CMU-CS-82-139

Incremental Acquisition of a Three-dimensional Scene Model from Images

Martin Herman

Takeo Kanade

Shigeru Kuroe

October 1982

Computer Science Department

Carnegie-Mellon University

Pittsburgh, PA 15213

This research was sponsored 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.

Abstract

We describe the current state of the **3D Mosaic** *project, whose goal is to incrementally acquire a 3D model of a complex urban scene from images. The notion of incremental acquisition arises from the observations that (1) single images contain only partial information about a scene, (2) complex images are difficult to fully interpret, and (3) different features of a given scene tend to be easier to extract in different images because of differences in viewpoint and lighting conditions. In our approach, multiple images of the scene are sequentially analyzed so as to incrementally construct the model. Each new image provides information which refines the model. We describe some experiments toward this end. Our method of extracting 3D shape information from the images is stereo analysis. Because we are dealing with urban scenes, a junction-based matching technique proves very useful. This technique produces rather sparse wire-frame descriptions of the scene. A reasoning system that relies on task-specific knowledge generates an approximate model of the scene from the stereo output. Gray scale information is also acquired for the faces in the model. Finally, we describe an experiment in combining two views of the scene to obtain a refined model.*

1. Introduction

The goal of the **3D Mosaic** project is to automatically acquire a detailed 3D description (or model) of a complex urban scene from images. We are currently working with aerial photographs of Washington, D. C. Fig. 1 shows a stereo pair of images from our database.

Our approach to this problem is based on the notion of incremental acquisition of the scene model. A single image or view of a complex scene is generally not adequate for deriving a complete, accurate description of the scene. Some reasons for this are:

1. Many surfaces in the scene are occluded in any particular view.
2. Because of the complexity of an image, it would be difficult to interpret all the detailed parts.
3. Some characteristics of visible surfaces may not be as apparent in one image as in a different image. For example, it may be difficult to analyze a highly oblique surface because of lack of resolution in the image, or it may be difficult to analyze surfaces with shadows cast across them.
4. Errors in analyzing and interpreting the image may create errors and inconsistencies in the scene description.

Our method involves using multiple views of the scene in a sequential manner. A partial description is derived from each view. As each successive view is analyzed, the model of the scene is incrementally updated with information derived from the view. The model is initially an approximation of the scene, and becomes more and more refined as new views are processed. At any point along its development, the model should be usable for the following types of tasks:

1. When information is derived from a new view, it must be matched to the model so that updating can occur. The model should, therefore, contain information that facilitates this matching.
2. The model should permit higher-level components to determine which parts of the scene should be analyzed in more detail, and whether a different view is required for further analysis of these parts.
3. The model should be usable in its task domain, e.g. for photointerpretation or display generation.

In our approach, 3D features that are relatively inexpensive to obtain, such as certain corners of buildings, are extracted from the images. A model of the scene is then hypothesized from these features by utilizing task-specific knowledge (e.g. block-shaped objects in an urban scene). Updating and refinement of the model is facilitated by remembering which parts of the model have been hypothesized and which parts have been directly derived from the images.

There are several applications we have in mind for the types of models that are acquired. The first involves model-based photointerpretation. A scene model can provide significant help in interpreting images of the scene taken from arbitrary viewpoints [4, 15]. Furthermore, the analysis results can be used in the incremental acquisition loop to update and refine the model. Another area of application deals with generating flight plans (simulating the appearance of the scene along potential flight paths) or familiarizing personnel with a given area. Our methods provide the ability to acquire a model of a scene from only a few views and then generate arbitrary views from the model. Finally, our incremental **3D Mosaic** approach should be applicable to robot navigation and manipulation tasks. The ability to incrementally acquire approximate descriptions of complex environments could prove useful for these tasks, since these descriptions may then be used to make decisions dealing with path planning or determining which parts of the environment to analyze in more detail.

In the rest of this paper, we first discuss how to extract a 3D scene description from a single view. The stereo pair of images shown in Fig.1 constitutes the single view to be considered. Afterward, we discuss combining information from multiple views.

2. Stereo Analysis

Our current method of extracting 3D shape information from the images is via stereo analysis. In the future, we may add other methods, such as shadow analysis [16].

Our approach to the stereo matching problem is to match junctions and lines found in the images. There are several reasons for this:

1. Our goal is to recover the 3D structure in the scene. We approach this problem by first extracting 3D information dealing with vertices and edges in the scene. In an urban scene, vertices often correspond to corners of buildings. Therefore, by recovering scene vertices and edges that emanate from them, we obtain portions of boundaries of the buildings. These boundaries then allow us to construct 3D approximations of the buildings. (See [10] for a different approach developed for the same task domain.)

2. Our stereo images are fairly wide angle and the scene consists of tall buildings. As a result, there are large discontinuities in disparity and the appearance of many objects differ significantly in the two images. This has caused problems for most previous stereo matching techniques since there are large portions of the scene that are visible in one image but not in the other. In our approach, we are not interested in matching scene faces that are occluded in one of the image pairs. Rather, our goal is to match face boundaries that are visible in both images. We do this by explicitly taking into account the way junctions change from one image to the other. We find a junction in one image, use

task-specific constraints to predict its appearance in the other image, and search for the corresponding junction by making use of the predicted appearance. Currently, our method of predicting junction appearances is based on the following task-specific knowledge:

a. In aerial photography, image planes tend to be almost parallel to the ground plane.

b. In urban scenes, roofs of buildings tend to be almost parallel to the ground plane, while walls tend to be perpendicular to this plane.

c. Therefore, features lying on a roof or on the ground will maintain the same shape in both images. Edges in the scene that are perpendicular to the ground plane will appear in each image to be directed toward the origin, defined by the intersection of the camera axis with the image plane [11].

If an L junction is found in one image, it is initially assumed to arise from a corner of a roof, and thus its appearance in the other image can be predicted. If an ARROW or FORK junction is found, the line directed toward the origin is initially assumed to arise from a scene edge which is perpendicular to the ground, while the other two lines of the junction are initially assumed to arise from scene edges lying on a roof or on the ground. Again, its appearance can be predicted.

3. Many stereo systems have trouble with wide angle stereo images because they rely heavily on local similarities in the two images [2, 3, 9, 12, 13]. In our approach, however, because the junction is intended to represent a structural component in the scene, we also rely on more global, structural similarities in the two images to perform the matching.

4. For a scene with many occlusion boundaries, an approach based on feature matching results in much more accurate 3D positions for these boundaries than an approach based on gray scale area matching.

2.1. Steps in Stereo Analysis

Exracting lines. The first step in the stereo analysis is to extract linear features. A 3x3 Sobel operator is used to extract edge points, as shown in Fig. 2. Then the edges are thinned using a modified Nevatia and Babu algorithm [14], as shown in Fig. 3. The resulting edge points are linked and straight lines are fitted to them. The method used to fit straight lines to a set of linked points is based on iterative end-point fitting [7]. However, since this method determines a line using only two end points, the line equation for the set of points is recalculated using least squares. Finally, short lines are discarded. The resulting line images are shown in Fig. 4.

Extracting junctions. The next step is to extract junctions from the line images. A junction is a group of lines that meet at a point, and often arises from a vertex in the scene. We consider the following four junction types: L, ARROW, FORK, and T. To find junctions, a 5x5 window around each end point of each line is searched for ends of other lines. Lines in the window that are close and nearly parallel are combined into a single line. Then, if the window contains the ends of three lines, the lines are classified as an ARROW, FORK, or T junction depending on the angles between the lines. The position of the junction point is the middle of the three end points. If a window contains the ends of two lines, the lines are classified as an L junction. The intersection of the two lines determines the position of the junction point. If a window contains more than three lines, each set of two lines is assumed to form a distinct L junction. Junctions that have been found in this manner are labeled in Fig. 5.

Find potential junction matches. The next step in the stereo analysis is to match the junctions found in one image with those in the other. Let us consider how L junctions are matched. As explained previously, each L junction in one image is initially assumed to lie on a plane which is almost parallel to the camera image planes. The shape and orientation of its corresponding junction in the other image, therefore, can be predicted. Each L junction in the first image may be matched with several junctions in the second image that lie along the corresponding epipolar line and that have, within tolerance, the predicted shape and orientation. An interesting point is that we do not try to match only with junctions in the second image that have been previously found. Rather, the shape and orientation of the corresponding junction in the second image is predicted for every point lying on the epipolar line (on the appropriate side of the infinity point), and at each of these points, a search is made within a pre-specified window for lines that might correspond to the predicted junction. The requirements, however, for two lines to be a junction is more relaxed than the requirements during initial junction search. We therefore improve feature detection in each image by using the features found in one image to predict features in the other image. (Matching is performed in two directions, from the first image to the second, and vice versa.)

To match ARROW, FORK, and T junctions, each pair of lines forming the junction is treated as if it were an L junction and matched in the manner described above.

Search for unique junction matches. Next, a beam search [15] is used to arrive at a unique combination of junction matches. There are two factors involved in computing costs for the various combinations of matches:

1. Local cost between two potentially matching junctions is computed by the similarity of the image intensities inside the junctions. The assumption here is that the two junctions will have similar intensities if they arise from the same face corner.

2. Global cost is based on the consideration that if there are two vertices in the scene with the same heights, the positional relationship between their corresponding junctions in one image will be the same as in the other image. This is due to the image planes being

parallel to the ground plane. We make the assumption that junctions which are close to one another will often correspond to vertices lying on top of the same building, thus having approximately the same height. Global cost between two potentially matching junctions is therefore computed by the similarity of the configuration of the neighborhoods around the junctions.

