

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

CMU-CS-84-102

University Libraries
Carnegie Mellon University
Pittsburgh PA 15213-3890

**The 3D MOSAIC Scene Understanding System:
Incremental Reconstruction of 3D Scenes
from Complex Images**

Martin Herman

Takeo Kanade

February 1984

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.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US government.

Abstract

The 3D Mosaic system is a vision system that incrementally reconstructs complex 3D scenes from multiple images. The system encompasses several levels of the vision process, starting with images and ending with symbolic scene descriptions. This paper describes the various components of the system, including stereo analysis, monocular analysis, and constructing and modifying the scene model. In addition, the representation of the scene model is described. This model is intended for tasks such as matching, display generation, planning paths through the scene, and making other decisions about the scene environment. Examples showing how the system is used to interpret complex aerial photographs of urban scenes are presented.

Each view of the scene, which may be either a single image or a stereo pair, undergoes analysis which results in a 3D wire-frame description that represents portions of edges and vertices of objects. The model is a surface-based description constructed from the wire frames. With each successive view, the model is incrementally updated and gradually becomes more accurate and complete. Task-specific knowledge, involving block-shaped objects in an urban scene, is used to extract the wire frames and construct and update the model.

1. Introduction

It is important for a general vision system to derive three-dimensional (3D) information about a given scene from images and store the information in a coherent manner so that it can be used for various matching, planning, and display tasks. The **3D Mosaic** system is a vision system that incrementally acquires a 3D description (or model) of a complex scene from multiple images. This paper describes the system and presents examples of how it is used to interpret complex aerial photographs of urban scenes.

The paper is organized as follows. First, we present the motivation for our approach of incrementally acquiring the scene model, together with an overview of the system. Then we discuss the two components used to extract 3D information from the images: the stereo analysis and monocular analysis components. Next we describe the representation of the scene model, and an example is presented that shows how the scene model is acquired. Finally, we show how information from a new view is incrementally combined with a current model.

2. The 3D MOSAIC System

The goal of the **3D Mosaic** system is to obtain an understanding of the 3D configuration of surfaces and objects in a scene. The significance of this goal may be demonstrated by the following tasks.

1. **Model-based image interpretation.** A known 3D scene model can provide significant aid in interpreting arbitrary images of the scene [Barrow, Bolles, et al. 77, McKeown 83, Rubin 80]. The **3D Mosaic** system performs the task of acquiring such a model of the scene.
2. **3D change detection.** Change detection is a task that determines how the geometry and structure of a scene changes over time. The conventional approach to this task involves comparing and detecting changes in images. However, because of different viewpoints and lighting conditions, changes in the images do not necessarily correspond to changes in the geometry and structure of the scene. If 3D scene descriptions were obtained from the images first, such descriptions could be compared in 3D to determine changes in the scene.
3. **Simulating the appearance of the scene.** If a 3D description of the scene were to be obtained, displays as seen from arbitrary viewpoints could be generated from it. This is useful for tasks such as familiarizing personnel with a given area, and flight planning by generating the scene appearance along hypothetical flight paths.
4. **Robot navigation.** Three-dimensional descriptions of complex environments may be used to make decisions dealing with path planning or determining which parts of the environment to analyze in more detail.

Note that to perform these tasks, a vision system must do more than classify images, segment them, or identify objects in them; it must be able to generate a 3D description of the scene.

The **3D Mosaic** system deals with complex, real-world scenes (e.g., Fig. 4). That is, the scenes contain

many objects with a variety of shapes, the object surfaces have a variety of textures and reflectance characteristics, and the scenes are imaged under outdoor lighting conditions. Because of the complexity, there are many difficulties in interpreting the images, including:

1. Any particular image contains only partial information about the scene because many surfaces are occluded.
2. Even portions of the scene that are visible are often difficult to recover. For example, surfaces with dark shadows cast across them, or with highlights, may be difficult to interpret. Highly oblique surfaces may be difficult to analyze if their resolution in the image is poor. Such portions of the scene, therefore, may be recovered with errors and inconsistencies, or may not be recovered at all.

Our approach to the problems of complexity is to use multiple images obtained from multiple viewpoints. This approach aids interpretation in two ways. First, surfaces occluded in one image may become visible in another. Second, features of surfaces that are difficult to analyze and interpret in one image (such as scene edges and texture) may become more apparent in another image because of different viewpoint and/or lighting conditions.

2.1. Incremental Approach

A large number of views will, in general, be required to obtain a fully accurate and complete description of a complex scene. Typically, all these views will not be simultaneously available, while some may never become available. Many of them will only be obtained gradually through interaction with the scene environment. Our system must therefore have the ability to utilize partial descriptions and incrementally update them with new information whenever a new view happens to become available. As a practical example, consider a robot (perhaps a mobile ground robot or an automatically guided airplane) which is attempting to navigate through an unknown environment. The robot would sequentially acquire images of the environment as it moves about. Information derived from each new image would serve to update its internal model, and this partial model would be used to decide where to go next, or where to analyze in more detail.

We have adopted an approach in which the 3D scene model is incrementally acquired over the multiple views. The views of the scene are sequentially acquired and processed. Partial 3D information is derived from each view. The initial model is constructed from 3D information obtained from the first view, and represents an initial approximation of the scene. As each successive view is processed, the model is incrementally updated and gradually becomes more accurate and complete.

In our approach, the scene model plays the role of a central representation with two primary functions. First, it incrementally accumulates information about the scene. Second, at any point along its development,

it represents the current understanding of the scene. As such, it may be used for tasks such as matching, display generation, planning paths through the scene, and making other decisions about the scene environment. Two such tasks are important for the incremental acquisition process itself: (1) 3D information derived from a new view must be matched to the model so that updating can occur, (2) higher-level components should be able to use the model to determine which parts of the scene to analyze in more detail, and from which viewpoints to take the next images.

Most previous research efforts at acquiring 3D scene descriptions from multiple views have dealt with relatively simple scenes in controlled environments [Baker 77, Baumgart 74, Bourne, Milligan, and Wright 82, Martin and Aggarwal 83, Potmesil 83, Underwood 75]. This has led, in some cases, to only utilizing occluding contours in the image to form the 3D description [Baker 77, Baumgart 74, Bourne, Milligan, and Wright 82, Martin and Aggarwal 83]. The work of Moravec [Moravec 80] deals with complex indoor and outdoor scenes, but the 3D descriptions generated by his system consist of sparse sets of feature points. Our system, on the other hand, generates full, surface-based descriptions.

2.2. Overview

A flowchart for the 3D Mosaic system, showing the major modules and data structures, is displayed in Fig. 1. The input is a new view of the scene, which may be either a stereo image pair or a single image. The stereo pair undergoes stereo analysis, while the single image undergoes monocular analysis. The purpose of these analyses is to obtain 3D scene features such as portions of surfaces, edges, and corners. The stereo analysis component currently matches junctions extracted from the two images, and generates a sparse 3D wire-frame description of the scene. The monocular analysis component currently extracts linear structures from the image and converts these to 3D wire frames using task-specific assumptions.

The central scene model is a surface-based description which is constructed and modified from these features. It is represented as a graph in terms of primitives such as faces, edges, vertices, and their topology and geometry. It also has mechanisms to add and delete hypotheses for parts of the scene for which there are partial data. Before modifications to the scene model can occur, the 3D features from the new view must be matched to the current model. The scene model may, at any point along its development, be used for tasks such as image interpretation, planning, or display generation. A new view may then be acquired which may further modify the model.

For example, when the stereo analysis component is applied to the images in Fig. 4, the result is the set of wire frames in Fig. 29. The scene model constructed from these wire frames is shown in Fig. 31. When the monocular analysis component is applied to the image in Fig. 14, the result is the set of wire frames in Fig. 25. These, in turn, are converted into the scene model in Fig. 32. Finally, the result of modifying the model in

Fig. 31 with a new view is shown in Fig. 38.

3. Stereo Analysis

Most stereo matching methods involve matching low-level image features, such as image intensities [Baker and Binford 81, Hannah 74, Lucas and Kanade 81, Ohta and Kanade 83] or image edge points [Baker and Binford 81, Grimson 80, Ohta and Kanade 83]. Points to be matched may also be chosen as "interesting points", e.g., those with high variance in all directions [Barnard and Thompson 80, Moravec 80]. Our method involves matching structural features -- i.e., junctions -- extracted from the images. There are several reasons for this.

First, feature-based matching results in more accurate 3D positions for occlusion boundaries than gray scale area matching. Second, by extracting 3D information dealing with scene vertices and edges emanating from them, we obtain portions of boundaries of scene buildings, particularly building corners. These boundaries are then used to construct 3D approximations of the buildings.¹

Finally, because of our wide-angle stereo images, there are large disparity jumps and large portions of the scene are visible in one image but not the other. Because most stereo systems do not distinguish these from other regions of the image, they try to find matches for them and therefore have trouble [Baker and Binford 81, Barnard and Fischler 82, Barnard and Thompson 80, Grimson 80, Hannah 74, Lucas and Kanade 81].

In our approach, rather than attempting to find matches for scene faces occluded in one of the images, we match face boundaries visible in both images. We do this by explicitly taking into account the way junction appearances change from one image to the other, using the knowledge that in urban scenes, roofs of buildings tend to be parallel to the ground plane, while walls tend to be perpendicular to this plane. Edges in the scene perpendicular to the ground will appear in each image to be directed towards the vertical vanishing point [Kender 83].

