STEREO BY TWO-LEVEL DYNAMIC PROGRAMMING

Yuichi Ohta
Institute of Information Sciences and Electronics
University of Tsukuba
IBARAKI, 305, JAPAN

Takeo Kanade
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

This paper presents a stereo algorithm using 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 in two levels: the former supplies the consistency constraints to the latter while the latter supplies the matching score to the former. An interval-based similarity metric is used to compute the score.

1. 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 (ie., the relationship between the right and left cameras of the stereo pair), the depth can be computed by triangulation. In edge-based techniques, edges in the images are used as the elements whose correspondences to be found [3,4,6]. 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 problem 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 [2,3,7].

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 constraints 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

This research was done while Y.Ohta was visiting the Department of Computer Science, Carnegie-Mellon University. The research was sponsored in part by the Defence 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.

searches, and both searches proceed simultaneously in two levels. This method reduces the computation to a feasible amount.

2. Use of Inter-Scanline Constraints

As mentioned above, for a pair of rectified stereo images, matching edges within the same scanline (ie., the intra-scanline search) should be sufficient in principle. However, in 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 a in the right image matches with a point on a connected edge *v* in the left image on scanline t. 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 constraint.

A few methods have been used to combine the inter-scanline search with the intra-scanline search. Henderson [7] 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 by 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 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. These two searches are combined as shown in figure 1. 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. The scheme to use dynamic programming in two levels was first employed in the recognition of connected spoken words [10]. They used one search for the possible segmentation at word boundaries and the other for the time-warping word matching under the constraint given by the

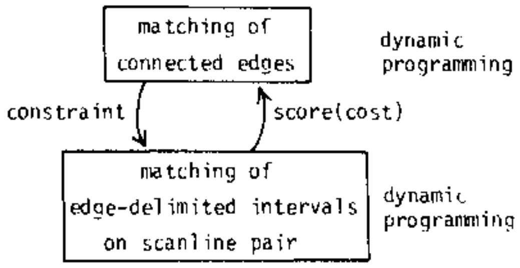

matching of connected edges
dynamic programming
constraint
score(cost)
matching of edge-delimited intervals on scanline pair
dynamic programming

Figure 1. Two searches involved in stereo matching.

former. In connected word recognition, however, the pattern to be processed is a single ID vector. In our case, a connected edge crosses over multiple scanlines (ie., ID vectors). This means that we need a 3D search space which is a stack of 2D search planes for intra-scanline matching.

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 combinatorical 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 them. 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.

3. Correspondence Search Using Two-Level Dynamic Programming

3.1. Intra-scanline search on 2D plane

The problem of obtaining a correspondence between edges on the right and left epipolar scanlines can be solved as a path finding problem on a 2D plane. Figure 2 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 thecorresponden.ee 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 tor 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. 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 no-reversal constraints 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 image. A path has a vertex at node m=(m,a) when right edge *m* and left edge *n* are matched.

The cost of a path is defined as follows. Let D(m,k) be the minimal cost of the partial path from node K to node m. D(m,0) is the cost of the optimal path to node 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 2. It should be noted that a primitive path actually corresponds to matching the intervals delimited by edges at the start and end nodes rather than edges themselves. The cost of a path is the sum of those of its primitive paths. Let d((m,k) be the cost of the primitive path from node k to node m. (Our actual definition of d(m,k) will be given in section 4.) Obviously, d(m,k)>D(m,k) and on an optimal path ^(m,k)sD(m,k).

Now, D(m,k) can be defined recursively as:

$$D(\mathrm{m},\mathrm{k}) = \min_{\{\mathrm{i}\}} \left(d(\mathrm{m},\mathrm{m}-\mathrm{i}) + D(\mathrm{m}-\mathrm{i},\mathrm{k}) \right)$$

$$D(\mathrm{k},\mathrm{k}) = 0 \qquad (1)$$

where $\mathrm{m}=(m,n)$, $\mathrm{k}=(k,l)$, $\mathrm{i}=(i,j)$, $0 \le i \le m-k$, $0 \le j \le n-l$, $i+j \ne 0$.