The matching procedure is applied from the first image to the second and vice versa. The results are then merged. Fig. 6 shows junctions and lines in one image that have matches in the other image.

Search for third legs of junctions. The next step is to find lines in the images that might be the third leg of matched junctions and that might represent scene edges perpendicular to the ground plane. The method used is to find lines near the junctions in both images that are directed toward the origin.

Generate 3D wire frames. Finally, 3D coordinates are derived using triangulation. Fig. 7 shows a perspective view of the 3D vertices and edges that result. We call this a wire-frame description of the scene.

3. Representing and Modifying the 3D Scene Model

Our requirement that the scene model is to be incrementally acquired leads to several issues: (1) representing partial constraints on 3D structure, (2) incremental accumulation of these partial constraints, and (3) handling discrepancies in information acquired at different times.

Our approach involves representing the model in a modular manner. Constraints on 3D structure are represented in the form of a graph, called the *structure graph*. The nodes and links represent primitive topological and geometric constraints. The structure graph is incrementally constructed through the addition of these constraints. As constraints are accumulated in the graph, their effects are propagated to other parts of the graph so as to obtain globally consistent interpretations.

3.1. Representation of Model

The current structure-graph representation models surfaces in the scene as polyhedra. The components of a polyhedral surface are the face, edge, and vertex. We distinguish the topology of the polyhedral components from their geometry [1, 8]. The geometry involves the physical dimensions and location in 3-space of each component, while the topology involves connections between the components.

In the structure graph, nodes represent either primitive topological elements -- faces, edges, vertices, objects, and edge-groups (which are rings of edges on faces) -- or primitive geometric elements -- planes, lines, and points. Vertex, face, and edge nodes are tagged as either *confirmed* or *unconfirmed*. Confirmed means that the element represented by the node has been derived directly from the images. Unconfirmed means that the element has only been hypothesized.

The primitive geometric elements serve to constrain the 3-space locations of faces, edges, and vertices. Plane and line nodes contain plane and line equations, respectively. Point nodes contain coordinate values. The graph contains two types of links: the *part-of* link, representing the part/whole relation between two topological nodes, and the *geometric constraint* link, representing the constraint relation between a geometric and topological node.

3.2. Modifications to Model

Modifications to the model will occur as part of the process of incremental construction. Deletions and changes are made when new information is found to conflict with information currently contained in the model. This will happen most often with portions of the model that have been hypothesized. Additions to the model are made to incorporate the new information as part of the model.

Modifications to the structure graph are made by adding or deleting nodes and links, or changing the equations of line and plane nodes, or the coordinates of point nodes. All effects of modifications are propagated to other parts of the graph.

As an example, consider adding or deleting a geometric constraint link between a geometric and topological node. Any of the three geometric nodes -- points, lines, and planes -- may constrain any of the three topological nodes -- vertices, edges, and faces. Fig. 8 shows how a constraint on one node may propagate to others. The arrows in the figure indicate the direction of propagation. For example, if a point constrains a vertex, it must also constrain all edges and faces containing that vertex. Similarly, a point that constrains an edge also constrains all faces containing that edge.

When a geometric constraint link is deleted, the rest of the structure graph must be made consistent with this change. Our approach to this problem is based on the TMS system [6], using the notion that when an assertion is deleted, all assertions implying it and all assertions implied by it should also be deleted, unless they have other support. Assertions that imply a given assertion are obtained by following backwards along the arrows in Fig. 8. Assertions implied by a given assertion involve following forward along the arrows.

Consider the example in Fig. 9a, which depicts three topological nodes (vertex *v*, edge *e*, face *f*) constrained by one geometric node (point *p*). Suppose now that link 4 is deleted (Fig. 9b), i.e.,the assertion "*p* constrains *e*" is deleted. To find the assertion that might imply this one, locate the box in Fig. 8 that represents a point constraining an edge, follow backwards along the arrow, and the result is the box that represents the point constraining any vertex of the edge. In Fig. 9b, this represents the assertion "*p* constrains *v*, and *v* is part of *e*". This assertion must therefore be made false. To do so, we may delete either link 1, link 3, or both from Fig. 9b. We have arbitrarily decided that part-of links should dominate constraint links, and thus link 3 is deleted. This seems to work well for our examples.

We now must determine the assertions implied by the one initially deleted. We follow forward along the arrow from the box in Fig. 8 that represents a point constraining an edge, and the result is the box that represents the point constraining all faces containing the edge. In Fig. 9b, this represents the assertion "*p* constrains *f*", which is link 5. This link should therefore be deleted because it has no other support. The resulting structure graph is depicted in Fig. 9c.

4. Generating the 3D Scene Model

We now present an example showing how the scene model is generated from the output of the stereo analysis component. We start with the 3D wire-frame description shown in Fig. 10. The final model derived is a surface-based description.

Combine edges. First, if there are two wire-frame edges that are nearly parallel and very close to each other, they are merged into a single edge. This occurs only once in Fig. 10, for the two edges labeled E1 and E2.

Generate web faces. Next, each vertex is assumed to correspond to a corner of an object. Therefore each adjacent pair of legs ordered around the vertex corresponds to the corner of a planar face. Thus far in our experiments, we have dealt only with trihedral vertices. In this case, every pair of legs of each vertex corresponds to the corner of a separate face. A partial face, called a *web face*, is generated for each such pair.

Merge partial faces. After all web faces have been created, those that represent the corners of a single face are merged. Two partial faces that contact each other (e.g. F1 and F2 in Fig. 10) should be merged if (1) they share exactly one edge, (2) the edge serves as a boundary of both faces, but does not partition them, and (3) the planes of the faces are nearly parallel and very close to each other.

Two partial faces that do not contact each other (e.g. F3 and F4 in Fig. 10) should be merged if (1) each face has a single chain of edges that is not closed, (2) each of the two end points of the edge chain of one face must be uniquely matched with those of the other face, where unique matching is determined by the distance between the two points being less than a threshold, and (3) the planes of the faces are nearly parallel and very close. When merging the two non-contacting faces, the two edges on which each matching pair of end points lie are extended in space and intersected. The intersection points form two new vertices on the resulting face.

Complete the shapes of faces. After all mergers have been performed, many faces may still be incomplete, i.e., they do not have a closed boundary. In these cases, task-specific knowledge is used to hypothesize the shape of each face, and it is completed by generating the appropriate edges and vertices. The rules used here are:

1. If the partial face consists of a single corner, i.e., it contains only two connected edges, the shape is completed as a parallelogram.

2. If the partial face contains three or more edges connected as a single chain, the shape is completed by connecting the two end points of the chain with a new edge.

Find holes in the faces. After all faces have been completed, one face is assumed to represent a hole in another face if (1) the planes of the faces are nearly parallel and close to each other, and (2) the boundary of the first face, when projected onto the plane of the second face, falls inside the boundary of that face. When these conditions are met, the bounding edges of the first face are converted into an inner ring of edges of the second face.

Generate vertical faces for incomplete objects. At this point, many objects will be only partially complete because they are not closed. Task-specific knowledge may be used to add more faces to the object. Because our 3D information is obtained from aerial images of an urban scene, many faces that lie high enough above the ground plane represent roofs of buildings. For each such face, vertical walls are dropped toward the ground plane from each edge of the face, unless the edge is also part of another face. The equation of the ground plane is currently interactively obtained. A vertical wall is dropped either down to the ground plane, or to the first face it intersects on the way down.

Our procedure for dropping vertical faces from a face F is as follows. First, an edge is dropped from each vertex of F either to the ground plane or to the first face it intersects. Next, web faces are created for each new edge pair at each vertex. Newly created faces are then merged and completed in the ways described above. Fig. 11 shows several perspective views of the resulting scene model.

4.1. Comparison with Depth Map

There are several interesting points about the generated model. First, notice that it is a higher level description than a depth map. The product of most stereo analysis systems is a depth map [2, 13] which, like an image, is an array of numbers that requires description. Our approach, on the other hand, has been to extract a sparse amount of 3D information using stereo analysis (as shown in Fig. 10) and to use task-specific knowledge to go directly to a higher level 3D description. This description is much more compact than one based on surface points, and allows properties such as topology, shape, absolute size, and absolute position of scene objects to be easily available. It should therefore be easier to update and refine the model from information obtained from subsequent views. Furthermore, the model should be more useful for matching with 2D image information, with 3D information extracted from images, and with other models.

4.2. Mapping Gray Scale onto Faces

In order to render more realistic displays, gray scale is added to them [5]. This is accomplished by associating with each face in the model a normalized intensity image patch of the face. Although these patches are currently derived from a single image of the scene, we plan to generate them from multiple images. Geometric normalization, which eliminates the effects of perspective projection, is performed on the patches. We also hope to perform photometric normalization to eliminate the effects of varying illumination conditions. Fig. 12 shows the results of adding gray scale to the faces of the model. The red portions in the figure indicate faces and parts of faces that are occluded in the original image. An interesting future problem involves incrementally updating the intensity patch of a face as information is acquired from successive images. Note that the gray scale displays might also be useful in performing a 2D match between the projected image of the model and an image of the real scene.

