cluster in a feature space formed by those intensity values, the variance may be called *within-class variance.* As described in Section III-A, some edges may be ignored in the matching process. Ignoring an edge means that two intervals separated by that edge are merged. Therefore, the intra-scanline search which uses (6) as the cost of a primitive path tries to find simultaneously the division of a scanline into segments and the matching between those segments to form an optimal set of clusters which minimizes the sum of class within-class variances.

Occlusion has not been considered in the definition of (5). As illustrated in Fig. 9, a horizontal or vertical path means that the interval marked with X is left 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 alternative, imaginary paths. When we denote the variances for those two paths by σ_1^2 and σ_2^2 [computed by the second formula of (5)], the cost of a vertical (horizontal) primitive path (i.e., the cost of occlusion) is defined as follows:

$$C_{Occ} = k \times f((\sigma_1^2 + \sigma_2^2)/2; th). \tag{7}$$

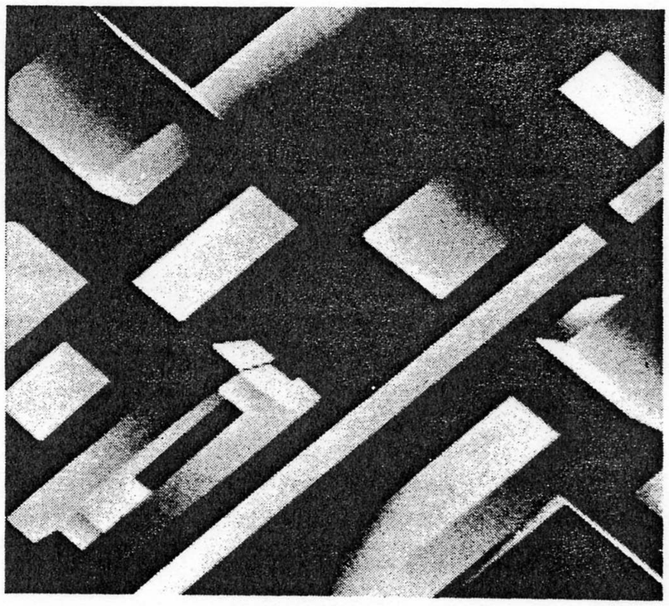
Here, k is the length of the primitive path, th is a threshold, and f is a function as shown in Fig. 10. When $(\sigma_1^2 + \sigma_2^2)/2$ is small, the function f gives a high cost, and when $(\sigma_1^2 + \sigma_2^2/2$ is large, the function f gives a low cost. The minimum value of f is determined by th.

V. Experimental Results

This section discusses the results obtained when we applied our stereo algorithm to images from various domains including synthesized images, urban aerial images, and block scenes.

A. Synthesized Images

We first applied our stereo algorithm to the synthesized stereo image pair shown in Fig. 11 which is from Control Data Corporation (CDC), and which has been used by Baker [2]. The image size is 256 × 206 pixels. We extracted every position where the intensity changes as edges, and Fig. 12 shows the edges thus extracted. Some edges in this image have very weak contrast; that is, the difference of intensity is only one gray level, and it is impossible to distinguish them from pseudo edges due to digitization. Our stereo algorithm can ignore the pseudo edges when they do not correspond to any edges 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. Fig. 13 displays the connected edges obtained from Fig. 12. The number attached to each connected edge indicates its ordering index.


(a)

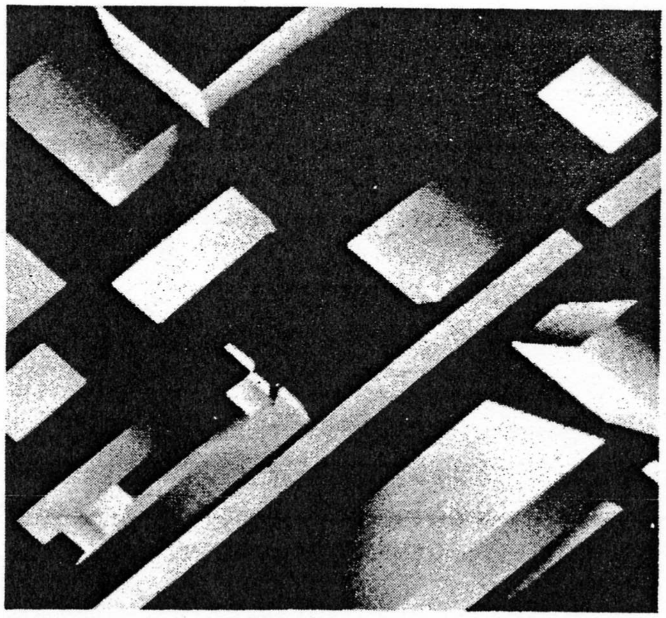

(b)

Fig. 11. The "CDC" stereo pair of synthesized images. (a) Right image. (b) Left image.

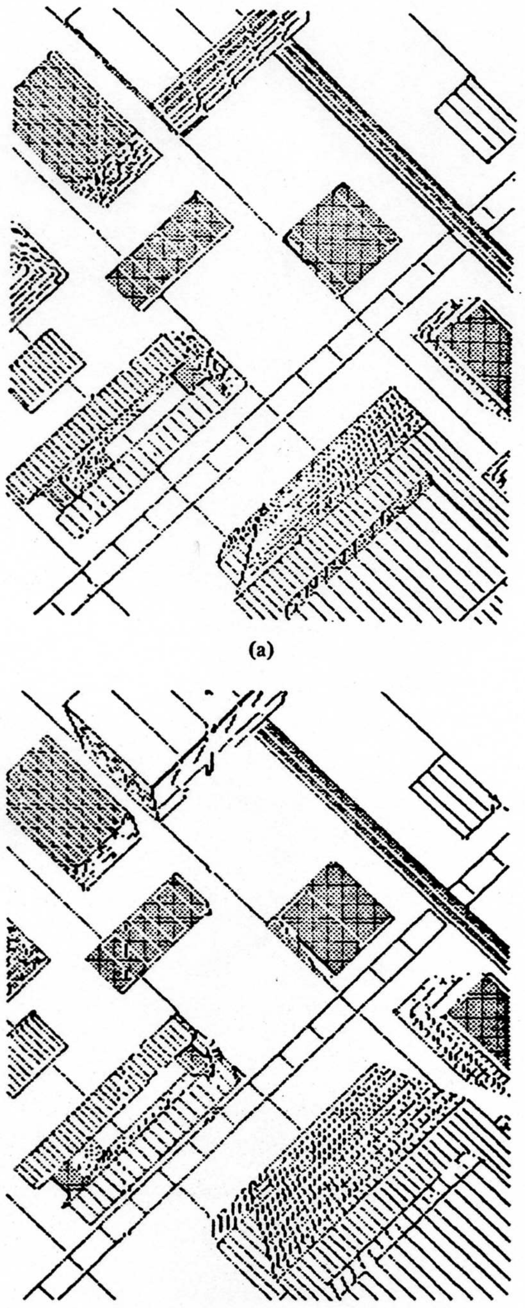


Fig. 12. Edges extracted from the images in Fig. 11. Only the edges extracted by horizontal operators are displayed. (a) Right image. (b) Left image.