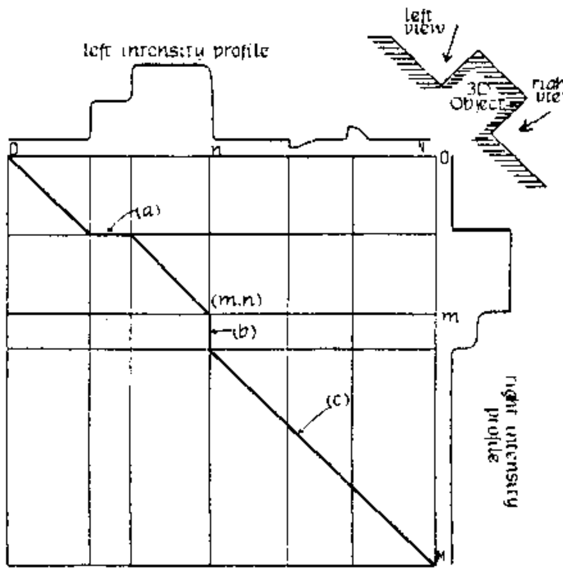


Figure 2. 2D search plane for intra-scanline search.
Intensity profiles are shown along each axis. The horizontal axis corresponds to the left scanline and the vertical one corresponds to the right scanline. Vertical and horizontal lines are the edge positions and path selection is done at their intersections.

Vector i represents a primitive path coming to node m. When i=0, the primitive path is horizontal, as shown at (a) in figure 2. It corresponds to the case in which a visible part in the left image is occluded in the right image. When y-0, the primitive path is vertical, as shown at (b). When *i>l* and/or *J>1,* the primitive path skips or ignores beyond *i-l* and/or y'-l 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 path with cost **D(M,0)** gives the optimal correspondence between a pair of scanlines.

3.2. Inter-scanline starch 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. Figure 3 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 (ie., 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

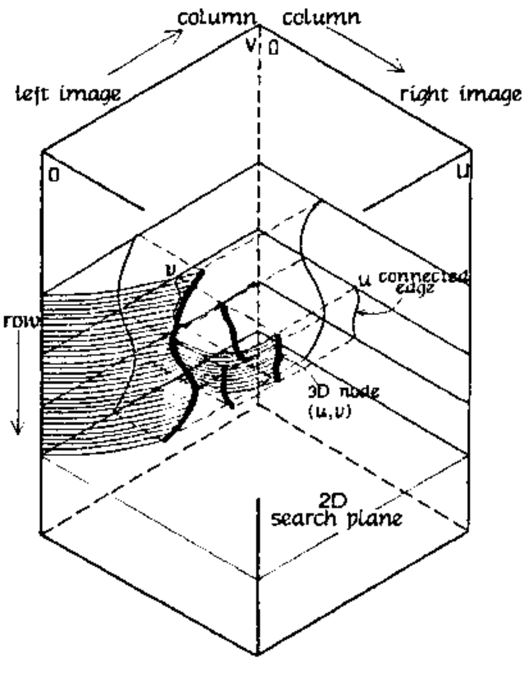


Figure 3. 3D search space for intra- and inter-scanline search.
This may be viewed as a rectangular solid seen from above. The side faces correspond to the right and left stereo images. Connected edges in each image form sets of intersections (nodes) in this space. Each set is called a 3D node. Selection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

to this set of 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.

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 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 computed 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 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 $_{u1}$ 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 The order of two connected edges which do not satisfy both the relations described above 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 is devised so that it does not make such cases.

Now we will present how the cost of a 3D path is defined. Suppose we assign indices $[0:U]$ 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's. Let $\mathbf{u}=(u,v)$ be a 3D node made by a connected edge u in the right image and a connected edge v in the left image. Let $C(\mathbf{u})$ be the cost of the optimal set of paths which reach to the 3D node $\mathbf{u}$. The cost $C(\mathbf{u})$ is computed as follows:

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

$$C(\mathbf{0}) = 0, \quad \text{i.e., } C(\mathbf{0};t)=0 \text{ for all } t \tag{2}$$

where $\mathbf{u}=(u,v)$, $\mathbf{i}(t)=(i(t),j(t))$, $0\le i(t)\le u$, $0\le j(t)\le v$, $i(t)+j(t)\ne 0$.