5. Multiple Views

This section describes an experiment in combining information from two views to generate the scene description. The 3D information shown in Fig. 10 is derived from one view (viewing the scene from the "front"). Another set of vertices and edges, depicted in Fig. 13, was manually generated to simulate the information available from an opposing point of view (viewing the scene from the "back"). The viewpoints for the perspective drawings of Figs. 10 and 13 are similar to allow easier comparison by the reader. Notice that the information in Fig. 10 emphasizes edges and vertices that are facing the front of the scene, while vertices and edges facing the back of the scene are emphasized in Fig. 13.

We have made the assumption in this experiment that we know the exact positions, relative to the first view, of the cameras used to obtain the second view. Therefore, the wire-frame descriptions in Figs. 10 and 13 can be expressed in the same coordinate system. We are currently working on the problem of matching such descriptions with a model so that relative positions of views can be automatically determined.

The procedure used in this experiment is similar to the one described in the last section, except that matching and merging of the two sets of wire-frames is also required.

First, for each set of wire frames, edges that are nearly parallel and very close to each other are merged. Next, each connected group of edges is labeled as a separate wire-frame object. We now want to merge objects derived from the first view with matching objects derived from the second view. Two objects are said to match if they have matching vertices or edges. The requirements for two vertices, one from each object, to match are: (1) they must be very close together, or (2) they must be part of matching edges, and the other two vertices of the edges match. The requirements for two edges, one from each object, to match are: (1) the two vertices of one edge must match the two of the other, or (2) one vertex of one edge matches one vertex of the other, and the two edges are close together and overlap in their lengths. These rules are used in a relaxation algorithm to obtain matching vertices and edges.

Two matching wire-frame objects are merged in the following manner. First, their matching vertices are merged. The coordinates of each resulting vertex are those of the midpoint of the line connecting the two initial vertices. Next, the matching edges are merged by using a type of "averaging" to obtain a resulting edge for two initial edges that do not coincide. This averaging is based on the observation that end points of edges that are vertices generally have much more accurate 3-space positions than end points that are not vertices. Therefore, the vertex end points are given greater weight in the averaging than the non-vertex end points. Finally, all other edges and vertices of the two objects are combined to generate a single wire-frame object.

From this point onward, processing continues as described in the previous section. Web faces are generated for each corner of each vertex, the web faces are merged, the shape of incomplete faces are completed, holes in faces are found, and vertical walls are dropped from faces floating above the ground. Fig. 14 shows several perspective views of the resulting scene model.

5.1. Results with Multiple Views

There are two important differences between the scene models shown in Figs. 14 and 11. First, the one in Fig. 14 contains more buildings. This is expected because more wire-frame data is available in constructing this model. Second, many buildings that are described in both models are more accurately described in the one in Fig. 14. That is, the positions of vertices and edges of these buildings are more precise. There are two reasons for this: (1) Since more wire-frame data is available for reconstructing these buildings, we obtain high accuracy for more vertices and edges. (2) Since many vertices and edges are redundantly available in both sets of data, their positions are "averaged", generally decreasing the amount of error.

This experiment demonstrates how the information provided by each additional view allows the scene model to be gradually refined and made more complete.

In this experiment, only wire-frame objects are matched and merged. Our next step will be to match and merge a wire-frame description with a scene model. Our current experiment can also be thought of as merging wire frames with a scene model by noting that it is equivalent to having generated a model from one set of wire frames, but using only confirmed vertices and edges of the model to match and merge with the other set of wire frames. This gives an indication of the importance of confirmed information for the more general matching and merging processes. Our next step will require determining which parts of the model, both confirmed and unconfirmed, require modification. Some of these parts may actually have to be pulled apart and rebuilt, while others may merely require modifications to their 3-space locations.

6. Conclusion

The current state of the **3D Mosaic** project has been described. The goal of this project is to acquire a detailed 3D model of a complex scene from images. A useful approach to this problem is to acquire the model in an incremental manner, over a sequence of images taken from multiple viewpoints. We have also shown that task-specific knowledge is very useful in interpreting complex images. Our stereo analysis component uses such knowledge for matching features in the images, and our higher level reasoning component uses such knowledge for reconstructing shapes from the stereo output.

Fig. 15 displays a flow chart for the whole system. The stereo analysis extracts 3D wire-frame descriptions representing portions of boundaries of the buildings in the scene. A surface-based model representing an approximation of the scene is then generated from the wire-frame descriptions. This model should be useful for tasks such as matching, photointerpretation, display generation, and path planning. As indicated in Fig. 12, parts of the scene not yet observed are displayed in red. This idea can be used in a task such as planning flight paths for reconnaissance, where a path that permits viewing the maximum amount of red portions might be optimal.

There are several extensions and improvements we have in mind for our system. In addition to continuing our experiments with multiple views as discussed in the previous section, the following are our main tasks for the immediate future:

1. Using the scene model for matching. This is required for performing model-based image understanding and for updating the model with information obtained from a new view.

2. Verifying a scene model in a top-down manner by projecting hypothesized edges and vertices into the image plane and then searching for them in the image.

3. Increasing the amount and accuracy of the wire-frame information extracted during stereo analysis. More boundaries of buildings in the scene than shown in Fig. 7 can probably be extracted by directly incorporating task-specific knowledge at the lowest levels in the process of extracting junctions from the image.

Acknowledgement

Fumi Komura did much work in exploring and experimenting with initial concepts dealing with this project. Duane Williams has provided excellent programming support and many ideas. In addition, Dave McKeown, Steve Shafer, and David Smith have provided useful comments and criticism.

References

1. Baer, A., Eastman, C., and Henrion, M. "Geometric Modelling: a Survey." *Computer-Aided Design 11* (September 1979).

2. Baker, H. H., and Binford, T. O. "Depth from Edge and Intensity Based Stereo." *Proc. IJCAI-81* (1981).

3. Barnard, S. T. and Thompson, W. B. "Disparity Analysis of Images." *IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-2*, 4 (July 1980).

4. Barrow, H. G., Bolles, R. C., Garvey, T. D., Kremers,J. H., Tenenbaum, J.M., and Wolf, H. C. "Experiments in Map-guided Photo Interpretation." *Proc. IJCAI-77* (August 1977).

5. Devich, R. N., and Weinhaus, F. M. "Image Perspective Transformations." *Proc. SPIE* (July 1980).

6. Dolye, J. "A Truth Maintenance System." *Artificial Intelligence 12* (1979), 231-272.

7. Duda, R.O. and Hart, P. E.. *Pattern Classification and Scene Analysis.* John Wiley and Sons, New York, 1973.

8. Eastman, C. M., and Preiss. K. A Unified View of Solid Shape Modeling Based on Consistency Verification. Carnegie-Mellon University, September, 1981.

9. Hannah, M. J. Computer Matching of Areas in Stereo Images. Tech. Rept. AIM-239, Stanford University, July, 1974.

10. Henderson, R. L., Miller, W. J., and Grosch, C. B. "Automatic Stereo Reconstruction of Man-made Targets." *Proc. SPIE 186* (August 1979).

11. Liebes, S. "Geometric Constraints for Interpreting Images of Common Structural Elements: Orthogonal Trihedral Vertices." *Proc. Image Understanding Workshop* (April 1981).

12. Lucas,B. D., and Kanade, T. "An Iterative Image Registration Technique With an Application to Stereo Vision." *Proc. IJCAI-81* (August 1981).

13. Marr, D., and Poggio, T. "A Computational Theory of Human Stereo Vision." *Proc. R. Soc. Lond. B 204* (1979).

14. Nevatia, R. and Babu, K. R. An Edge Detection, Linking and Line Finding Program. Image Processing Institute, University of Southern California, September, 1978.

15. Rubin, S. *The ARGOS Image Understanding System.* Ph.D. Th., Carnegie-Mellon University, 1978.

16. Shafer, S. A., and Kanade, T. Using Shadows in Finding Surface Orientations. Tech. Rept. CMU-CS-82-100, Carnegie-Mellon University, January, 1982.

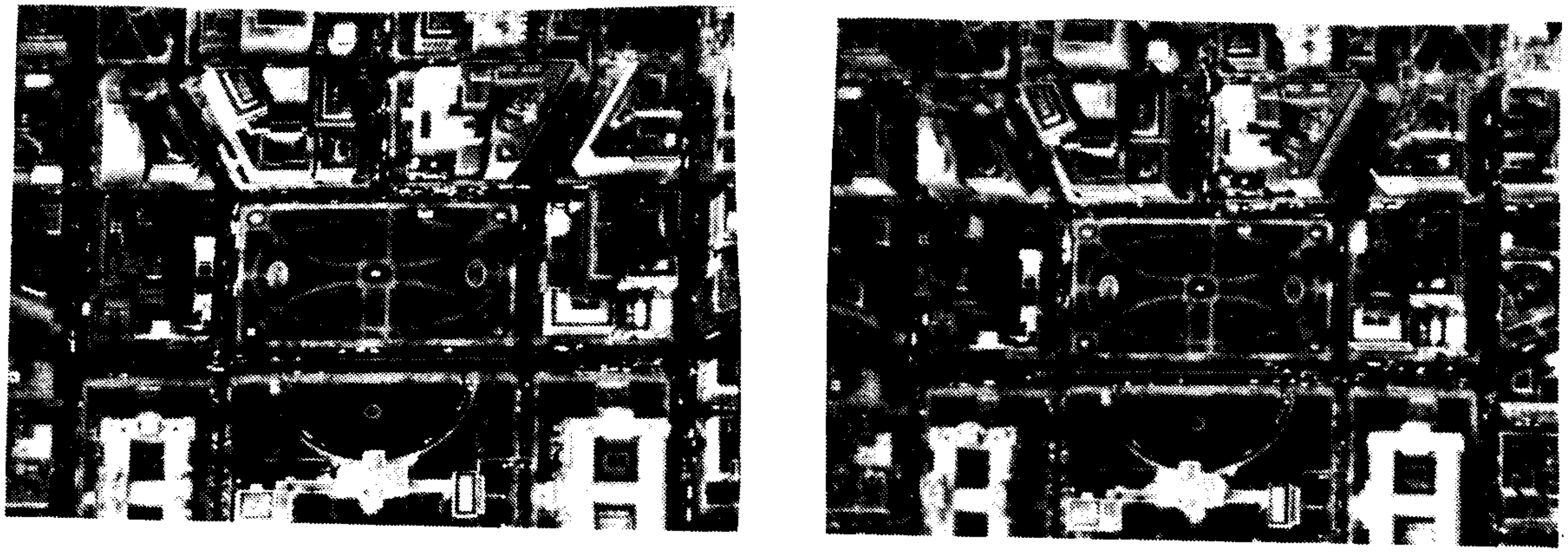


Figure 1: Gray scale stereo images of a region of Washington, D.C.