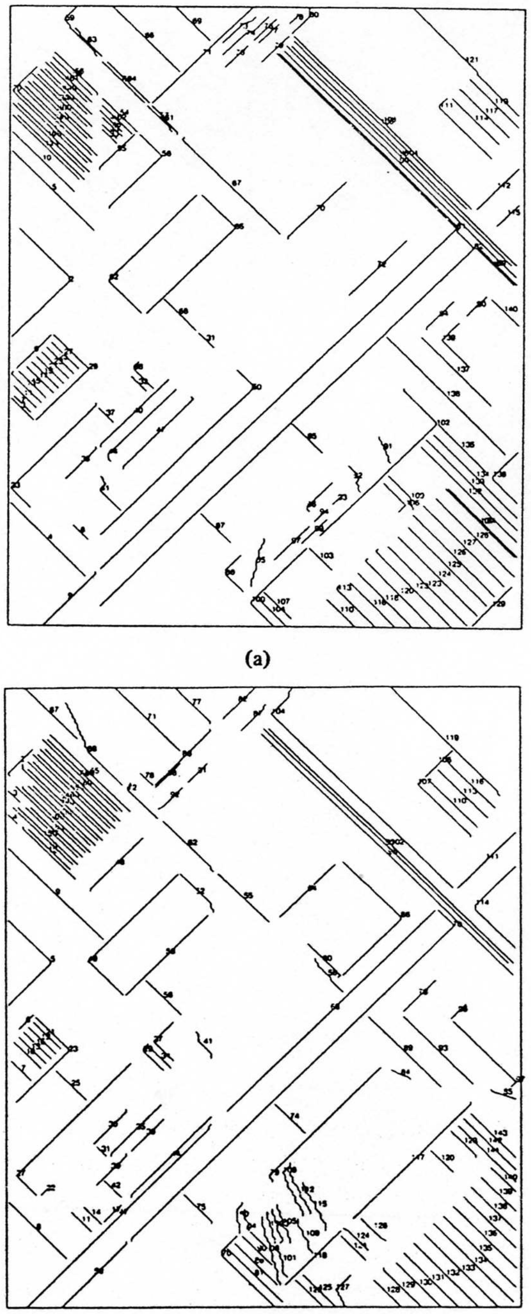


Fig. 13. Connected edges obtained from Fig. 12. The numbers attached are ordering indexes. (a) Right image. (b) Left image.

Fig. 14 illustrates a typical matching path obtained on a 2D search plane. The positions of connected edges are indicated by thicker lines and their indexes are attached. The thinner lines indicate isolated edges. The path shown by solid lines is the path obtained by the 3D search and the path shown by dotted lines is the one obtained without using the inter-scanline constraint (i.e., only by the 2D search). Note that using the inter-scanline constraint in the 3D search space results in a path which passes the 3D nodes (41, 42) and (50, 44).

Fig. 15 is a perspective view of edges which are matched in the 3D search space. Fig. 16 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 that is assigned. The black mat shows the regions where the disparity could not 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. 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 Fig. 3 or Fig. 14 illustrate the interpolation scheme. It should be noted that we did not

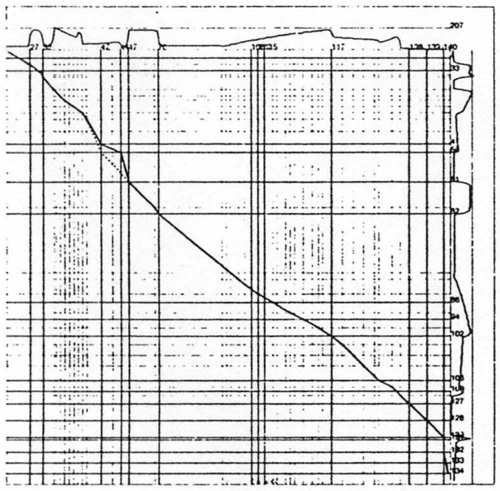


Fig. 14. A typical matching path on a 2D search plane. This is for scanline 207 of images in Fig. 11.

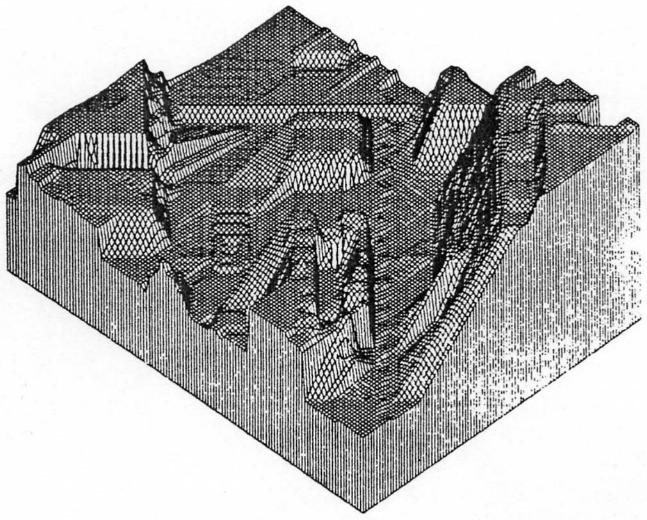


Fig. 17. An isometric plot of the disparity map in Fig. 16.

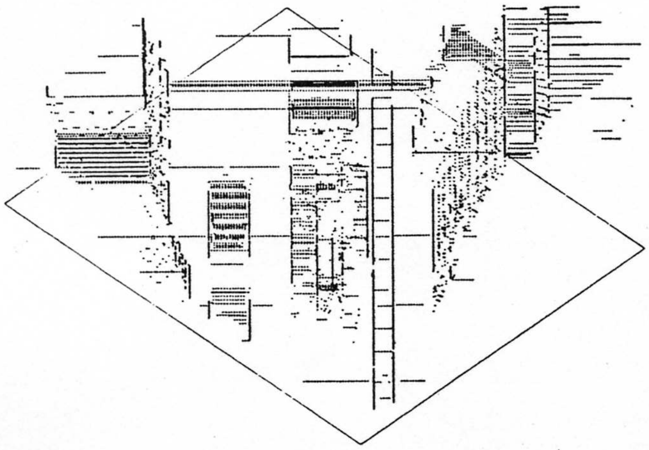
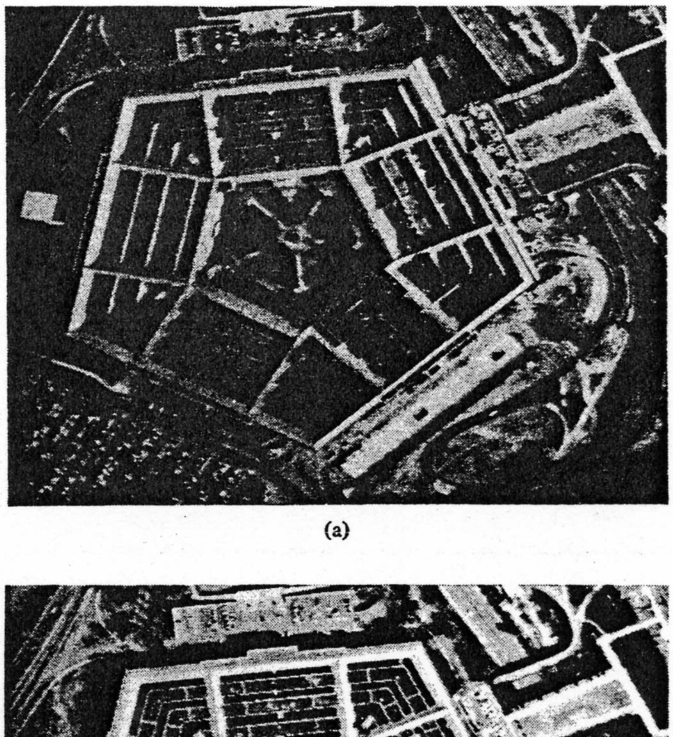