If a feature in an image lies on a roof, its appearance in the other image as a function of position along the epipolar line can be predicted if the normal to the ground plane is known.² To see why, consider Fig. 2. Suppose the junction $P_3P_1P_2$ in image1 is given, and our goal is to predict the junction $Q_3Q_1Q_2$ in image2, where the point Q_1 lies anywhere (inside the infinity point) on the epipolar line corresponding to P_1 . For the position Q_1 , the 3-space position of V_1 can be computed as the intersection of the rays through P_1 and Q_1 .

¹For a different approach developed for the same domain, see [Henderson, Miller, and Grosch 79].

²In stereo images, it is known that for each point in one image, the corresponding point in the other image lies along a line, called the epipolar line, which depends only on the camera model [Barnard and Fischler 82].

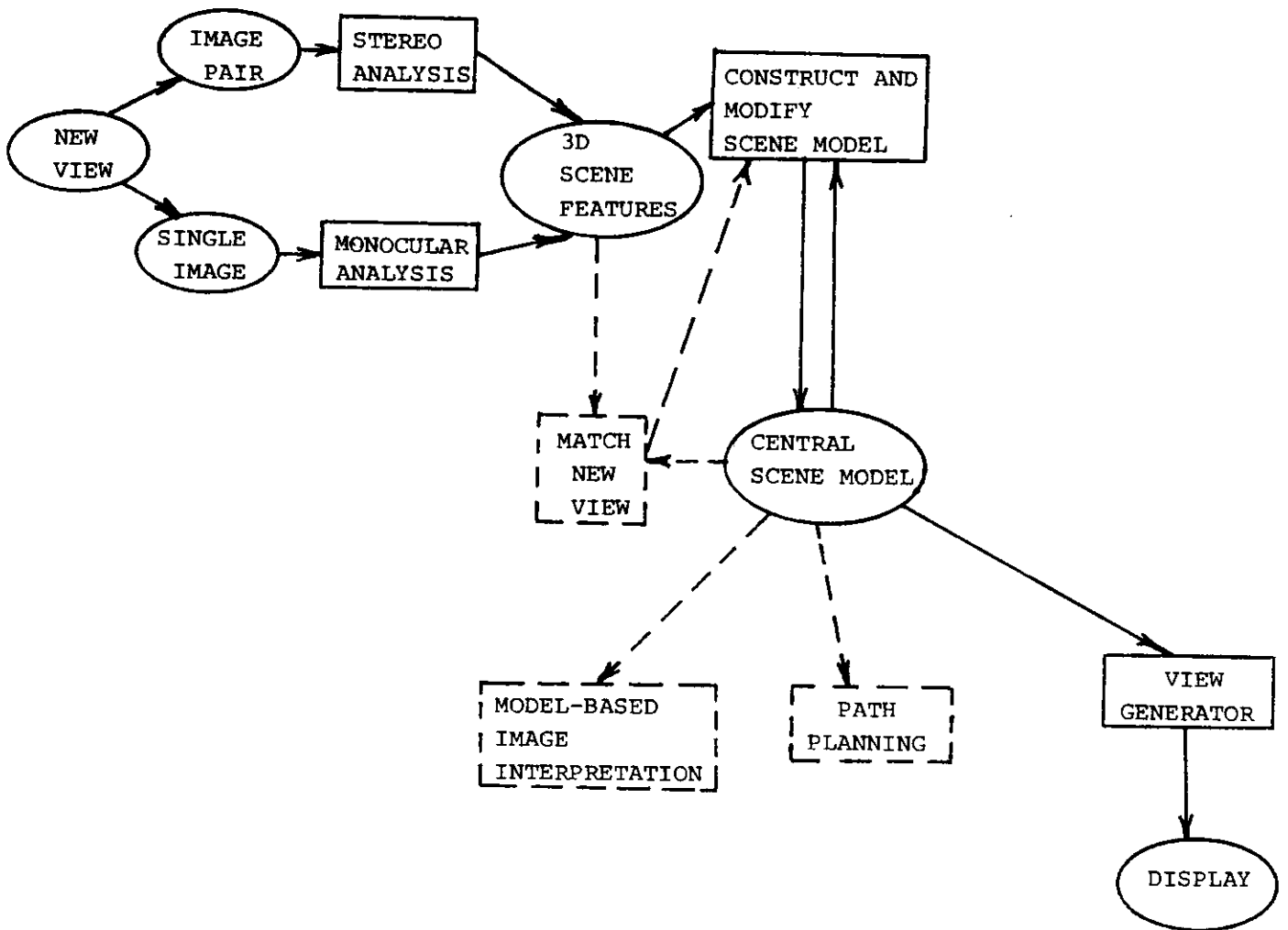


Figure 1: 3D Mosaic flowchart, showing major modules (boxes) and data structures (ellipses). The dashed lines represent components that have not yet been implemented; the solid lines represent components already implemented.

This uniquely determines the position of the plane parallel to the ground that contains V_1 . The 3-space positions of the points V_2 and V_3 can now be computed as the intersections of this plane with the rays corresponding to the points P_2 and P_3 , respectively. Finally, the points Q_2 and Q_3 are uniquely determined as central projections of the points V_2 and V_3 , respectively.³ Although this analysis is independent of the camera geometry relative to the scene, vertical aerial photography is in general more useful than oblique aerial photography because of the greater probability that an arbitrary junction in the image lies on a roof or on the ground. In oblique aerial photography, larger portions of horizontal surfaces would be occluded by vertical walls.

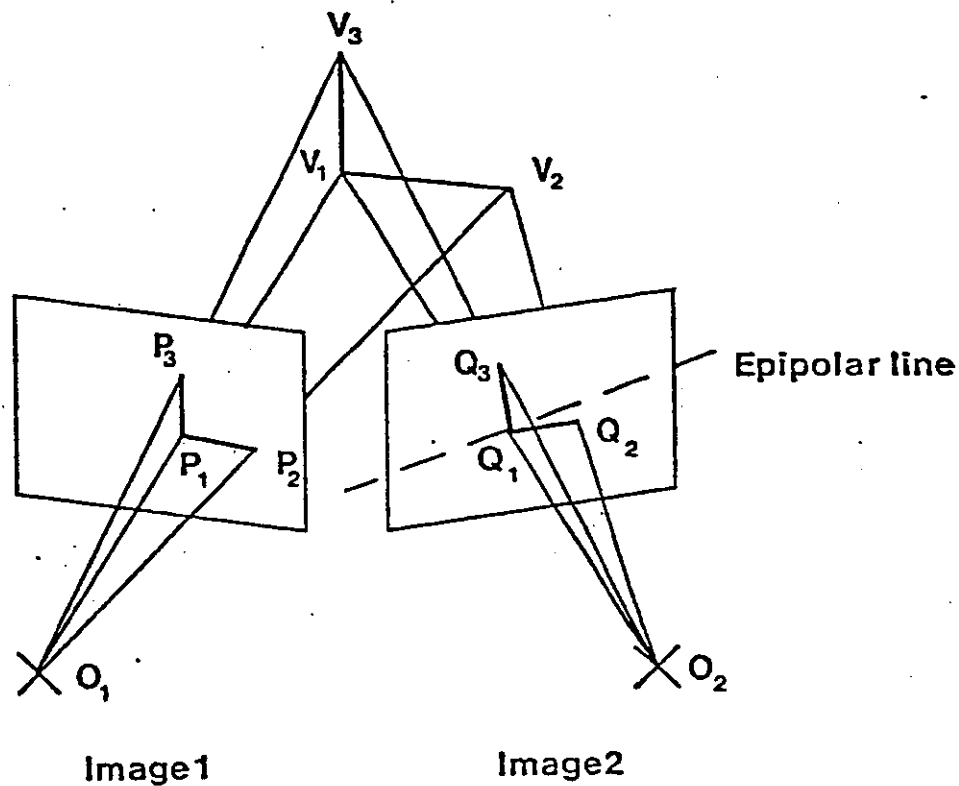


Figure 2: For junction $P_3P_1P_2$, its appearance in image2 can be predicted as a function of position Q_1 along the epipolar line. The normal to plane $V_3V_1V_2$ must be known.

Therefore, when an I. junction is found in one image, it is initially assumed to arise from a corner of a roof, and its appearance in the other image can be predicted. When an ARROW or FORK junction is found, the leg of the junction directed towards the vertical vanishing point is initially assumed to arise from a scene edge perpendicular to the ground, while the other two legs are initially assumed to arise from scene edges lying on a roof or on the ground. Again its appearance can be predicted.

³Note that this analysis is valid not only for features lying on horizontal planes in the scene, but for any family of parallel planes.

Structural relationships between scene vertices are also used to aid in the matching. If two junctions in an image arise from scene vertices at the same height above the ground, the positions of the corresponding junctions in the other image, as a function of position along the epipolar line, can be predicted if the normal to the ground plane is known. This can be shown using similar arguments as before. In Fig. 2, pretend that the points P_1 , Q_1 , and V_1 correspond to positions of separate junctions and vertices. For example, if P_1 and P_3 are two separate junctions in image 1, then for some point Q_1 on the epipolar line corresponding to P_1 , the position of the junction Q_3 , corresponding to P_3 , can be predicted if V_1 and V_3 are assumed to lie at the same height. We make the assumption that junctions close to one another in the image often correspond to vertices lying on top of the same building and therefore have approximately the same height. In this way, the configurations within the neighborhoods around junctions in the two images are used in the matching.

These matching techniques assume that the vector normal to the ground plane is known. To obtain this vector, we form a vector from the focal point to the vertical vanishing point. As shown in Fig. 3, this results in a 3-space vector in the vertical direction [Barnard 82]. The vertical vanishing point is the central projection of the "infinite" point of any vertical line. In other words, a line containing the focal point and vertical vanishing point intersects any vertical line at "infinity." Therefore they must be parallel.⁴ The focal length and vertical vanishing point are currently manually obtained.