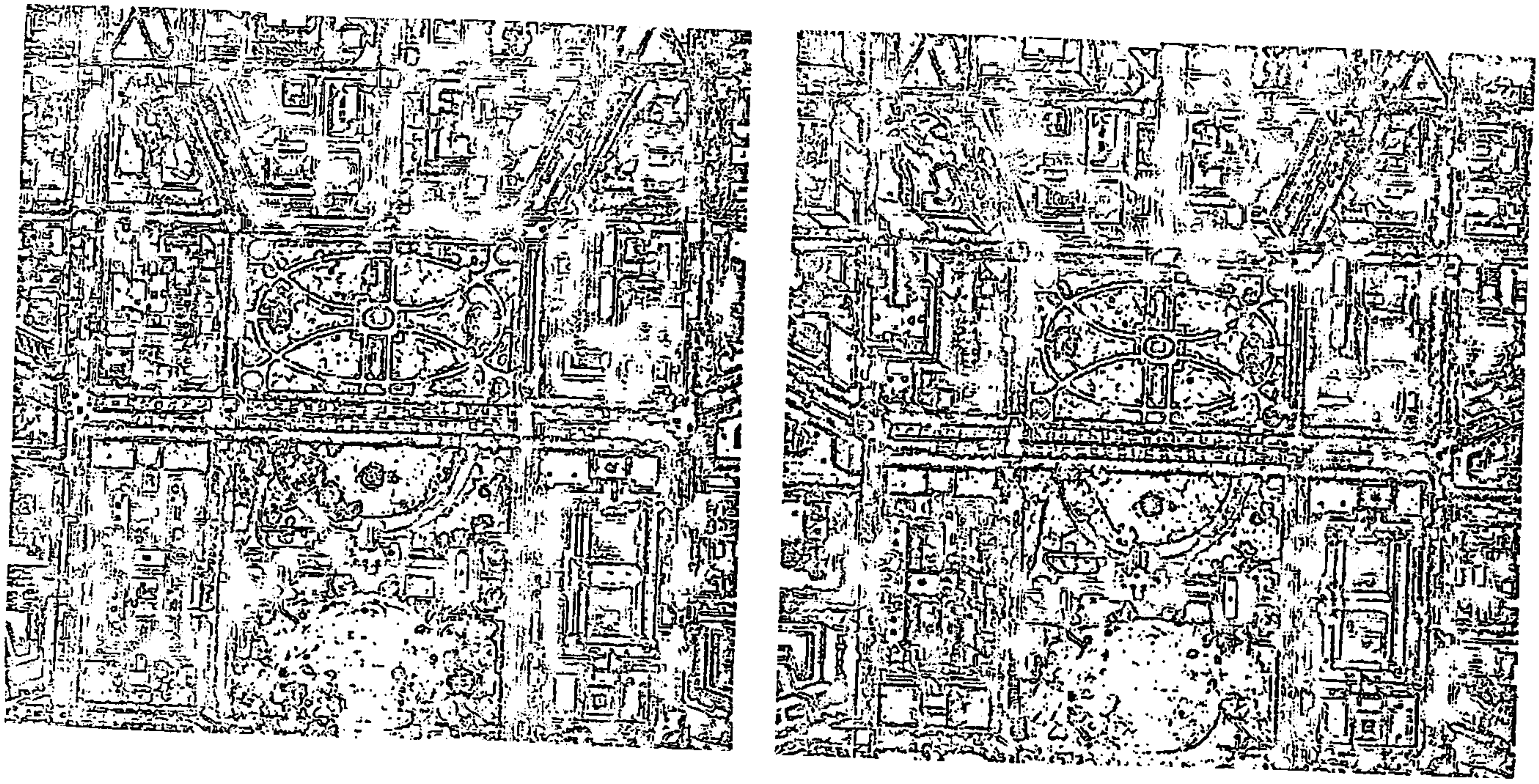


Figure 2: Edges resulting from a Sobel operator applied to the images of Fig. 1

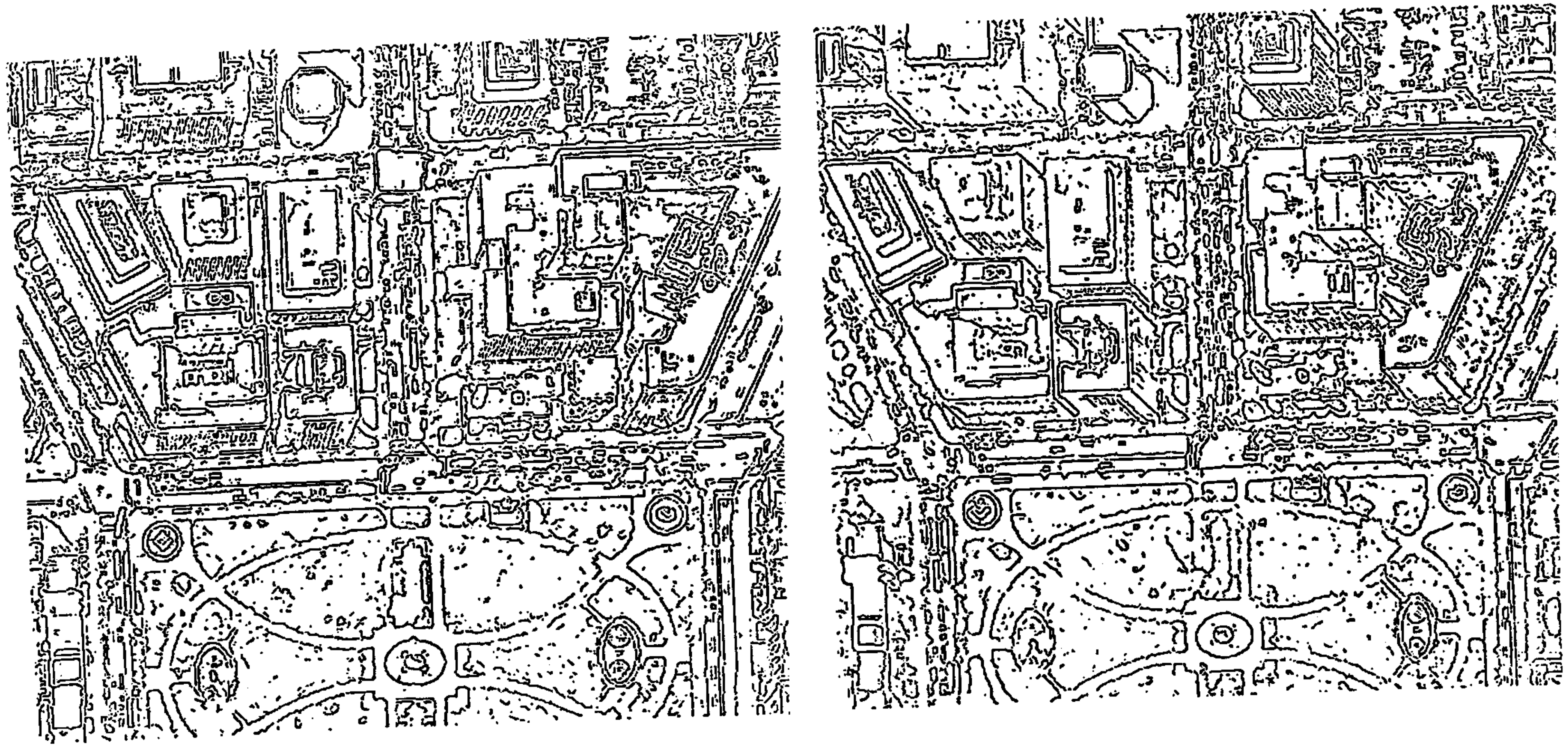


Figure 3: Result of thinning the edges in Fig. 2. Results for the upper middle part of each image in Fig. 2 are shown here.