Fig. 15. A perspective view of matched edges obtained fom Fig. 12. Viewed from lower left corner.

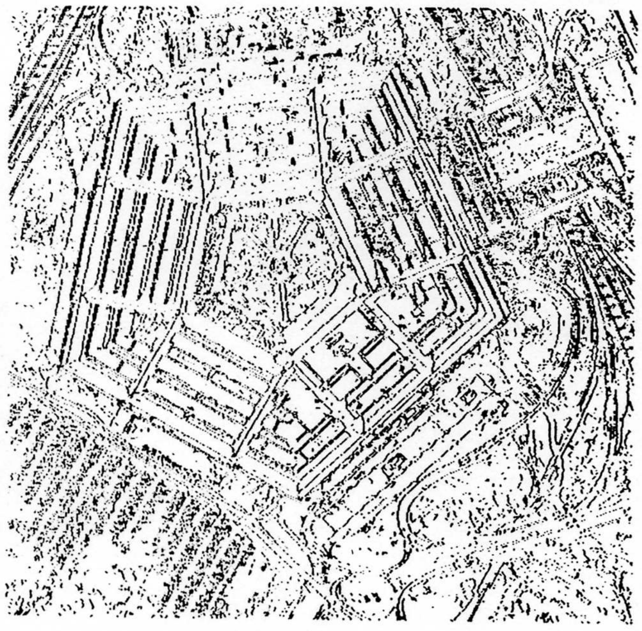
Fig. 16. Disparity map for the "CDC" stereo pair (Fig. 11). This map is registered in the right image coordinates. Higher elevation is displayed darker. The totally black mat areas indicate those for which disparity was not obtained.



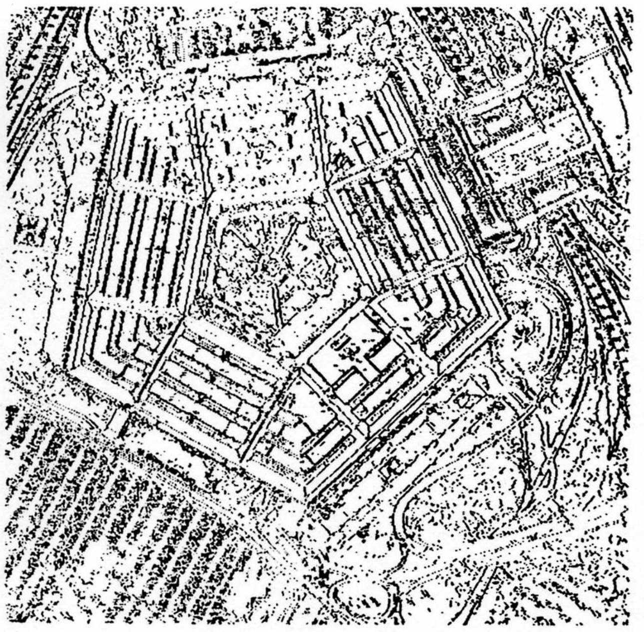
(a)

(b)

Fig. 18. The "Pentagon" stereo pair of urban aerial images. (a) Right image. (b) Left image.

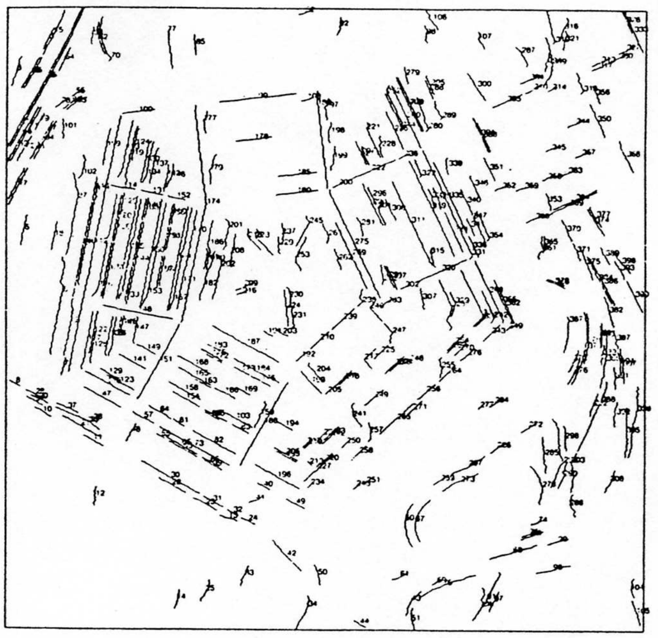


(a)

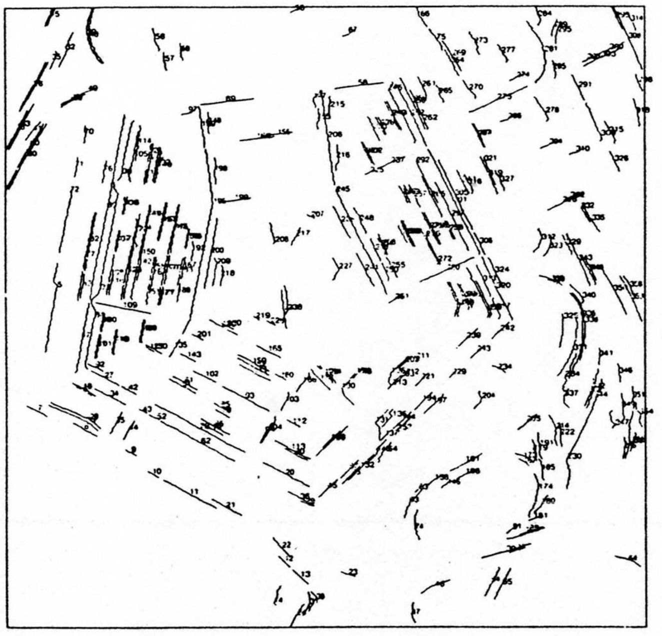


(b)

Fig. 19. Edges extracted from the images in Fig. 18. (a) Right image. (b) Left image.