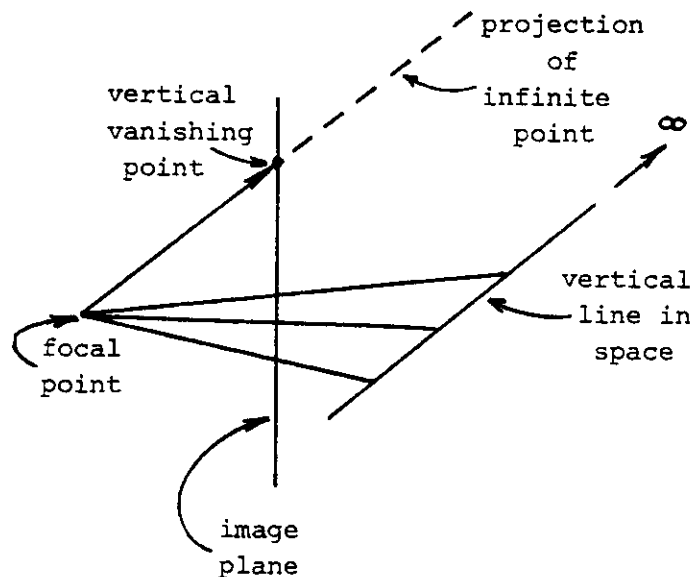


Figure 3: The vector from the focal point to the vertical vanishing point is a 3-space vector in the vertical direction.

⁴This analysis, of course, holds for all vanishing points, not only the vertical one.

3.1. Steps in Stereo Analysis

We now provide an example showing how the stereo analysis is performed on the stereo pair of images in Fig. 4.

Extracting 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. 5. Then the edges are thinned using a modified Nevatia and Babu algorithm [Nevatia and Babu 80], as shown in Fig. 6. The resulting edge points are linked and approximated by piecewise linear segments. The method used to fit linear segments to a set of linked points is based on iterative end-point fitting [Duda and Hart 73]. 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. 7.

Extracting junctions. The next step is to extract junctions from the line images. A junction is a group of line segments (*legs*) in the image 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 the window contains the ends of two lines, they are classified as an L junction: the intersection of the two lines determines the position of the junction. If the 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. 8. Notice that many of the junctions correspond to building corners.

Finding potential junction matches. We now want to match the junctions found in one image with those in the other. Let us consider how L junctions are matched. Each L junction is initially assumed to lie on a horizontal scene plane. The shape and orientation of its corresponding junction in the other image, as a function of position along the epipolar line, can therefore be predicted. Each L junction in the first image may therefore usually be matched with several junctions in the second image that have, within tolerance, the predicted shape and orientation. However, we do not try to match only with junctions in the second image that have been previously found. Rather, for every point on the epipolar line (on the appropriate side of the infinity point), 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 form 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. The matching is performed in two directions, from the first image to the second, and vice versa.

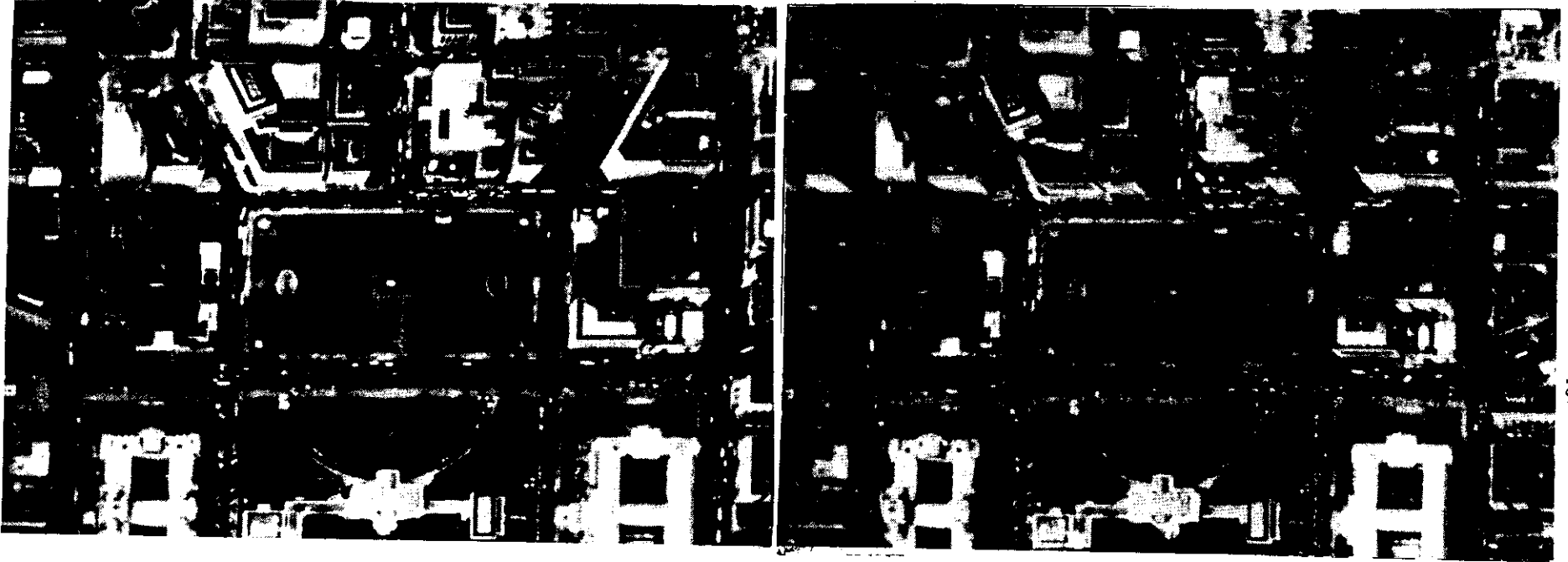


Figure 4: Gray scale stereo images of a region of Washington, D. C.

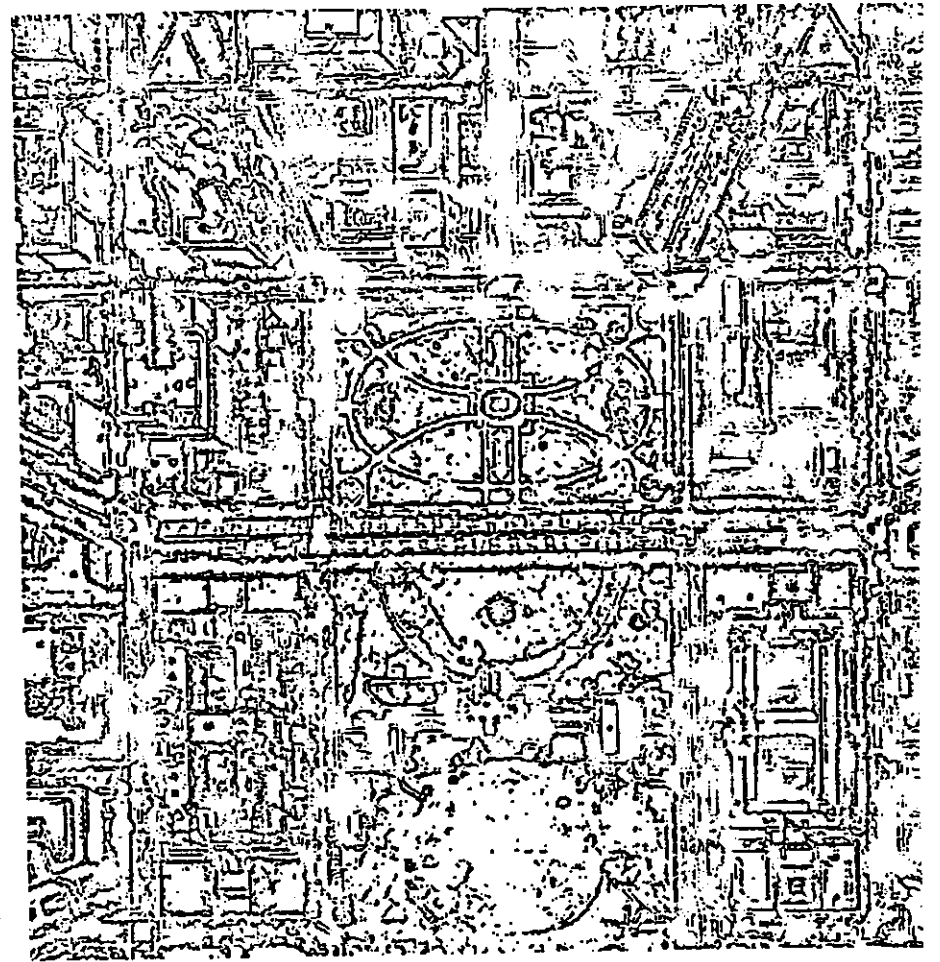
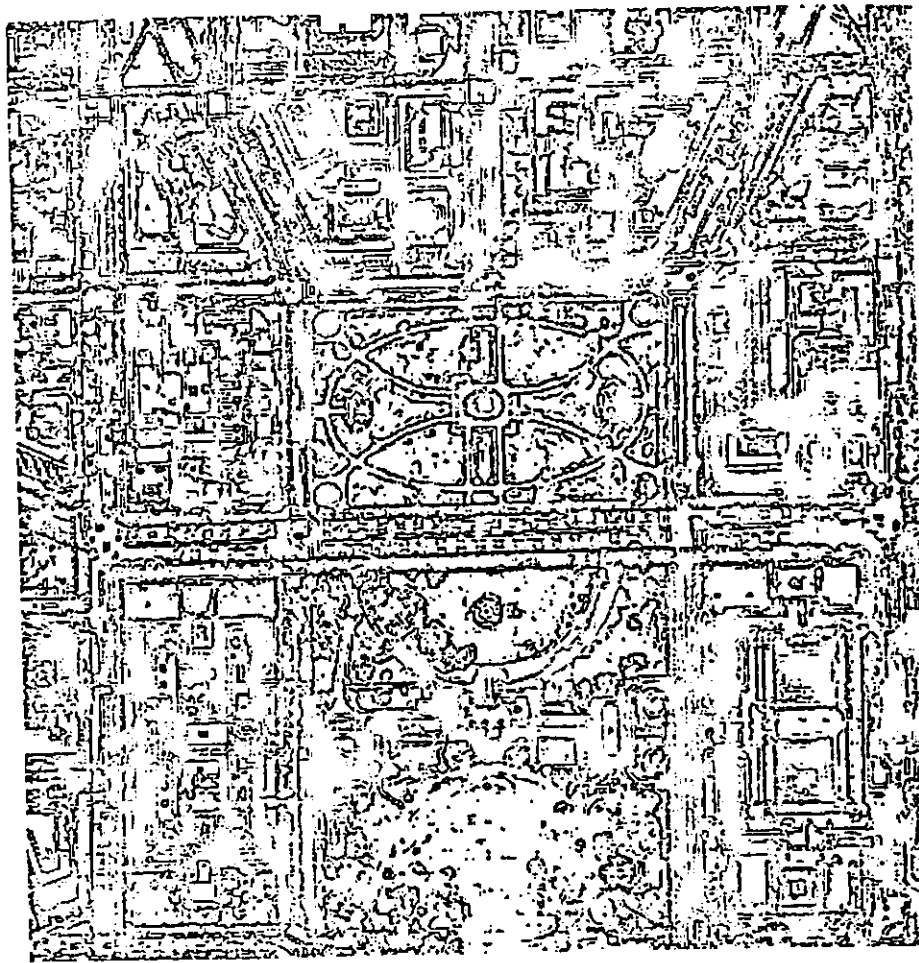