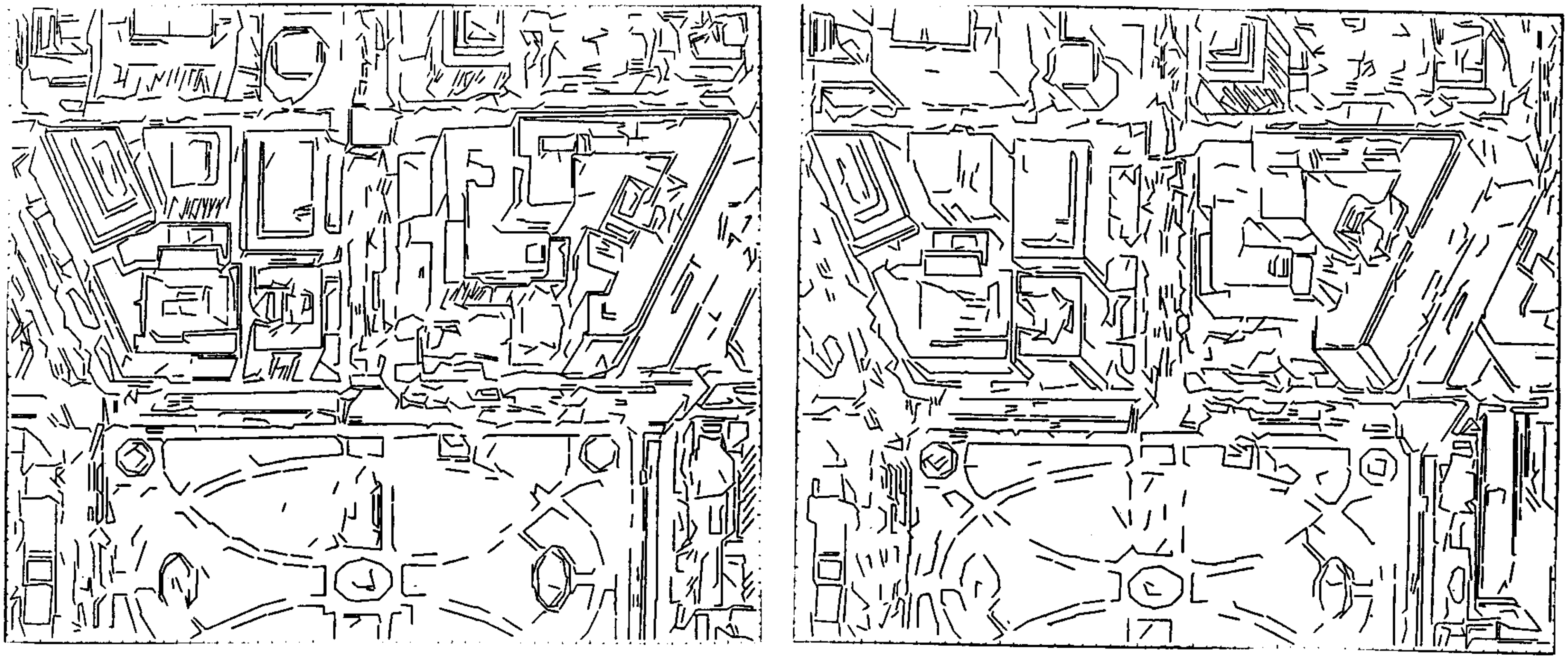


Figure 4: Straight lines fitted to the edge points of Fig. 3 after they are linked.

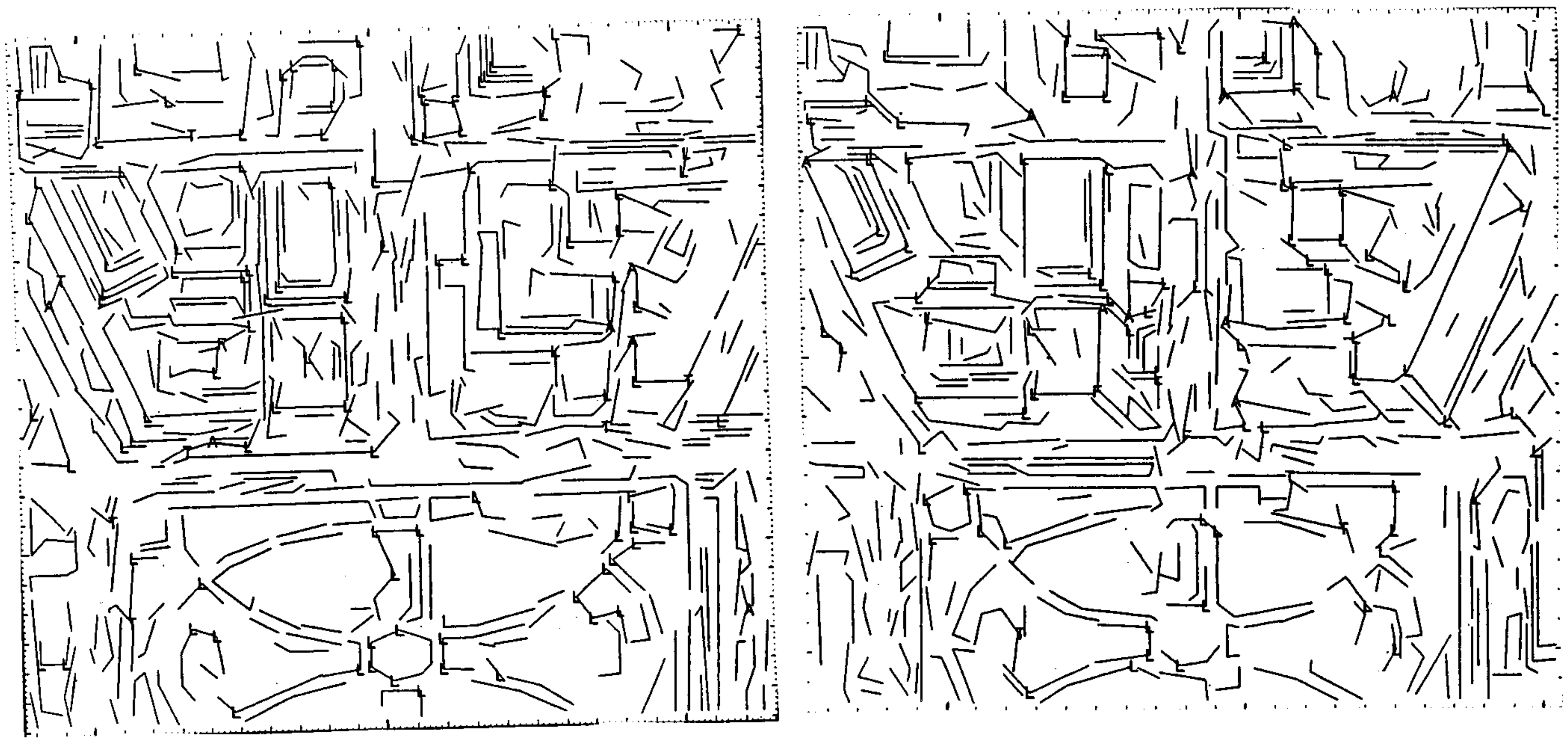


Figure 5: Result of classifying junctions in a different version of line images than shown in Fig. 4. Junctions are classified as L, A (arrow), F (fork), or T.

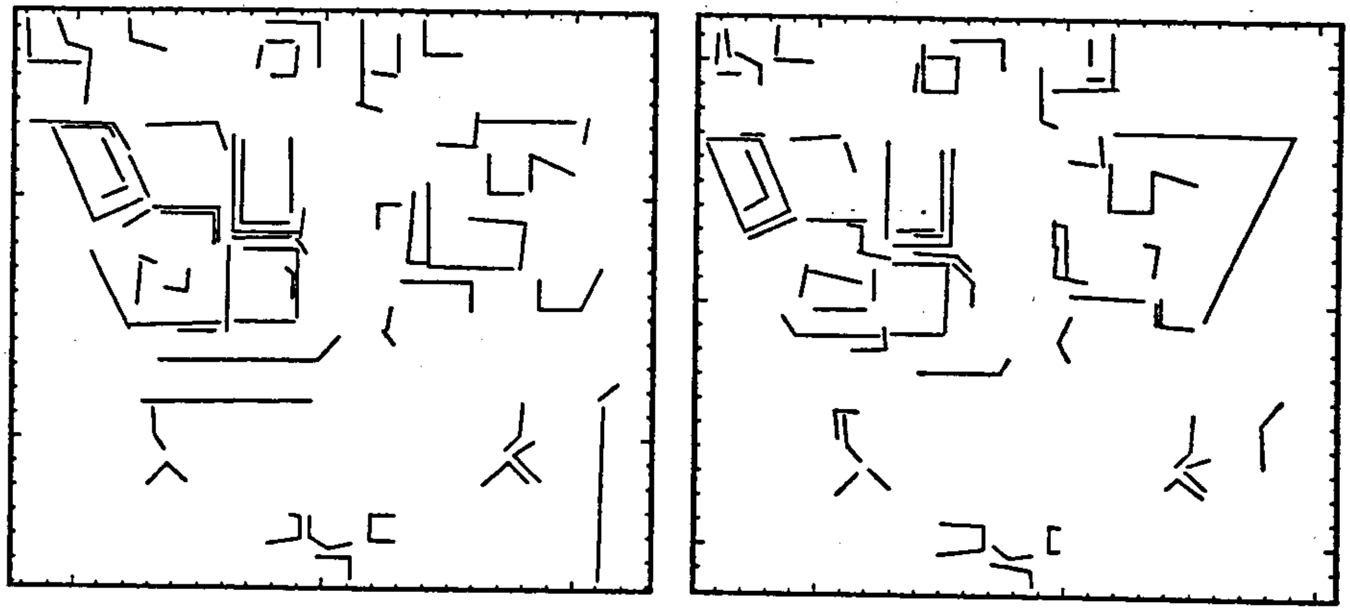


Figure 6: Matches that have been found for junctions and lines in Fig. 5.