(a)



(b)

Fig. 20. Connected edges obtained from Fig. 19. The numbers attached are ordering indexes. (a) Right image. (b) Left image.

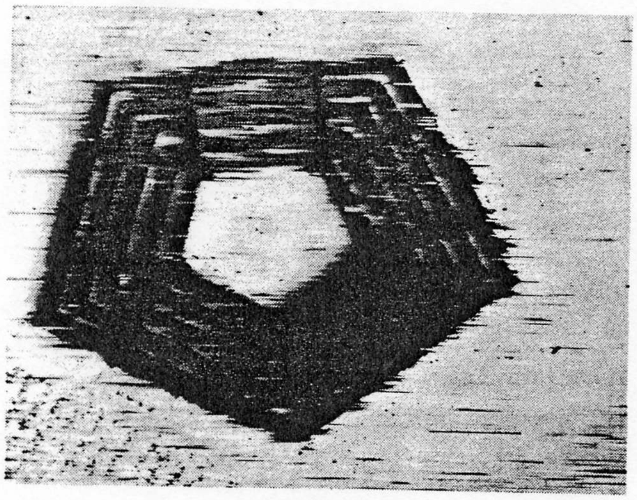
apply any smoothing operation to the disparity maps which are displayed in this paper. Fig. 17 is an isometric plot of the disparity map of Fig. 16.

B. Urban Aerial Images

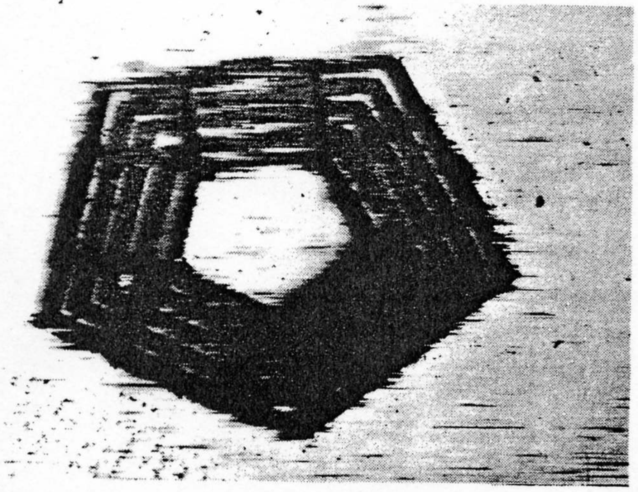
The stereo algorithm has also been applied to aerial photographs of the Washington, DC, area. The first stereo pair is "Pentagon" and the second one is "White House." They have been rectified using the camera models which were computed by Gennery's program [7] using manually selected point pairs.

Figs. 18–20 show the original stereo pair, edges, and connected edges for the "Pentagon" scene, 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. Fig. 21 (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. Fig. 22 displays an isometric plot of the disparity map.

Fig. 23 emphasizes the difference between the two disparity maps obtained by 2D search and 3D search. Many local mismatches on the roof of the building in Fig. 21 (a) are corrected in 21(b). We also counted the number of positions where the



(a)



(b)

Fig. 21. Disparity map obtained for the "Pentagon" stereo pair (Fig. 18). Both are registered in the left image coordinates. Notice that the detailed structures of the building roof and the bridge over the highway (upper right corner) have been recovered. (a) Result of 2D search. (b) Result of 3D search.

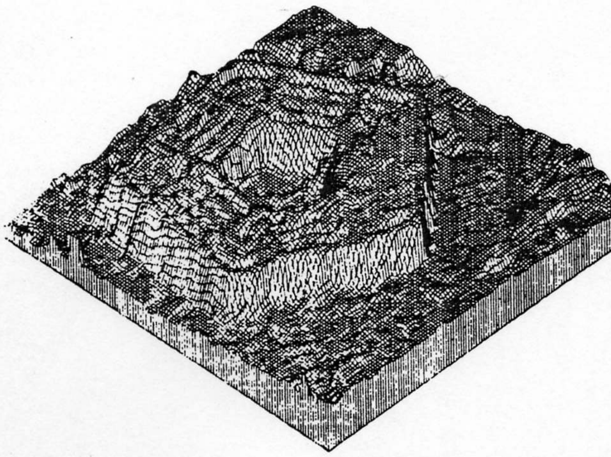


Fig. 22. An isometric plot of the disparity map of Fig. 21 viewed from lower left corner.

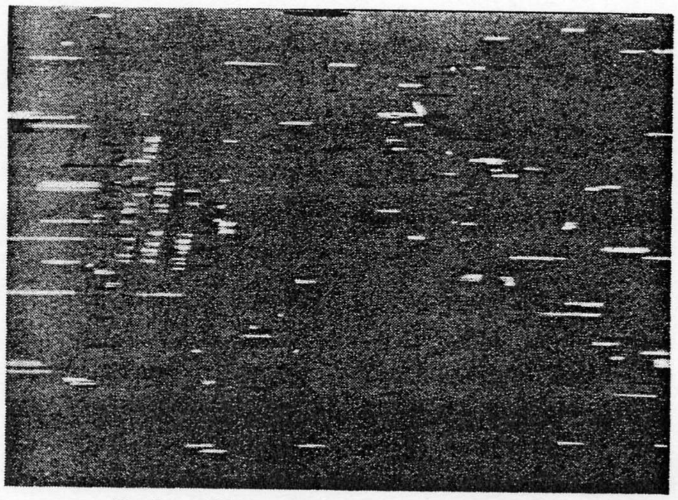
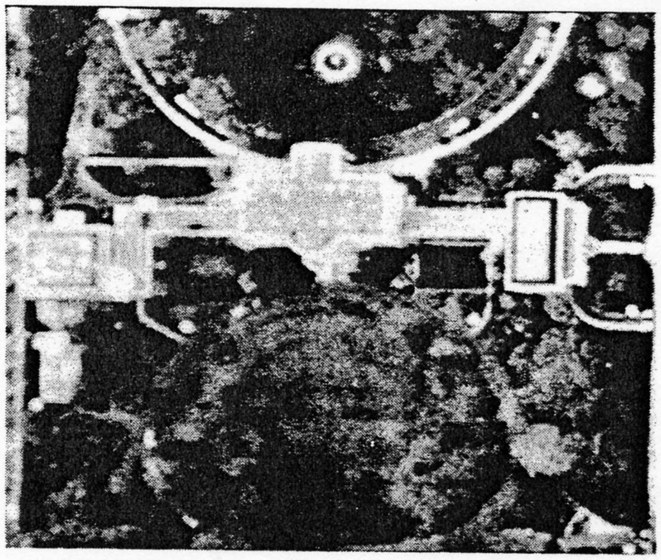
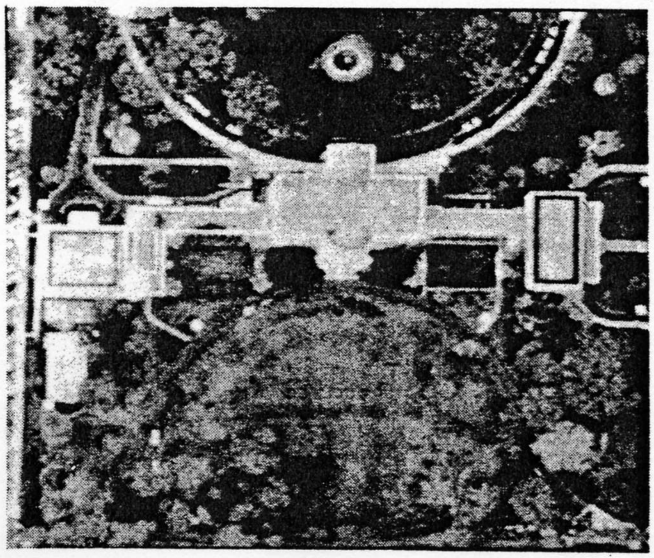