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

The iteration starts at u=(0,0) and computes C(u) for each 3D node u in ascending order of 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«-(C/I/) (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.

3.3. Consistency constraints in int«r-scanlin«

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 constraints as the relation between *\(t)* and i(f-l) in equation (2). To do this, let us consider the example in figure 4. Suppose we are trying to obtain a set of 3D primitive paths which reach 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-l). The cases when the starting point is 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(f-l) disappears on scanline t and the starting point is forced

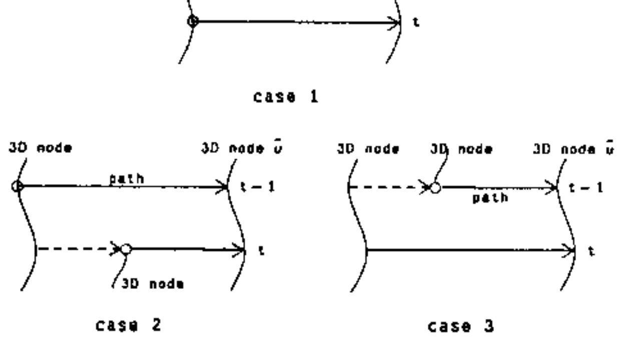


Figure 4. Three cases for consistency constraint.

to move elsewhere.

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

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

The rules in case2 and case3 require that the decision at 3D node u depend on decisions at preceding 3D nodes. Unfortunately, a decision system with such a property is not *Markov urn.* 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, we modify the rules in (3) as follows.

$$\begin{aligned} case1&: \quad frm(u;t) = frm(u;t-1) \\ case2&: \quad frm(u;t) \geq frm(u;t-1) \\ case3&: \quad frm(u;t) \leq frm(u;t-1) \end{aligned} \tag{4}$$

The new rule for case2 requires 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 case3, 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 equation (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 represented by the rules in equation (4) are weaker than those of equation (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. Experiments

Implementation of the stereo algorithm which has been presented requires a method of detecting edges, linking them into connected edges, and ordering the connected edges. We do not describe, however, the details of the method in this paper because of space limitation and it can be found elsewhere [9J

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 $a_1 \ldots a_k$ and $b_l \ldots 6_1$ 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:

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

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

$$d = \sigma^2 \times (k^2 + l^2)^{1/2} \tag{6}$$

We have applied our stereo algorithm to images from various domains including synthesized images, urban aerial images, and block scenes. Only a result with urban aerial images is presented here.

The stereo pairs used are aerial photographs of the Washington, D.C. area. They have been rectified using the camera models which was computed by Gennery's program [5] using manually selected point pairs.

Figures 5, 6, and 7 show the original stereo pair, edges, and connected edges for the "white house" scene, respectively. The image size is 388x388 pixels and the intensity resolution is 8 bits. This example is an interesting and difficult one because it includes both buildings and highly textured trees. 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 8. 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. Considerable improvements can be observed at the boundaries of buildings. In the textural part, the two algorithms provide approximately the same results.

We counted the number of positions where the consistency constraint, described in section 3.3, is not satisfied. It is 436 in the 2D search and 32 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 for the constraint as described earlier.

5. Conclusion

In this paper, we have described a stereo algorithm which searches for an optimal solution in a 3D search space using dynamic programming. The algorithm has been applied to urban aerial images successfully. Perhaps one of the major reasons that our algorithm works well for such a complex 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