Figure 5: Edges resulting from a Sobel operator applied to the images in Fig. 4.

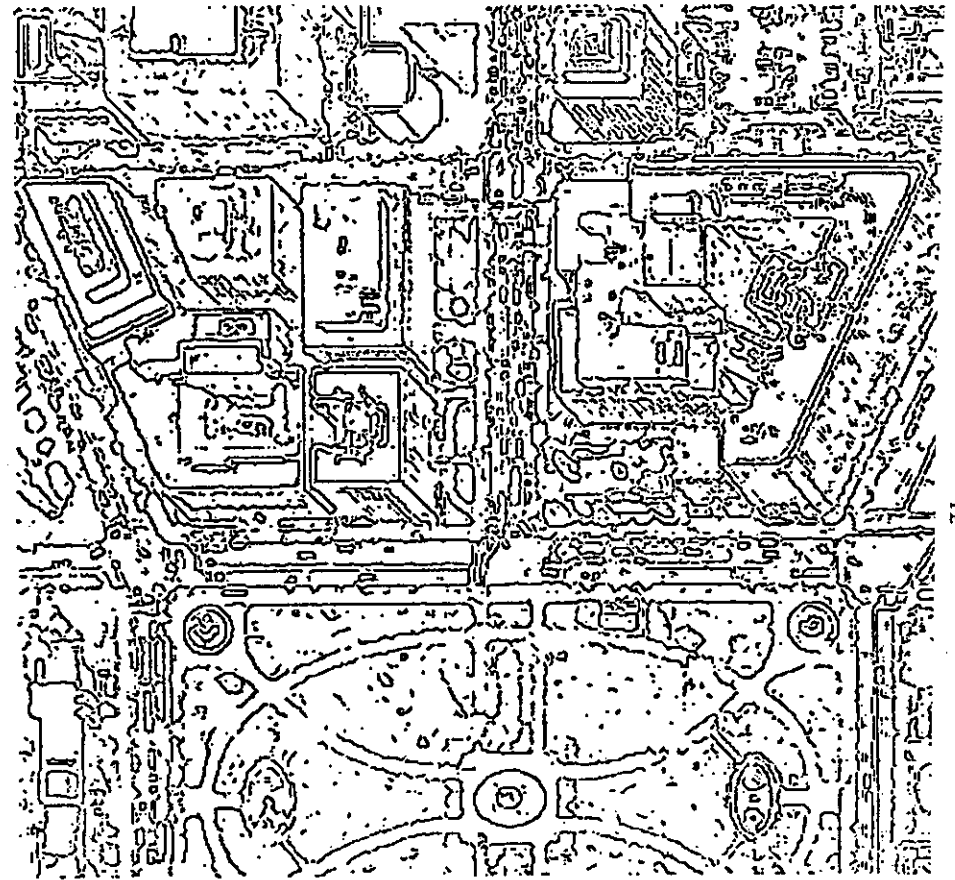
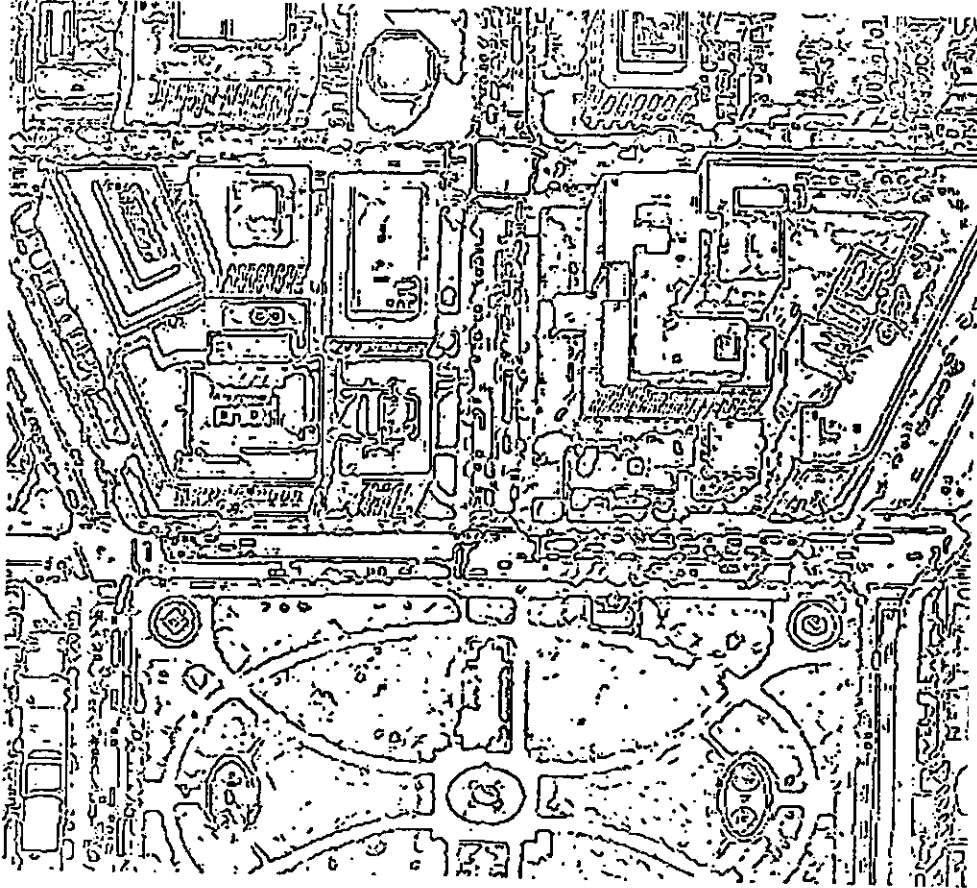


Figure 6: Result of thinning the edges in Fig. 5. Results for the upper middle part of each image in Fig. 5 are shown here.