Fig. 23. Differences between Fig. 21(a) and (b).

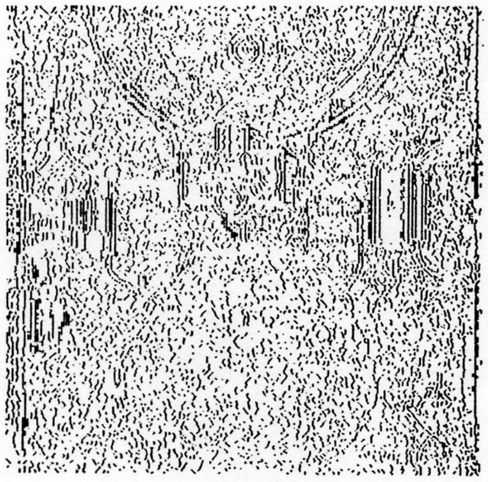


(a)



(b)

Fig. 24. The "White House" stereo pair of urban aerial images. (a) Right image. (b) Left image.


(a)

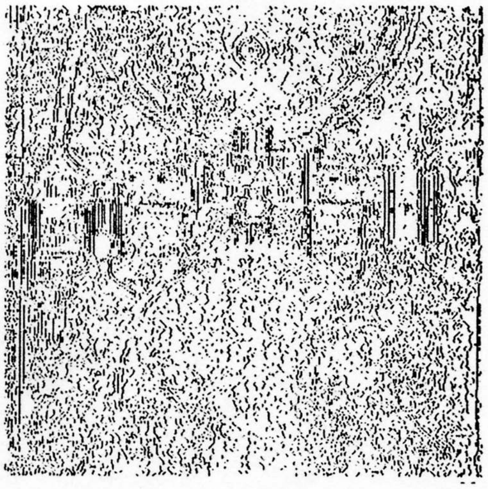
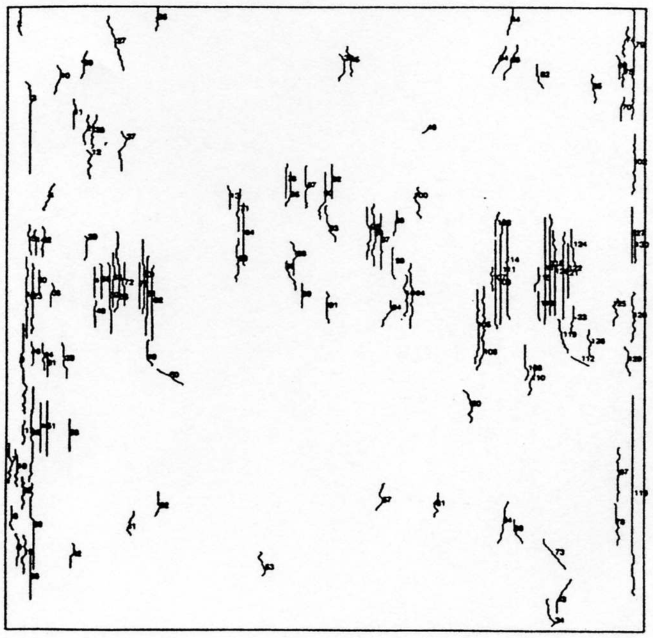

(b)

Fig. 25. Edges extracted from the images in Fig. 24. (a) Right image. (b) Left image.

consistency constraint, described in (3), 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 completely removed in the 3D case is that we used "weaker" rules (4) for the constraint as described earlier.

The image in Fig. 24 is the "White House" stereo pair. This example is an interesting and difficult one because it includes both buildings and highly textured trees. Fig. 25 and 26 show the edges and connected edges, respectively. 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 Fig. 27. Since the maps are registered in the right image coordinates, 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. Fig. 28 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.


(a)

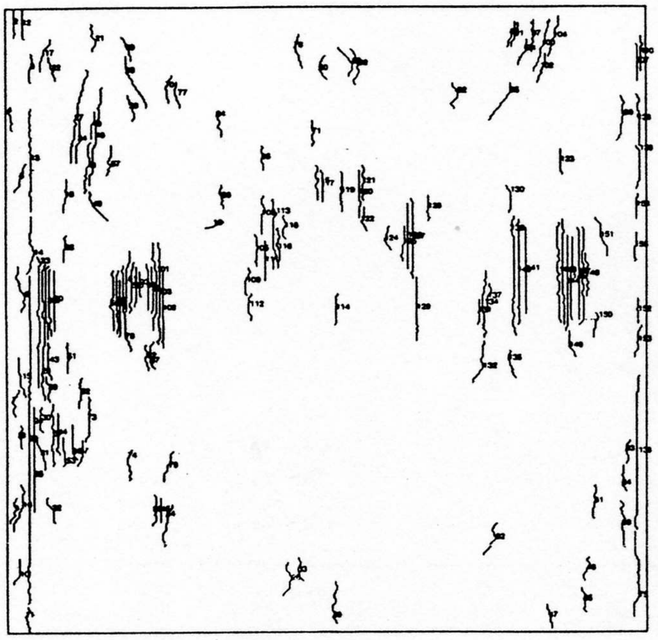

(b)

Fig. 26. Connected edges obtained from Fig. 25. (a) Right image. (b) Left image.

C. Block Scenes

We also applied our program to block scenes (obtained by courtesy of Dr. G. Medioni 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.

Fig. 29 shows the "arch" stereo pair. The image size is 512 X 512 pixels. Fig. 30 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.