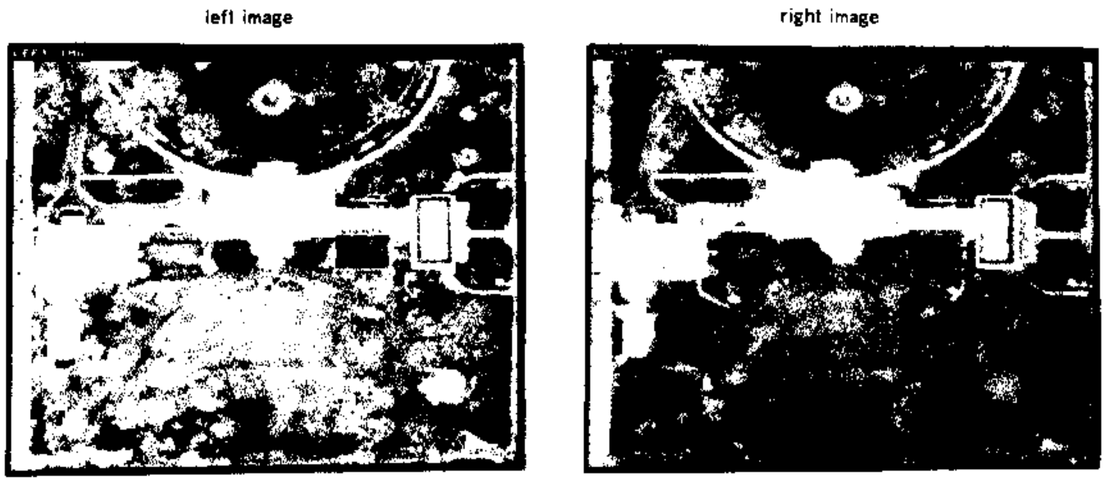


Figure 5. The "white house" stereo pair of urban images.

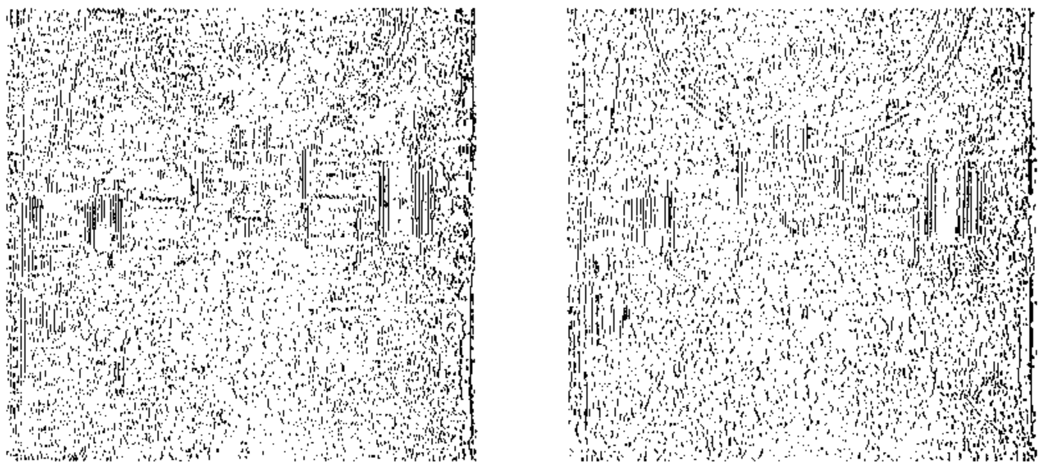


Figure 6. Edges extracted from the images in figure 5.