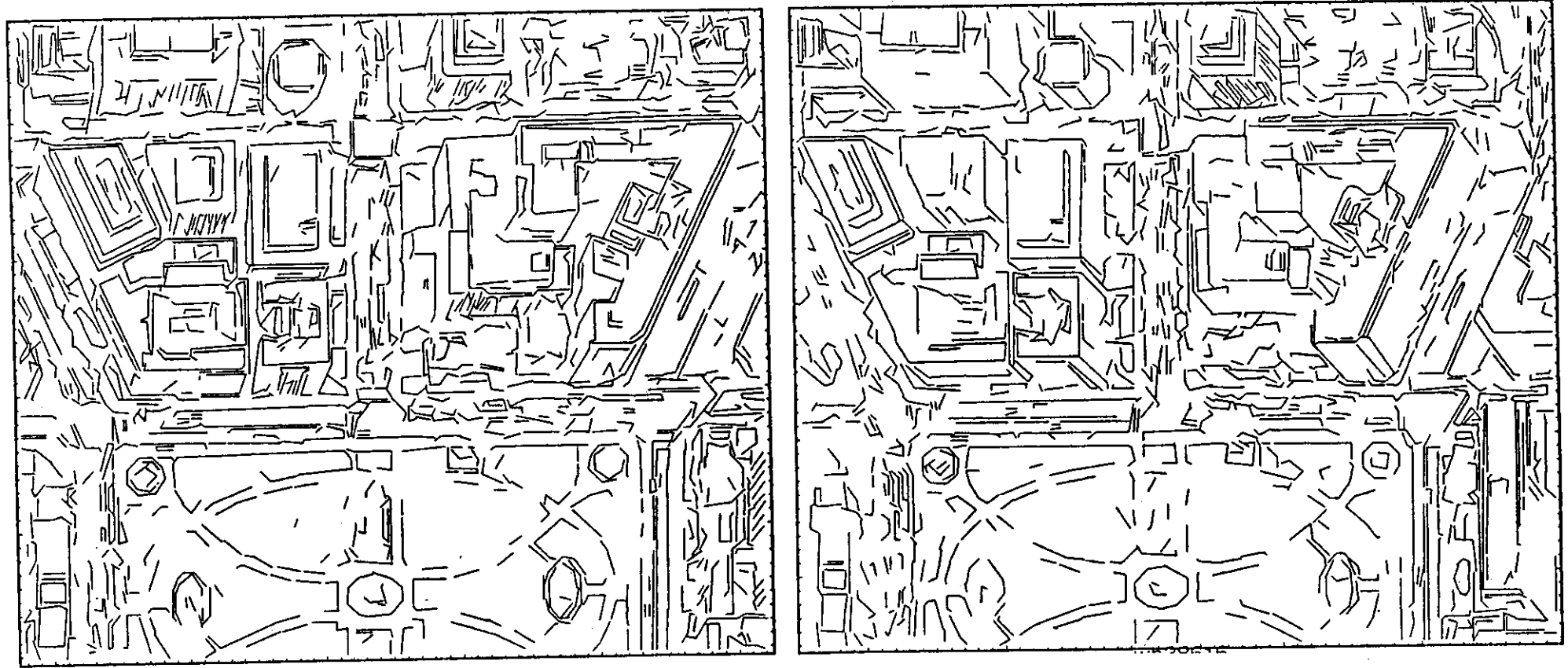


Figure 7: Linear segments fitted to the edge points of Fig. 6 after they are linked.

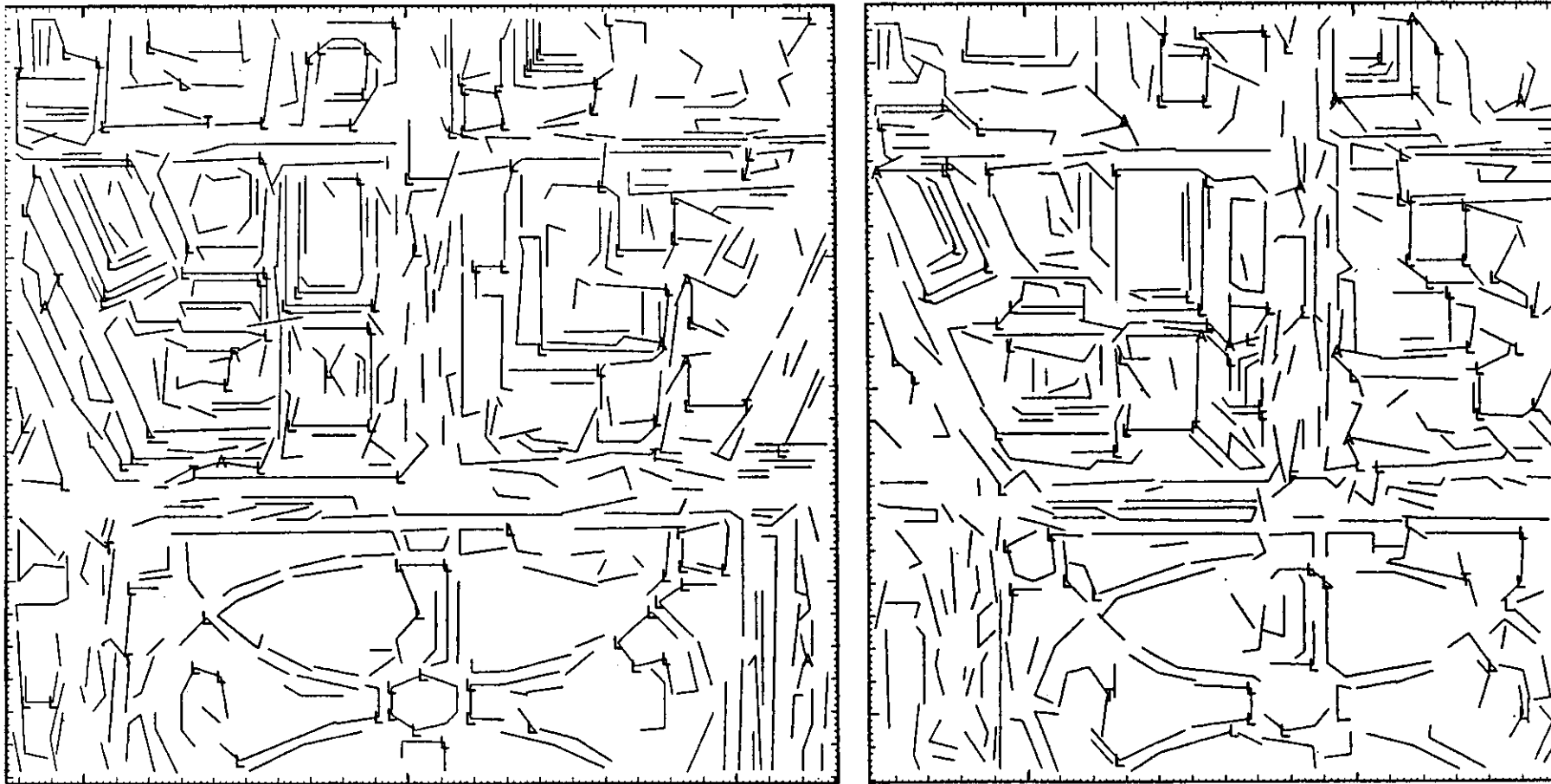


Figure 8: Result of classifying junctions in a different version of line images than shown in Fig. 7. Junctions are classified as L, Λ (arrow), F (fork), or T.

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.

Searching for unique junction matches. At this point, each junction in one image is associated with a set of potentially matching junctions in the other image. The next step is to find the best of the potential matches, resulting in a single match for each junction. Two criteria are used in determining the best matches:

1. If the image intensities inside two potentially matching junctions are similar, the likelihood that they really match is increased. This is because the two junctions will often have similar intensities if they arise from the same face corner. To measure the degree of similarity, we compute the average intensities of regions along the two legs of the L junction in each image. As depicted in Fig. 9, let A and B be the average intensities of these regions in one image, and let A' and B' be the average intensities of corresponding regions in the other image. Then the degree of similarity, called the *local cost*, is defined as

$$C_{\text{local}} = |A - A'| + |B - B'|.$$

Similar intensities in the two junctions result in small local cost, while diverse intensities result in large local cost.

2. As described previously, if two junctions in an image arise from scene vertices that are at the same height, the relative positions of the corresponding junctions in the other image, as a function of position along the epipolar line, can be predicted. We use this to determine whether two sets of junction matches are consistent with one another. Suppose, in Fig. 10, that the junctions J_1 and J_2 in image1 arise from scene vertices that are at the same height. Suppose also that the junction matches (J_1, J'_1) and (J_2, J'_2) have been hypothesized. To measure the degree of consistency between these two sets of matches, we predict the position of the junction in image2 that corresponds to (say) J_2 . Let us refer to the predicted position as J''_2 . If the vector from J'_1 to J''_2 is (a_1, b_1) and the vector from J'_1 to J'_2 is (a_2, b_2) , then the degree of consistency between the two sets of matches, called the *global cost*, is defined as

$$C_{\text{global}} = |a_1 - a_2| + |b_1 - b_2|.$$

Two sets of junction matches whose relative positions are near the prediction result in small global cost, while positions far from the prediction result in large global cost.

To arrive at a unique set of junction matches, the space of potential matches is searched using a beam search [Rubin 80], which is guided by the above two criteria. The search space is represented by a network whose nodes are the possible pairs of junction matches. This is depicted in Fig. 11, where each junction in (say) image1 (i.e., J, K, L, \dots) is paired with each of its potential matches in image2 (i.e., J', K', L', \dots). The junctions in image1 are ordered so that the junction in column k is within an $M \times M$ window of the junction in column $k - 1$. M is chosen so that there is a good probability that junctions within the window arise from vertices on top of the same building.

In Fig. 11, each junction and its candidates lie in a single column, and each candidate is represented by a node in the network. Any path through the network that visits a single node at each column represents a set of unique junction matches. Associated with each such path is a cost obtained by adding all the local costs of the

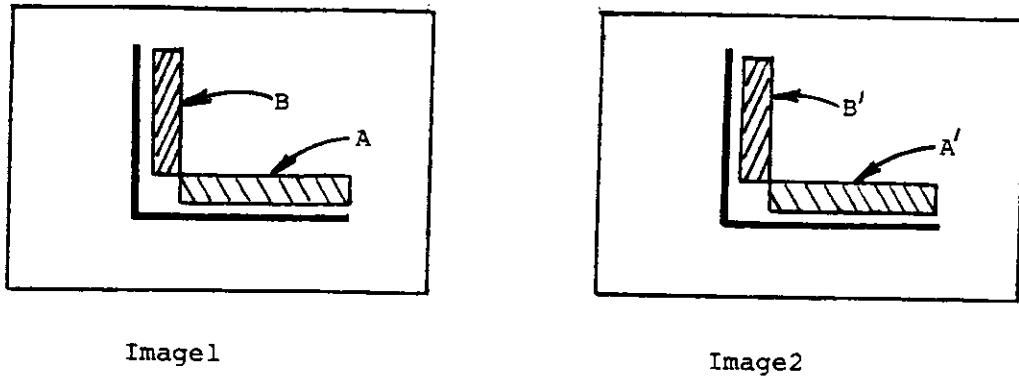


Figure 9: Intensities of corresponding regions of L-junctions in the two images are used to compute the local matching cost.