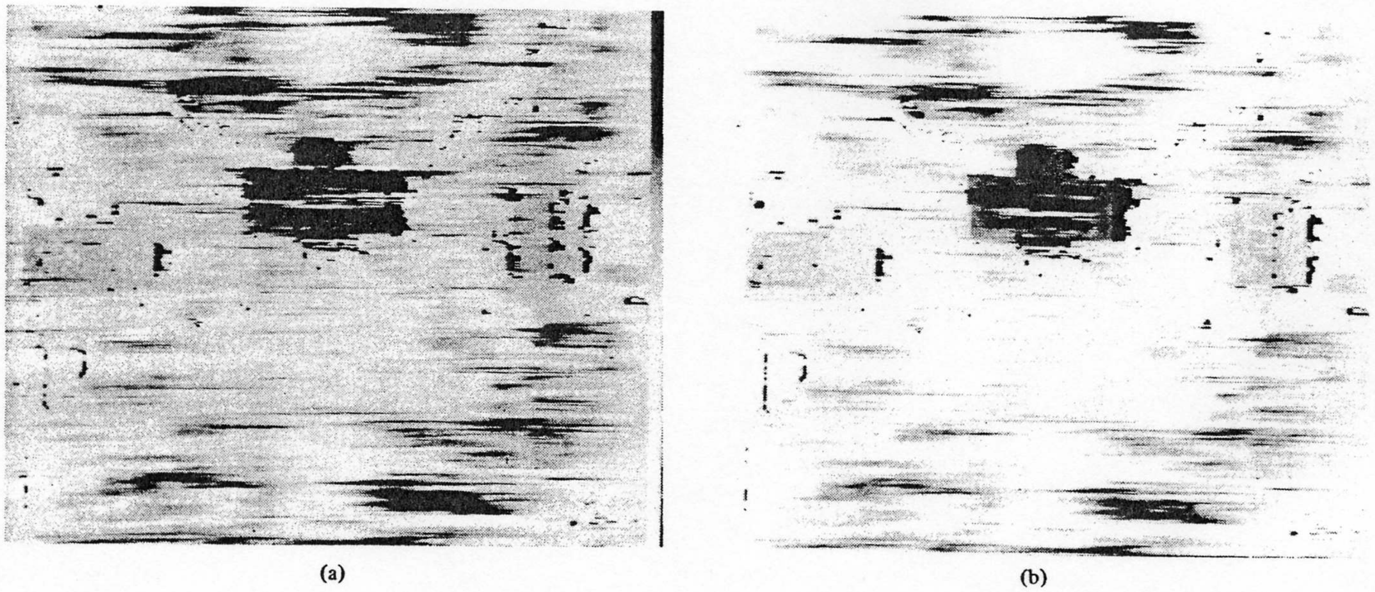

(a) (b)

Fig. 27. Disparity map obtained for the "White House" stereo pair (Fig. 24). Both are registered in the right image coordinates. (a) Result of 20 search. (b) Result of 3D search.

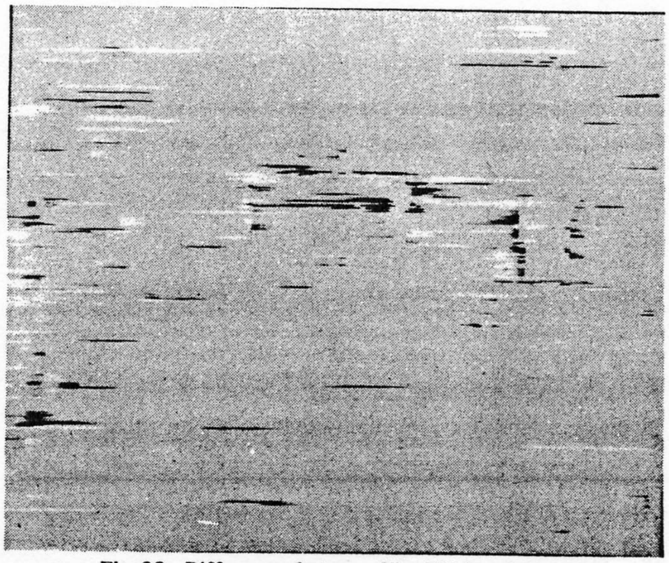


Fig. 28. Differences between Fig. 27(a) and (b).

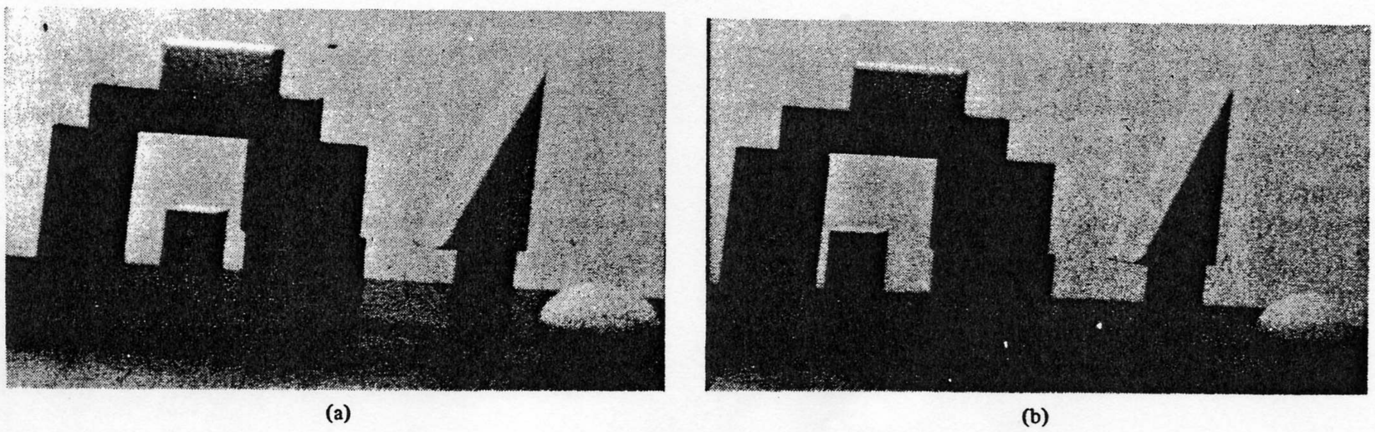
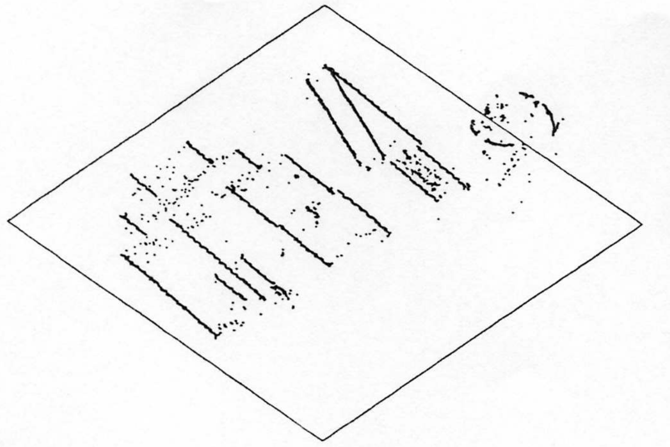

(a) (b)

Fig. 29. The "arch" stereo pair of a block scene. (a) Right image. (b) Left image.