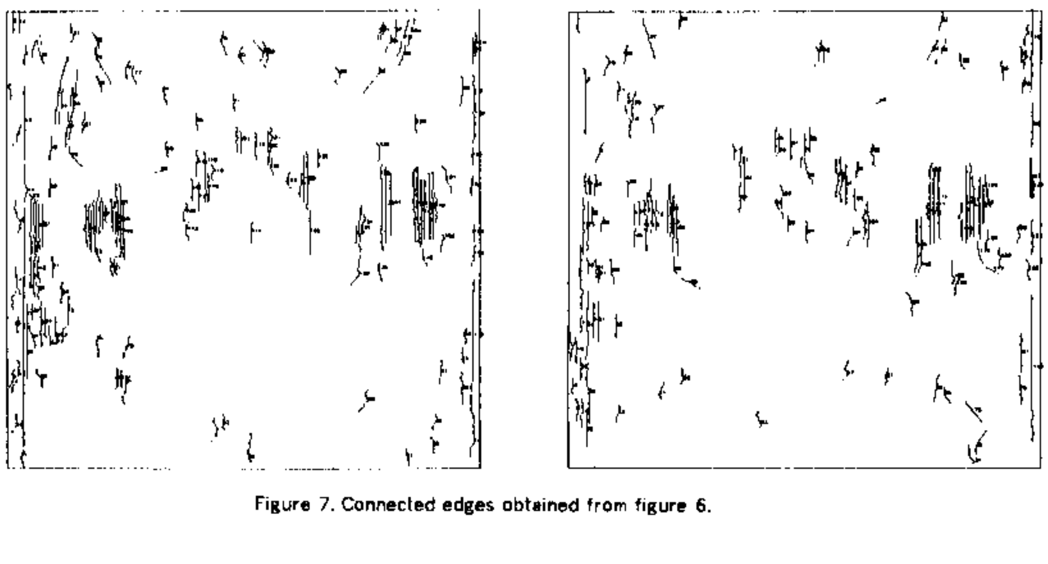


Figure 7. Connected edges obtained from figure 6.

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 search. This feature will be less obvious in segment-based algorithms, such as in [8], which depend heavily on the connectivity of edges.

References

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

[2] Baker,H.H
Depth from Edge and Intensity Based Stereo.
Technical Report AIM-347, Stanford A.I. Lab., 1982.

[3] Baker,H.k and BinfordJ.O.
Depth from Edge and Intensity Based Stereo.
In Proc. 7th IXAI, pp.631-636, Aug., 1981.

[4] Barnard,S.T. and Fischler,MA
Computational Stereo.
Computing Surveys 14(4), pp553-572, Dec, 1982.

[5] Gennery,D.
Stereo-Camera Calibration.
In Proceedings of Image Understanding Workshop, pp.101-107, DARPA, Nov., 1979.

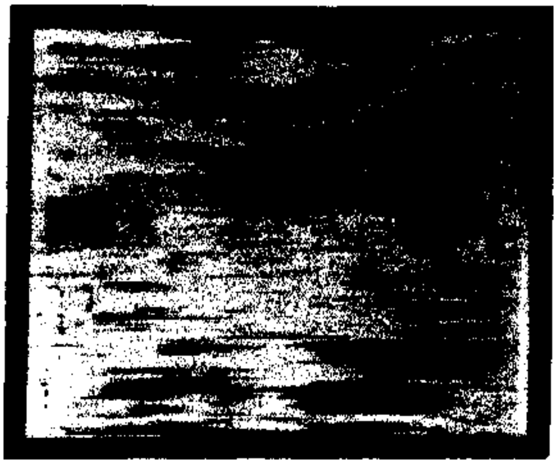
[6] Grimson,W£L. and Marr,D.
In Proceedings of Image Understanding Workshop, pp.41-47, DARPA, Apr., 1979.

[7] Henderson,R.L, Miller,W.I, and Grosch.C.B.
Automatic Stereo Reconstruction of Man-made Targets.
SPIE 186(6), pp.240-248, 1979.

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

[9] Ohta,Y. and KanadeJ.
Stereo by Intra- and Inter-scanline Search Using Dynamic Programming.
Technical Report, CMU-CS-83-162, Carnegie-Mellon University, 1983.

[10] Sakoe,H.
Two-Level DP-Matching - A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition.
IEEE Trans. ASSP, 27(6), pp.588-595, Dec, 1979.


(a) result of 2D search

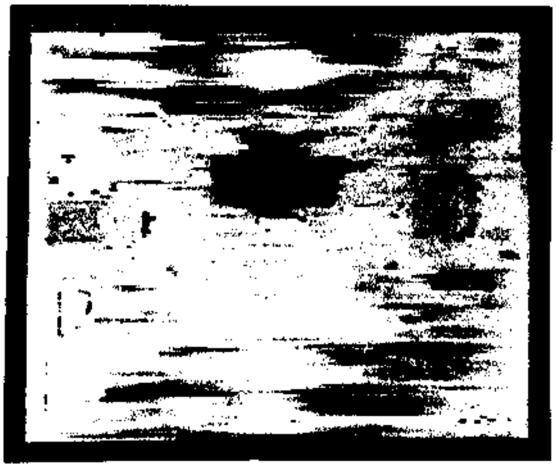

(b) result of 3D search

Figure 8. Disparity map obtained for the "white house" stereo pair (figure 5).
Both are registered in the right image coordinates.