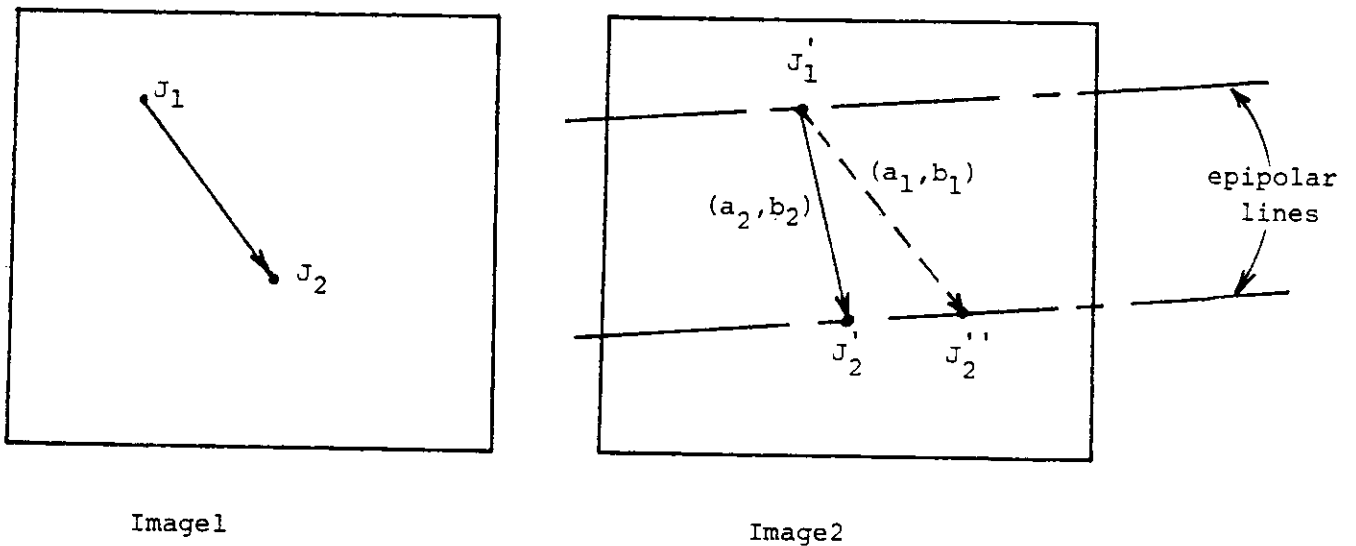


Figure 10: Positional vectors of predicted and actual positions of two junction matches are used to compute the global matching cost.

nodes visited by the path and all the global costs between each successive pair of nodes in the path. The goal of the search is to find the minimum cost path. With beam search, only a limited number of paths are explored.

The search starts at column 1 (Fig. 11) and proceeds successively to each column. At each column k , the best N partial paths from column 1 to k are extended to column $k + 1$ as follows. Suppose that each node in column k has a cost corresponding to the minimum cost path from column 1 to the node. Then for each of the N lowest cost nodes J'_i in column k , compute the cost of the path when extended from J'_i to each node K'_i in column $k + 1$. This cost is the sum of the cost of the partial path to node J'_i , the global cost between nodes J'_i and K'_i , and the local cost of node K'_i . Then add a link in the network between nodes J'_i and K'_i .

At the end of this set of steps, there will be a link from each of the best N nodes in column k to each node K'_i in column $k + 1$, and each node K'_i will now have several costs associated with it, one for each link into the node. Suppose the link from node J'_i has the lowest cost to K'_i . A backpointer from K'_i to J'_i is added, and the associated cost is stored. All other links and costs associated with node K'_i are discarded. Each of the best N nodes in column $k + 1$ are then extended to column $k + 2$. Notice that this search is not guaranteed to result in the lowest cost path in the network. A path discarded at column k because it is not among the best N may have been part of the best path at column $k + j$ if it were extended that far.

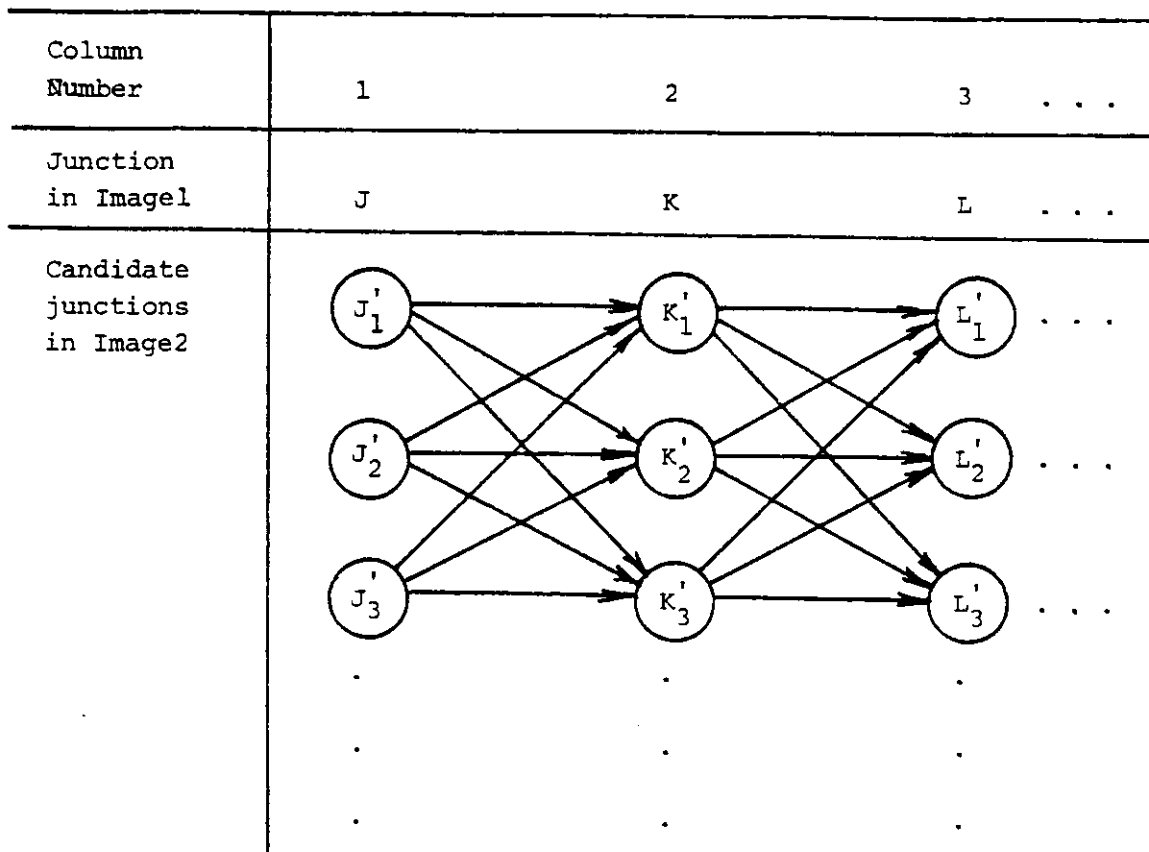


Figure 11: Each column contains a junction from image1 and its candidate matches from image2. The candidates form the nodes of a network which is searched by a beam search.

The matching procedure is applied from the first image to the second and vice versa. The results are then merged by retaining all pairs of junction matches except those in which one of the junctions appears in more than one junction pair. The results are displayed in Fig. 12, which shows junctions in one image that have matches in the other image.

Searching for third legs of junctions. The next step tries 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 finds lines near the junctions in both images that are directed toward the vertical vanishing point.

Generating 3D wire frames. Finally, 3D coordinates of vertices and equations of edges are derived using triangulation. Fig. 13 shows a perspective view of the 3D vertices and edges that result. We call this a wire-frame description of the scene.

4. Monocular Analysis

Although stereo is a major source of 3D information, some views of the scene will be only single images. We can also extract 3D information from these images by exploiting task-specific knowledge. We assume that the objects in the scene are trihedral polyhedra containing only vertical and horizontal faces, i.e., faces perpendicular and parallel, respectively, to the ground plane. Our monocular analysis extracts linear structures in the image that represent boundaries of buildings, and then converts these structures into 3D wire frames.

4.1. Steps in Monocular Analysis

This section provides an example showing how the monocular analysis is performed on the image in Fig. 14. This is a different view of the same scene shown in the earlier stereo pair (Fig. 4).

Extracting lines and junctions. The first step in the monocular analysis is to extract linear segments and junctions from the image. The method used here is the same as that used during stereo analysis (as previously described). The thinned edge points are shown in Fig. 15, and the result of extracting lines and junctions is shown in Fig. 16.

Locating 2D structures. Next we form linear connected structures in the image by hypothesizing new lines to connect the previously extracted junctions. These connected structures are meant to represent building boundaries and the hypothesized lines are meant to correspond to building edges. The process of hypothesizing connecting lines consists of two steps. First, two junctions may be connected only if a leg of one points at the other, that is, the extended leg meets the other junction. For each pair of junctions that passes this test, a line showing the connection between the two junctions is drawn in Fig. 17.