(a)

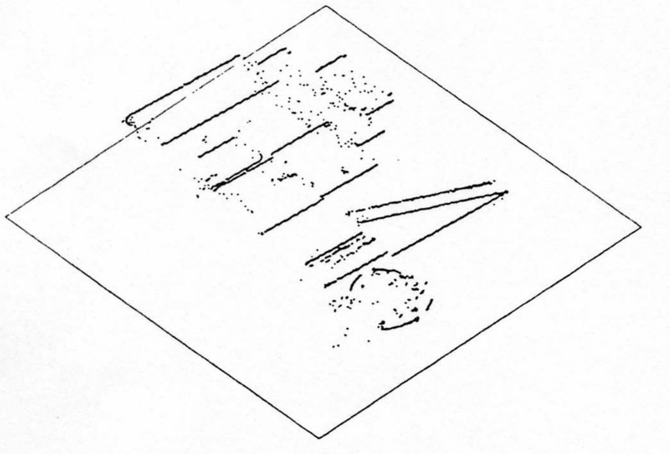
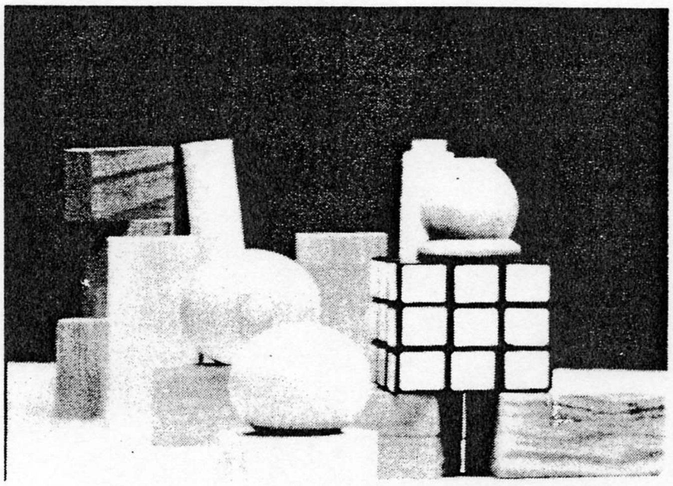

(b)

Fig. 30. Perspective views of matched edges for the "arch" stereo pair. (a) View from lower left corner. (b) View from lower right corner.


(a)

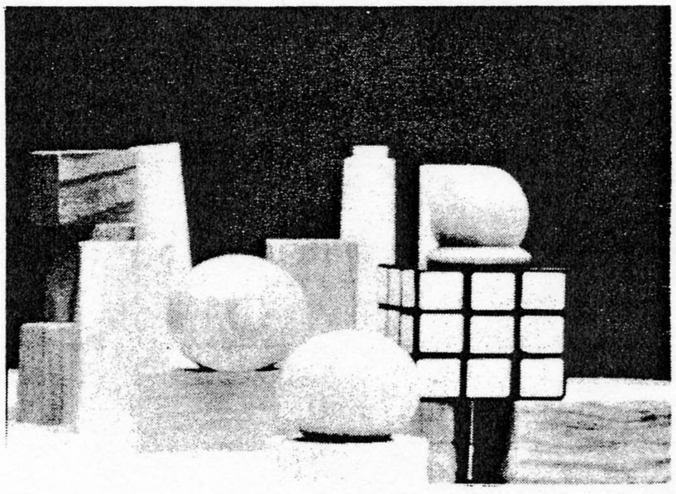

(b)

Fig. 31. The "Rubik" stereo pair of a block scene. (a) Right image. (b) Left image.

Fig. 31 shows another block scene "Rubik," whose image size is also 512 × 512 pixels, and Fig. 32 shows the extracted edges. The disparity range for this image is about 20 percent of the image width. The numbers of edges and connected edges are approximately 5 000 and 90, respectively, in each image. Though there are fewer edges than in the aerial images, 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 on the object Rubik Cube where repetitive patterns cause many ambiguities. In the 3D search, however, the inconsistencies are reduced to 36. Fig. 33 displays the perspective view of the matched edges.

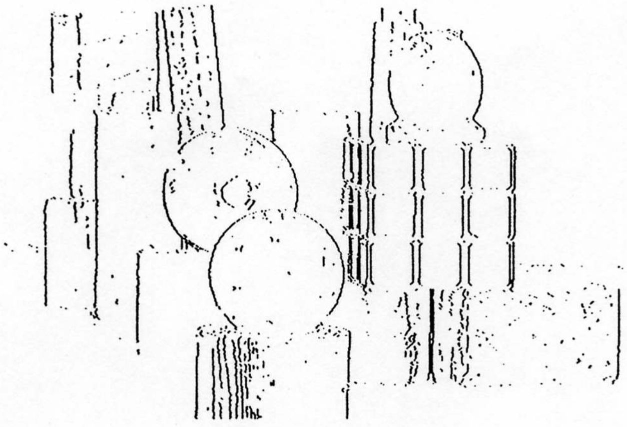
D. Summary of the Experiments

Table I summarizes the experiment of the stereo algorithm. 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.

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

VII. 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. Perhaps one of the major reasons that our algorithm works well for such a wide domain of images is as follows. For images which contain long connected edges such as linear structures in urban scenes, our 3D search scheme works effectively to enforce the consistency constraint. When images do not contain long connected edges, our stereo algorithm reduces to the ordinary 2D search which works efficiently to match isolated edges within each scanline pair. In other words, when inter-scanline constraints are available, our algorithm fully utilizes them, otherwise it works as the 2D


(a)

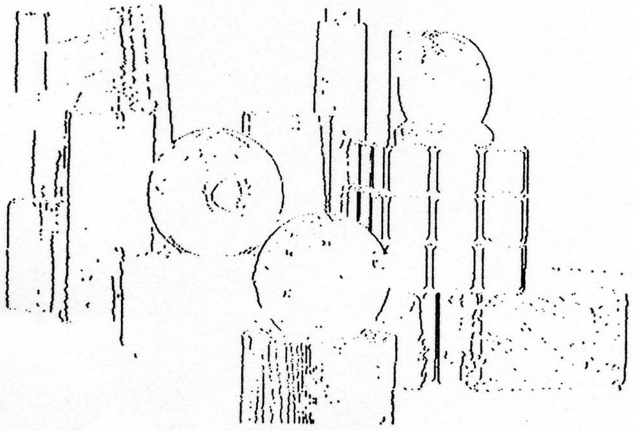
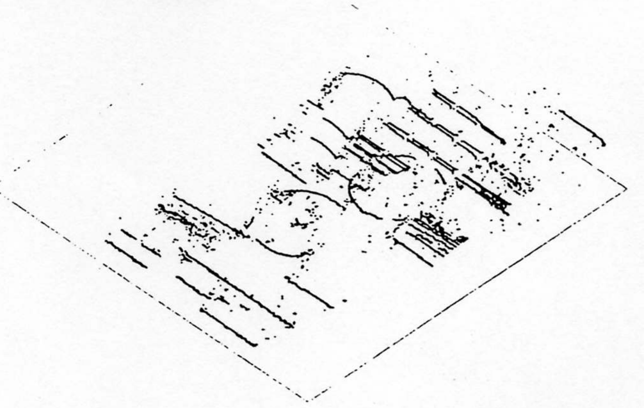

(b)

Fig. 32. Edges extracted from the images in Fig. 31. (a) Right image. (b) Left image.