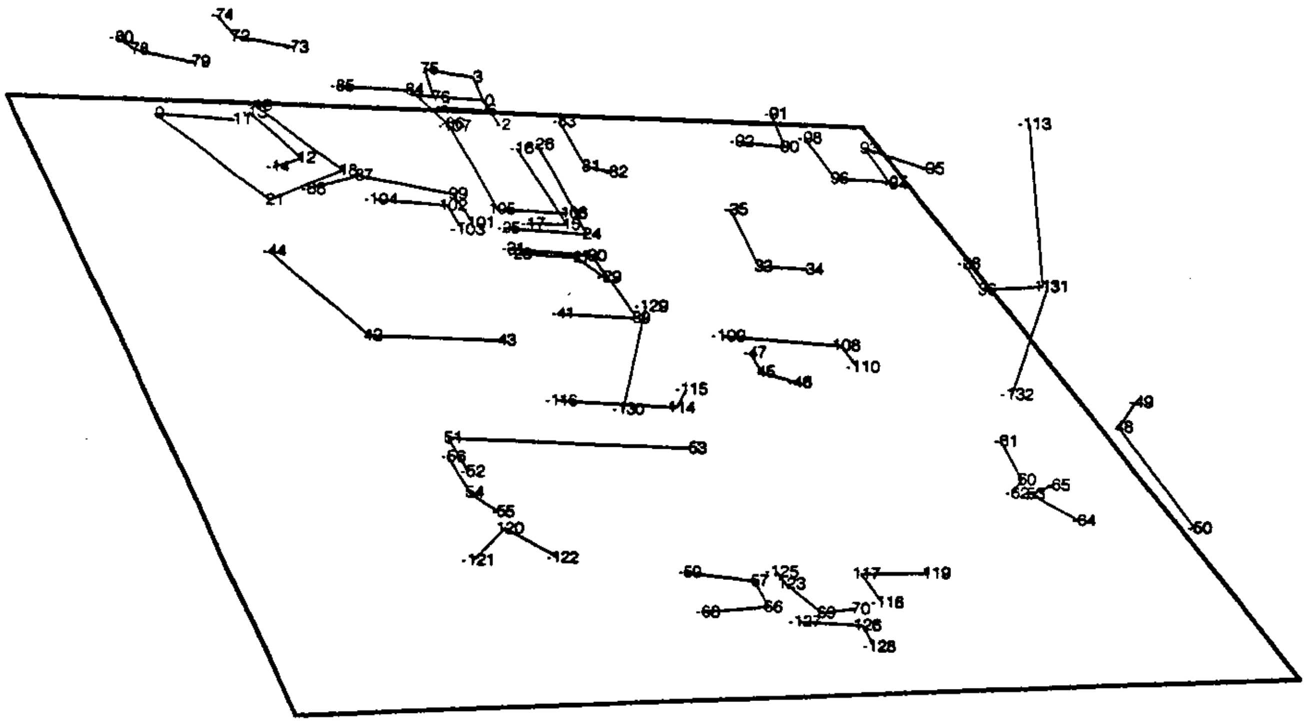


Figure 7: Perspective view of 3-D vertices and edges derived from matches shown in Fig. 6.

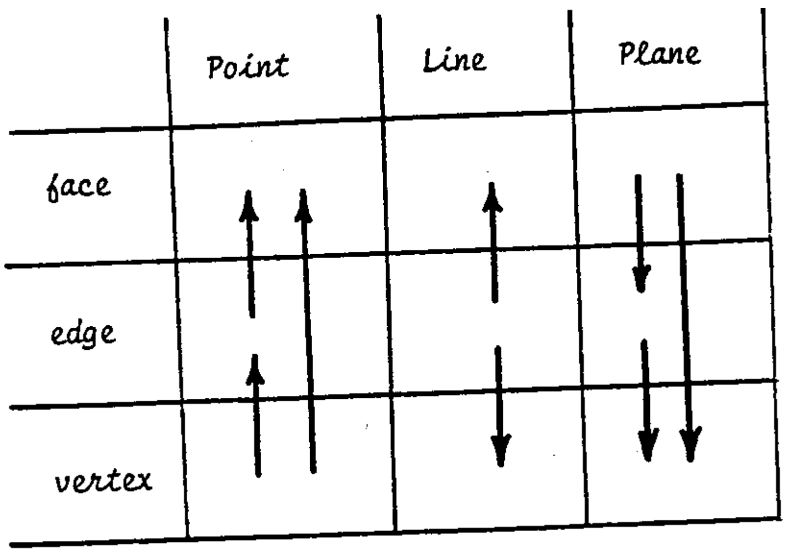
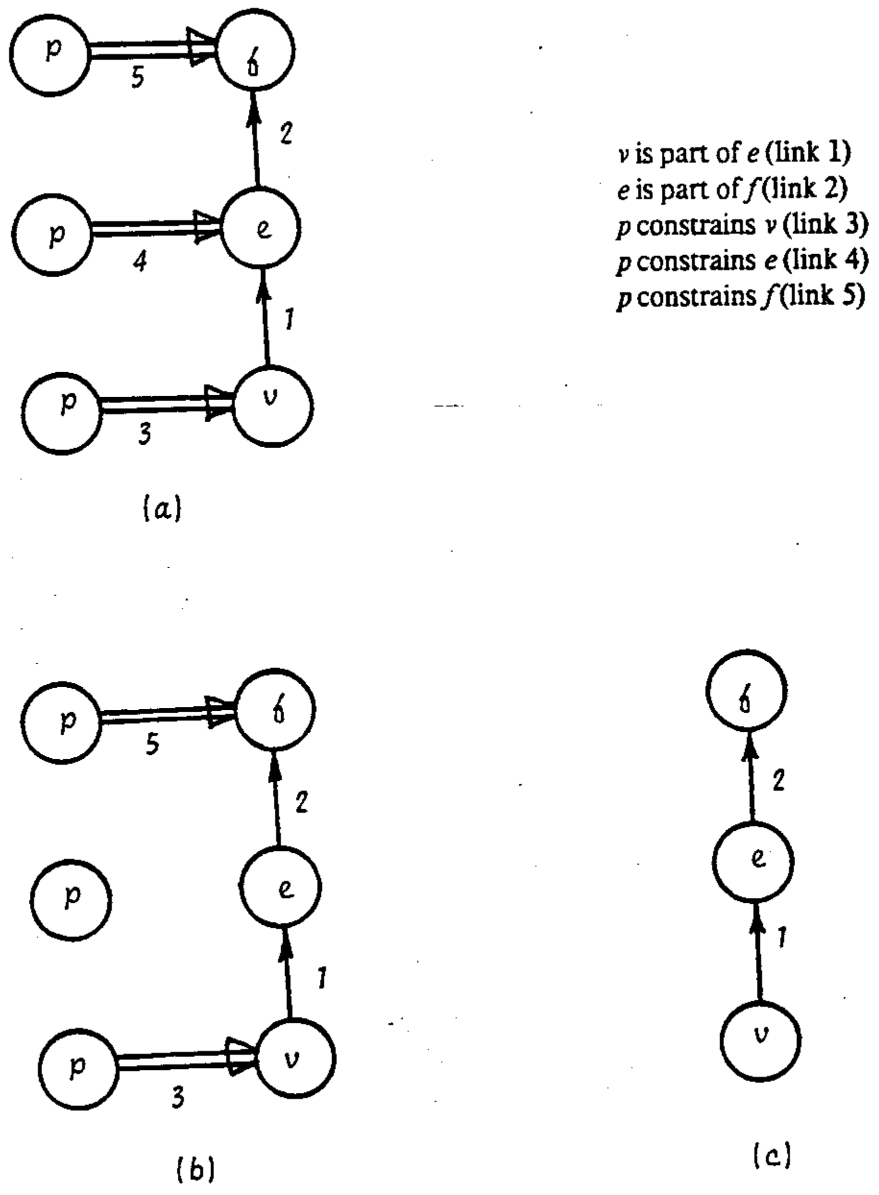


Figure 8: Rectangular boxes indicate geometric constraints on topological nodes. Arrows indicate direction of propagation of constraints.



v is part of *e* (link 1)
e is part of *f* (link 2)
p constrains *v* (link 3)
p constrains *e* (link 4)
p constrains *f* (link 5)

Figure 9: (a) Initial structure graph.(b) Link 4 is deleted.
(c) Resulting structure graph after effects of deletion have been propagated.