The second step involves determining which connections shown in Fig. 17 appear as connections in the line image (Fig. 16). For each pair of connected junctions J_i and J_k (Fig. 18), we find all segments in the line image that are contained within a thin rectangular window connecting J_i and J_k , and project these segments onto the line connecting the two junctions. Then we consider how much of this line is covered by projected

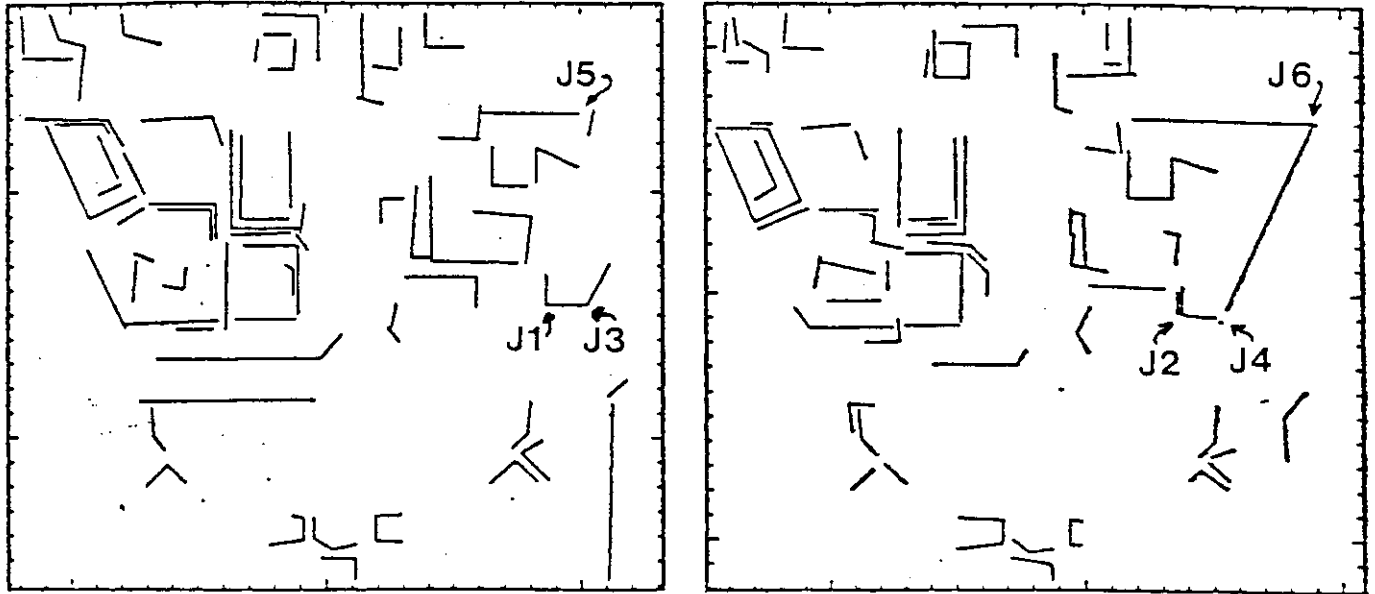


Figure 12: Matches that have been found for the junctions in Fig. 8. Actually, not all matches are correct. For example, although the junction matches (J1,J2) and (J3,J4) are correct, the match (J5,J6) is incorrect.

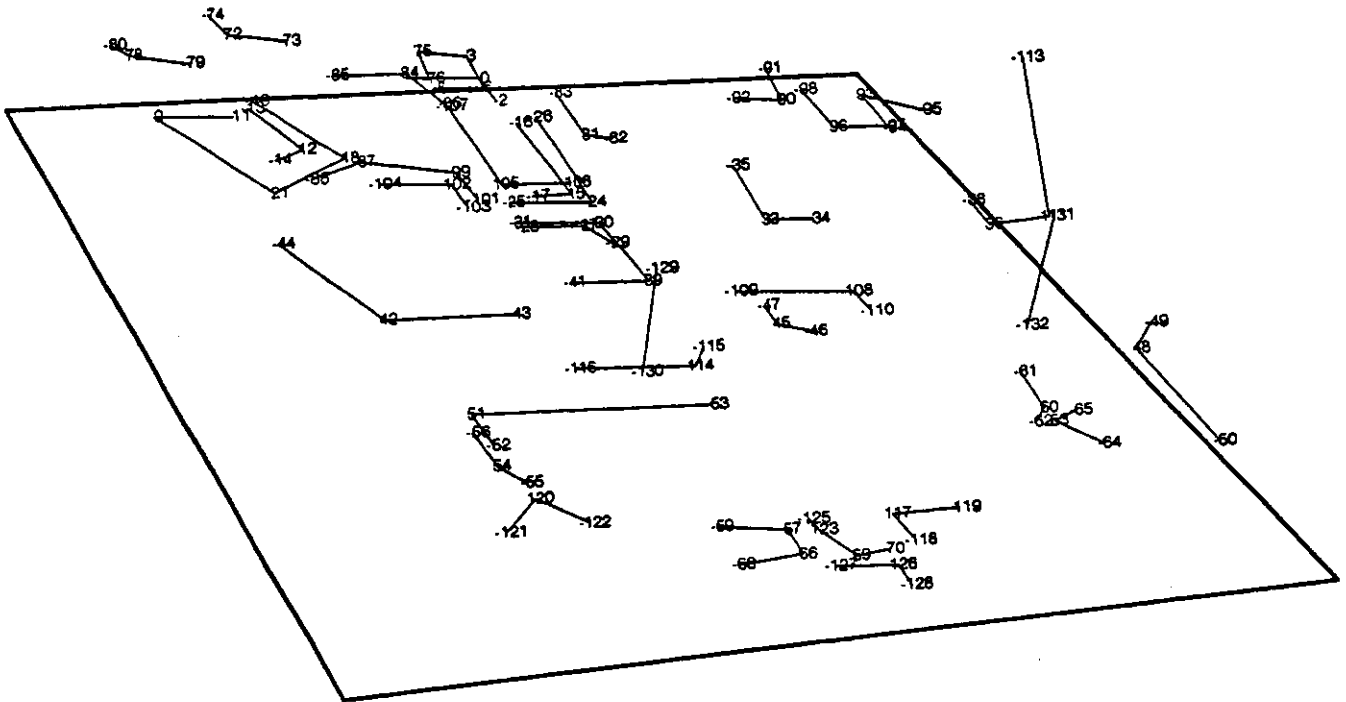


Figure 13: Perspective view of 3D wire-frame description (i.e., 3D vertices and edges) derived from matches shown in Fig. 12. The numbers represent unique identifiers for the end points of the edges.

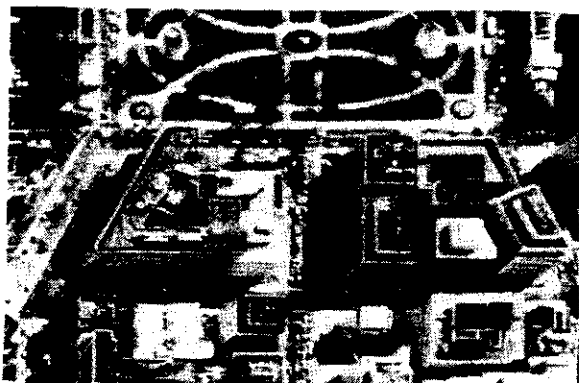


Figure 14: Aerial photograph showing part of Washington, D.C. This is a different view of the same scene as in Fig. 4.

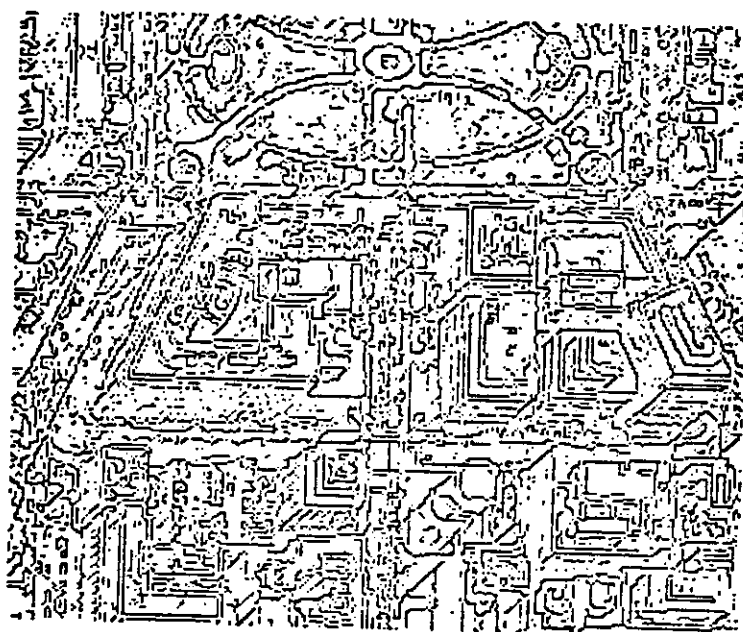


Figure 15: Result of thinning the edges obtained by applying a Sobel operator to the image of Fig. 14.