(a)

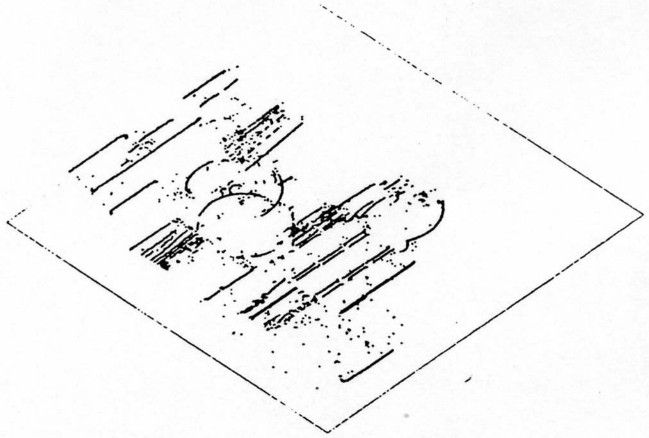

(b)

Fig. 33. Perspective views of matched edges for Fig. 32. (a) View from lower left corner. (b) View from lower right corner.

TABLE I
SUMMARY OF THE RESULTS OF THE EXPERIMENTS

	CDC	Pentagon	White 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 (mn) (2D search, 3D search)	19, 321	52, 858	50, 739	2, 7	11, 87

search. This feature will be less obvious in segment-based algorithms, such as in [11], which depend heavily on the connectivity of edges.

Table I shows that the 3D search achieves roughly $\frac{1}{10}$ error rate at the cost of computation which is ten times as much as the 2D search. For some images containing a large number of edges, it requires a much longer processing time than the method in which inter-scanline constraints are applied in a post-processing fashion [2]. However, we can argue that implementation of the algorithm on parallel hardware would reduce the processing time drastically, and then use of the 3D search for better results would be easily justified. If, for example, one can get a computational speed of 10^3 times faster, then even the most complex example in our experiments needs less than 1 mn to finish the 3D search. Implementation on parallel hardware is easier when algorithms are simple and uniform. That is the case for our 3D search algorithm which is a combination of two searches by dynamic programming. Actually, VLSI implementation of the dynamic programming algorithm is an existing technique in speech recognition [12]. Implementing our stereo algorithm on a systolic array processor [5] and on a semantic network array processor [6] is currently being investigated.

Acknowledgment

For discussions we had with J. Crowley, M. Herman, D. McKeown, and S. 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 N. Swartz and S. Thomas for their programming help, C. Hibbard for editorial comments, and K. Faught for drawing figures. Some of the stereo images ("CDC," "arch," and "Rubik") were supplied by G. Medioni, University of Southern California.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison-Wesley, 1974.

[2] H. H. Baker, "Depth from edge and intensity based stereo," Stanford Artif. Intell. Lab., Stanford Univ., Stanford, CA, Tech. Rep. AIM-347, 1982.

[3] H. H. Baker and T. O. Binford, "Depth from edge and intensity based stereo," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, Aug. 1981, pp. 631-636.

[4] S. T. Barnard and M. A. Fischler, "Computational Stereo," *Comput. Surveys*, vol. 14, pp. 553-572, Dec. 1982.

[5] P. M. Dew, "Implementation of the Ohta and Kanade dynamic programming stereo algorithm on a systolic array machine," unpublished internal rep., 1984.

[6] V. Dixit and D. I. Moldovan, "Semantic network array processor and its applications to image understanding," in *Proc. Image Understanding Workshop*, Oct. 1984, pp. 65-71.

[7] D. Gennery, "Stereo-camera calibration," in *Proc. Image Understanding Workshop*, DARPA, Nov. 1979, pp. 101-107.

[8] W. E. L. Grimson and D. Marr, "A computer implementation of a theory of human stereo vision," in *Proc. Image Understanding Workshop*, DARPA, Apr. 1979, pp. 41-47.

[9] R. L. Henderson, W. J. Miller, and C. B. Grosch, "Automatic stereo reconstruction of man-made targets," *SPIE*, vol. 186, pp. 240-248, 1979.

[10] B. T. Lowerre, "The HARPY speech recognition system," Comput Sci. Dep., Carnegie-Mellon Univ., Tech. Rep., Apr. 1976.

[11] G. G. Medioni and R. Nevatia, "Segment-based stereo matching," in *Proc. Image Understanding Workshop*, DARPA, June 1983, pp. 128-136.

[12] N. Weste, D. J. Burr, and B. D. Ackland, "Dynamic time warp pattern matching using an integrated multiprocessing array," *IEEE Trans. Comput.*, vol. C-32, pp. 731-744, Aug. 1983.

Yuichi Ohta (M'84) was born in Osaka, Japan, on November 26, 1949. He received the B.E. degree in electronical engineering and the M.Sc. and Ph.D. degrees in information science, all from Kyoto University, Kyoto, Japan, in 1972, 1974, and 1980, respectively.

From 1978 to 1981 he was a Research Associate, Faculty of Engineering, Kyoto University. Since 1981 he has been an Assistant Professor, Institute of Information Sciences and Electronics, University of Tsukuba, Ibaraki, Japan. He was a Visiting Scientist at the Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, from 1982 to 1983. His interests are in the fields of computer vision, picture processing, and artificial intelligence.

Dr. Ohta is a member of the Institute of Electronics and Communication Engineers of Japan and the Information Processing Society of Japan.

Takeo Kanade (M'80) was born in Hyogo, Japan, on October 24, 1945. He received the B.E., the M.Sc., and the Ph.D. degrees in electrical engineering from Kyoto University, Kyoto, Japan, in 1968, 1970, and 1974, respectively.

From 1973 to 1980, he was on the faculty at the Department of Information Science, Kyoto University. In 1980, he joined the faculty at Carnegie-Mellon, Pittsburgh, PA, where he is currently Associate Professor of the Computer Science Department and the Robotics Institute. His research projects include image understanding, robotics vision, development of the CMU Direct-Drive manipulator, and autonomous navigation.