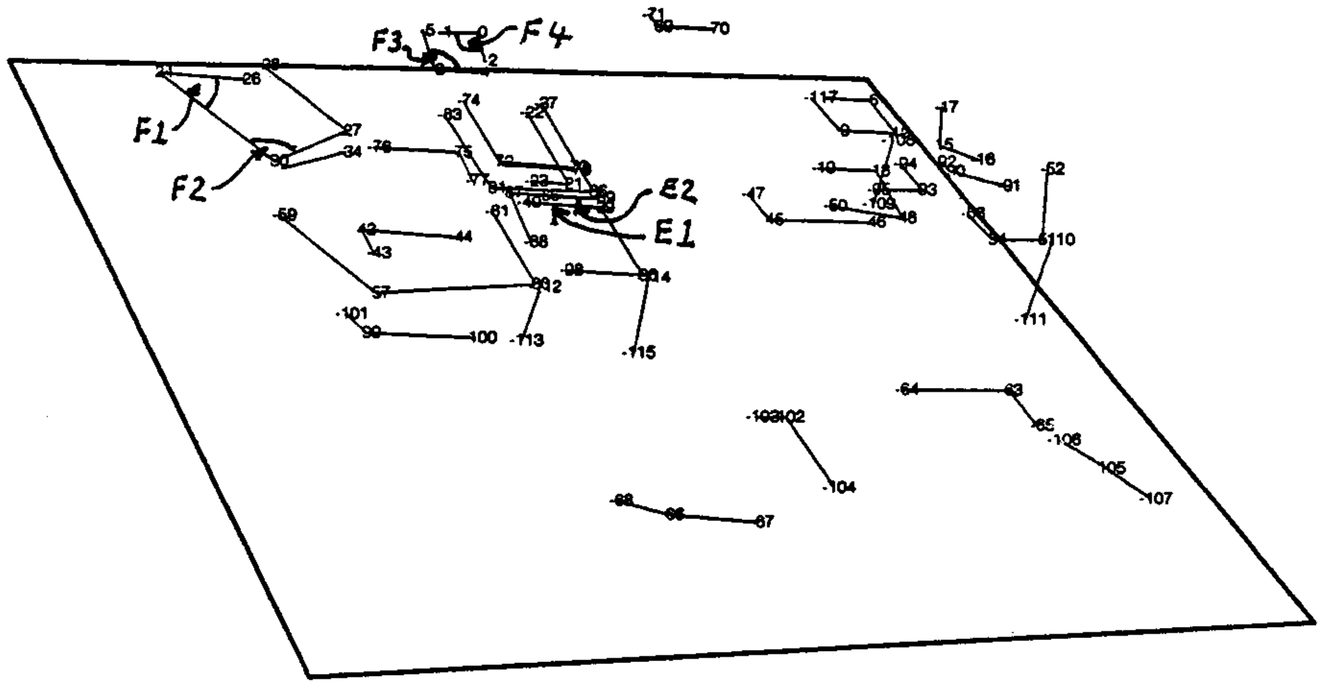


Figure 10: Perspective view of 3-D vertices and edges extracted from stereo pair. This version is different from the one shown in Fig. 7.

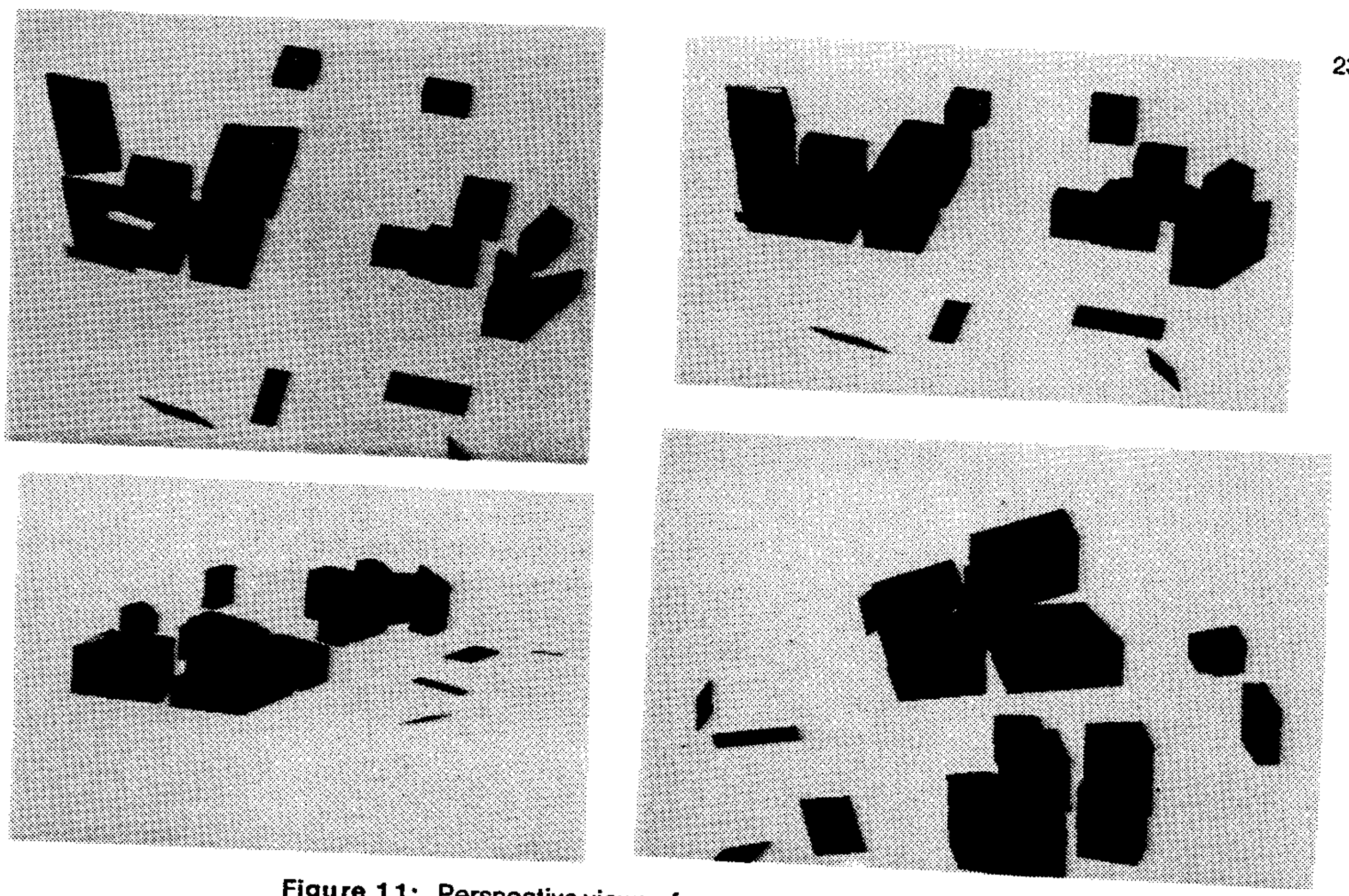


Figure 11: Perspective views of reconstructed buildings. These buildings correspond to the cluster of buildings at the upper middle parts of the images in Fig. 1.

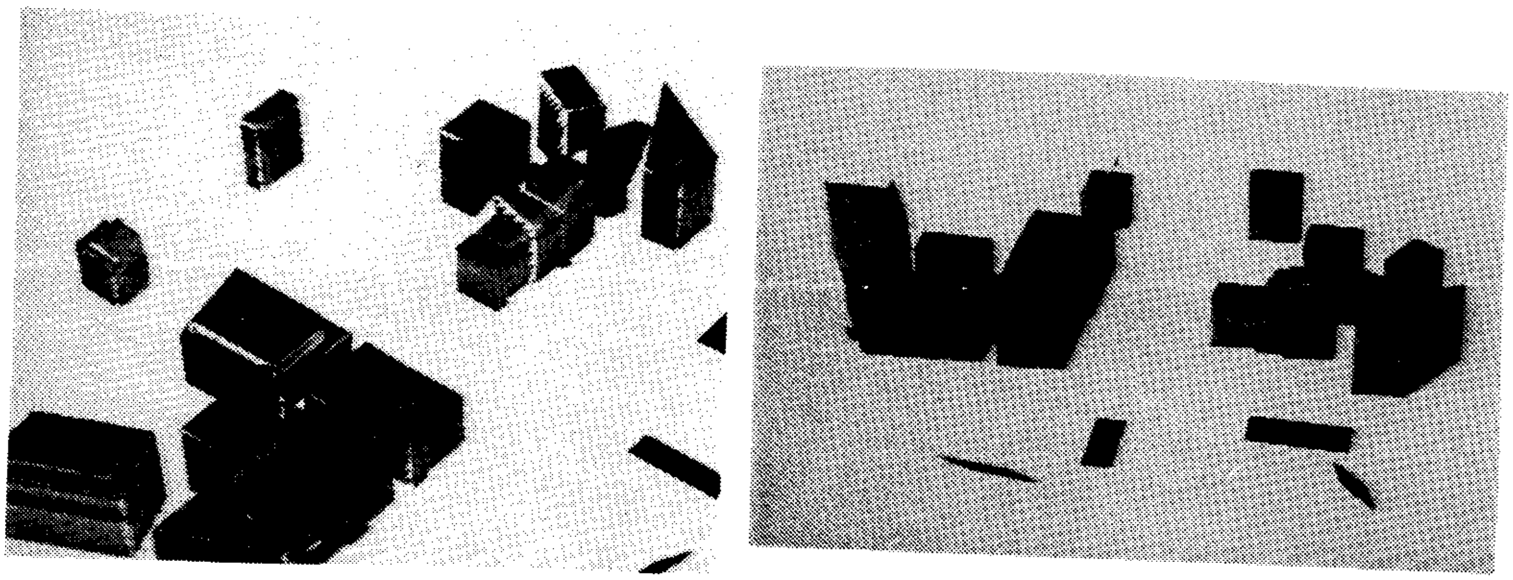


Figure 12: Reconstructed buildings of Fig. 11 with gray scale mapped onto faces. Gray scale values were derived from the left image in Fig. 1. The red in the display indicates faces and portions of faces that are occluded in the original image.

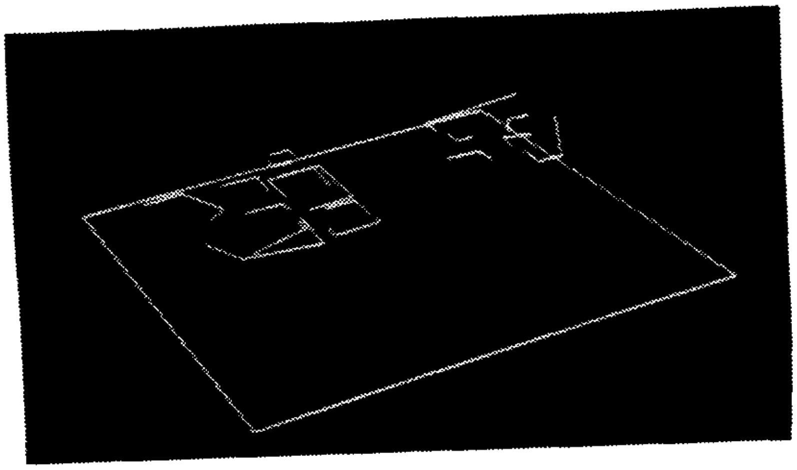


Figure 13: Perspective view of manually generated vertices and edges which simulates the information derived from stereo analysis of images obtained from an opposite point of view from that shown in Fig. 1. The viewpoint for this drawing is similar to Fig. 10, to allow easier comparison by the reader.

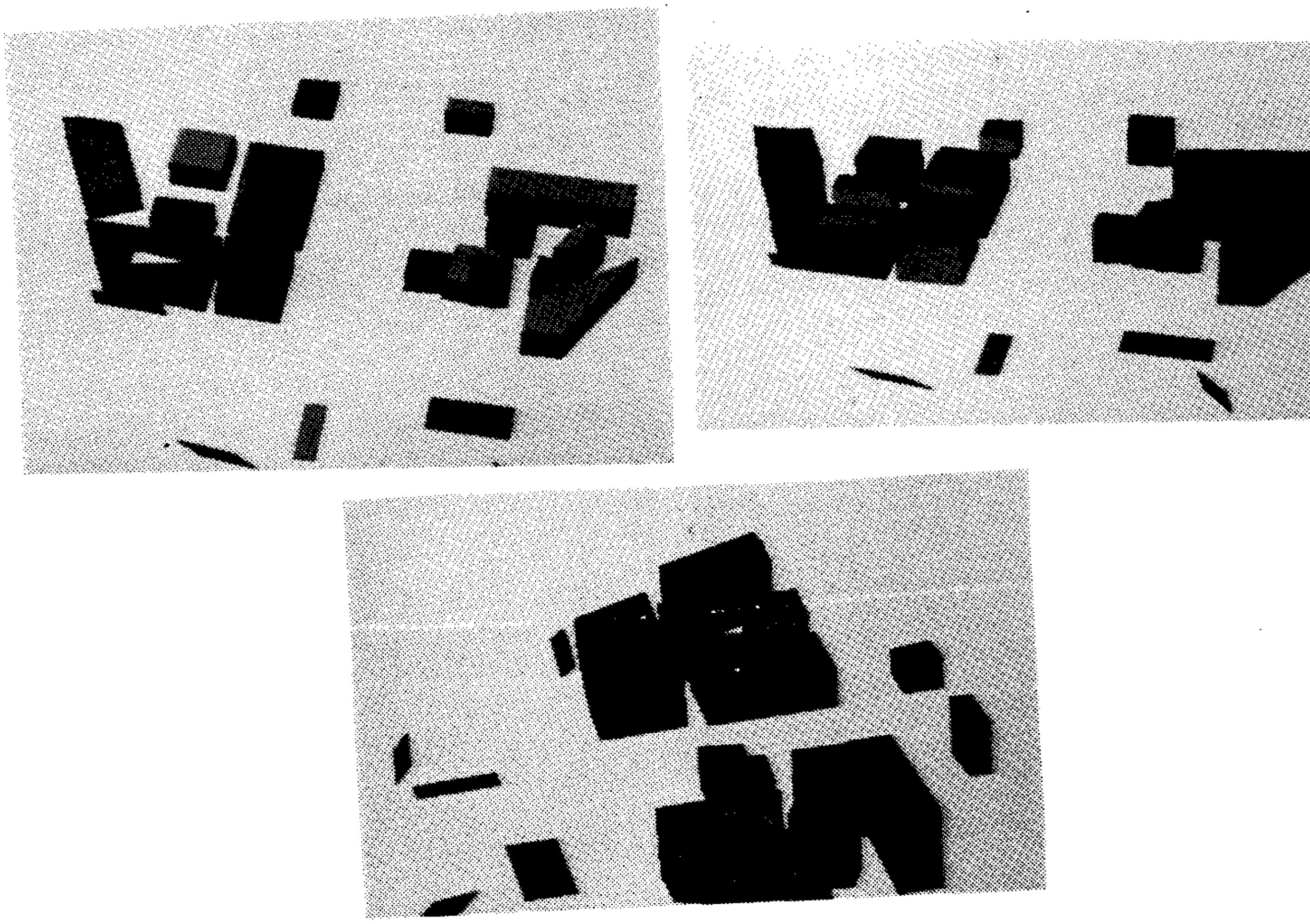


Figure 14: Perspective views of buildings reconstructed from two views.

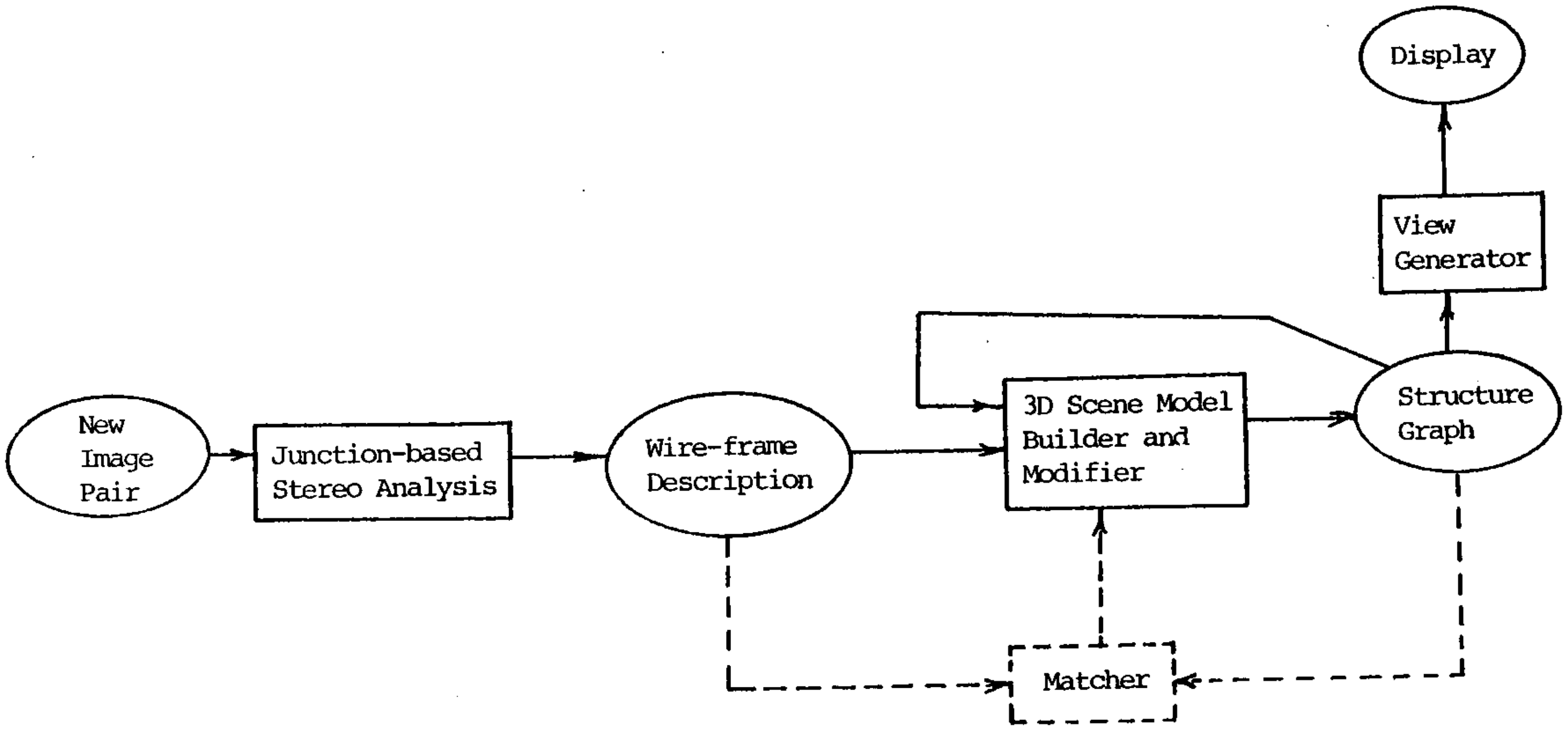


Figure 15: **3D Mosaic** flowchart, showing major modules (boxes) and data structures (ellipses). The matcher has not yet been implemented.