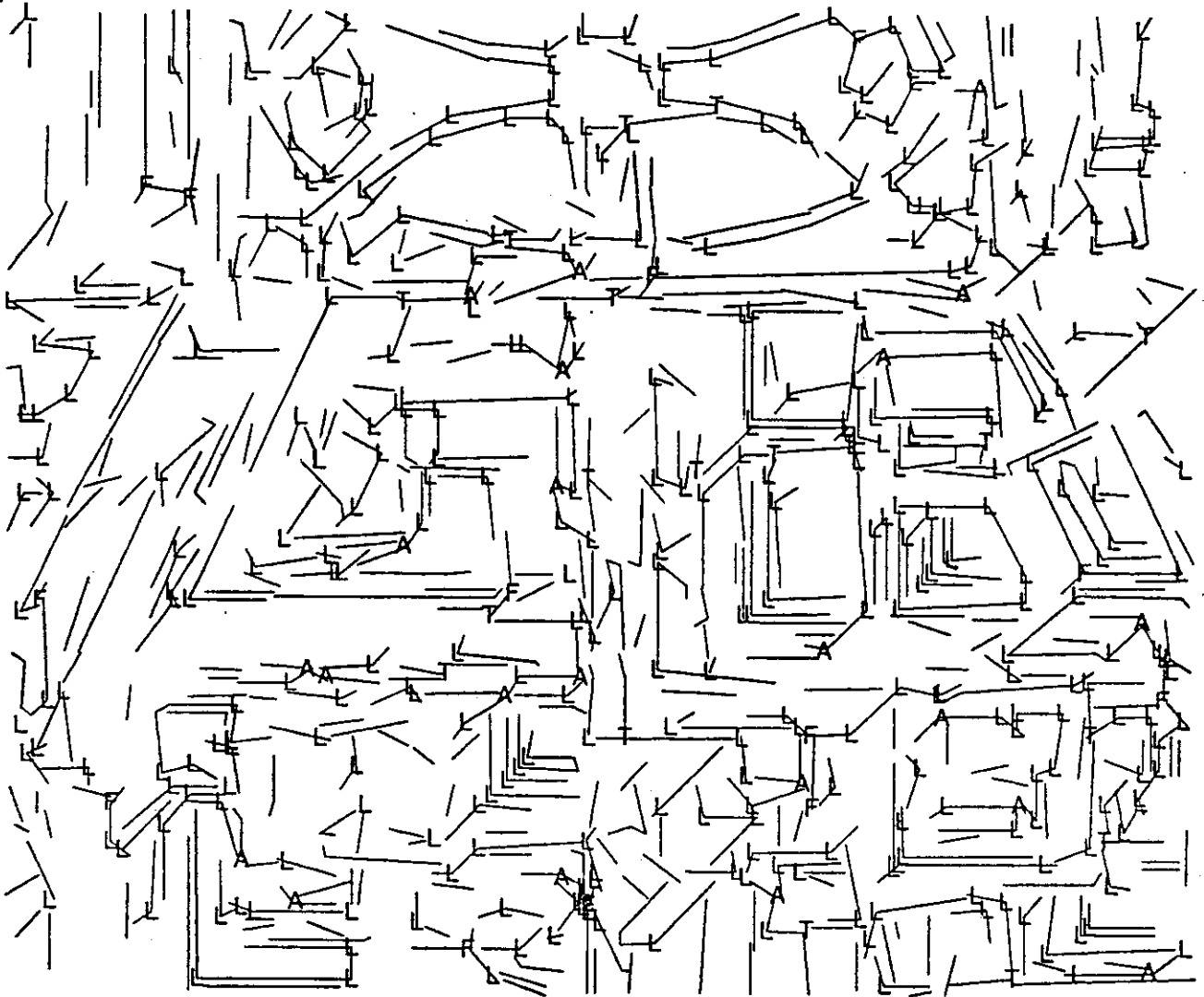


Figure 16: Lines fitted to the edge points of Fig. 15 after they are linked. Junctions in the image are classified as L, A (arrow), F (fork), or T.

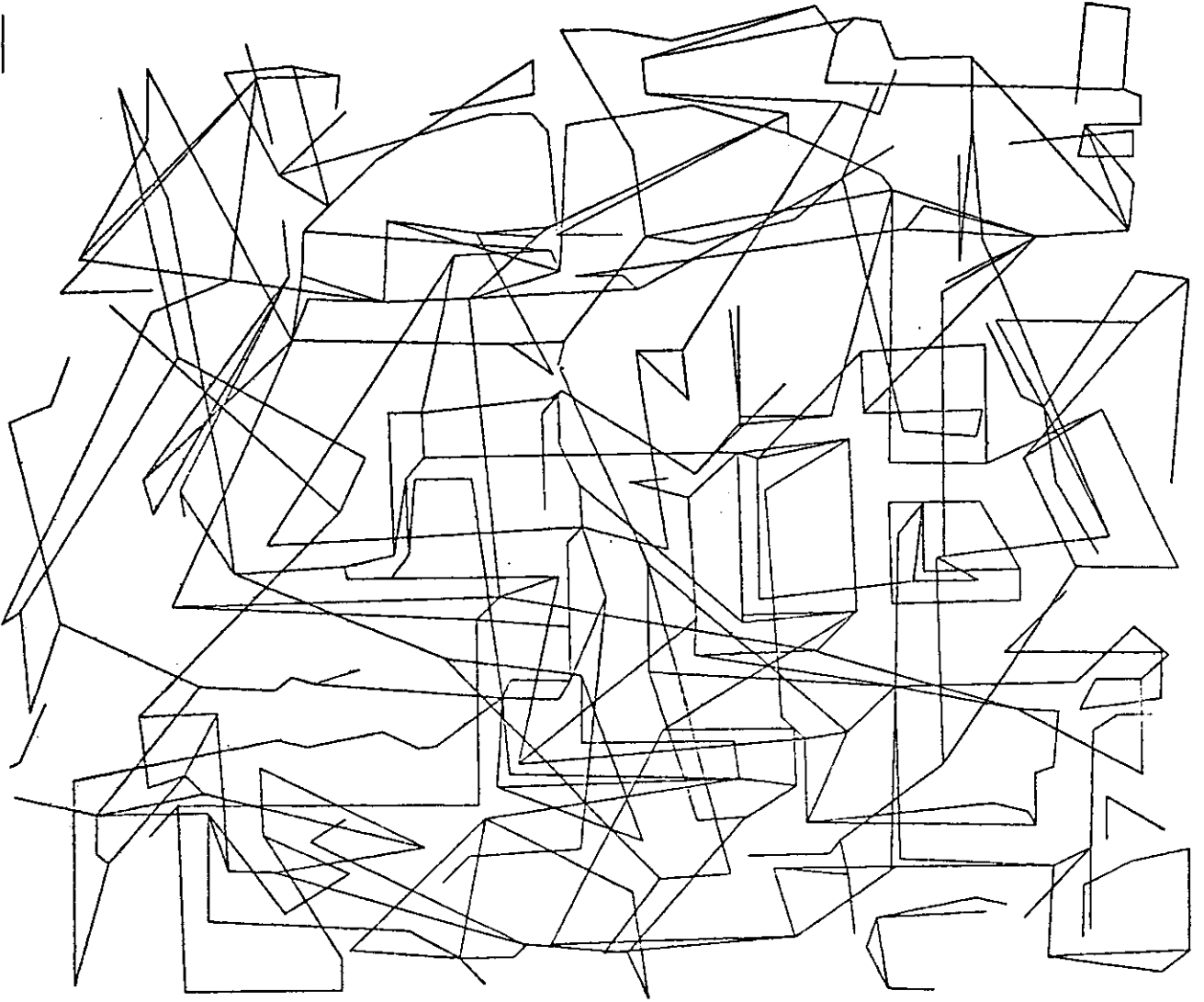


Figure 17: Each line represents a possible connection between the junctions at its two end points. Each end point corresponds to a junction in Fig. 16.

segments. The connection between J_i and J_k is retained only if the percentage of coverage exceeds a threshold. The result of this pruning step is shown in Fig. 19. Note that it does a good job in eliminating unwanted connections. These two steps illustrate how useful a hypothesize-and-test method can be for low-level image processing. In the first step, candidate connections are hypothesized on rather preliminary evidence. In the second step, the candidates that do not pass a rigid test are eliminated.

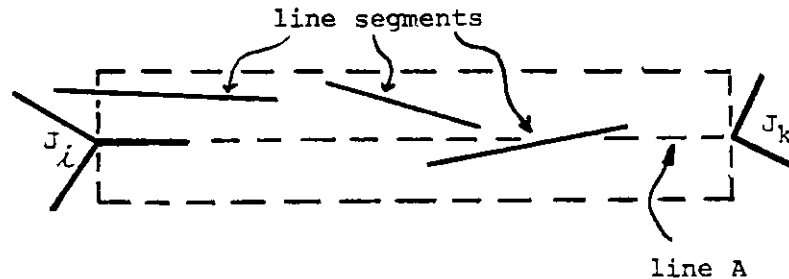


Figure 18: All line segments within the thin rectangular window connecting junctions J_i and J_k are projected onto line A to determine the amount of coverage.

The junction legs originally extracted in the junction finding step are then added to the results of Fig. 19, and extraneous legs are deleted. The final connected structures are displayed in Fig. 20.

Obtaining 3D wire frames. The next step is to convert the 2D structures into 3D wire frames. In order to do so, we assume that all lines that form the 2D structures arise from either vertical or horizontal scene edges. Furthermore, we use several features that aid us in relating an image to the 3D scene depicted in the image, including vanishing points, the ground plane constraint, propagation of 3D constraints, and colinearity (i.e., alignment of lines).

First, the lines that form the 2D structures are labeled as either "vertical" or "horizontal" depending on whether or not they are directed toward the vertical vanishing point [Kender 83]. Next, we use the position of the vertical vanishing point to calculate the vector in the vertical direction, as described in section 3. Let us now consider how to recover the 3D configuration of the junction $p_1 p_2 p_3 p_4$ in Fig. 21. Suppose that line $p_2 p_4$ has been labeled "vertical" and lines $p_1 p_2$ and $p_2 p_3$ have been labeled "horizontal". Let u be the unit vector in the vertical direction. This vector is normal to all horizontal planes. First we would like to determine the 3-space position of v_2 , corresponding to the junction point p_2 . Since it is impossible to determine the actual position of this point from a single image without special information, the position is determined as some arbitrary point lying on the ray through p_2 , i.e., the depth a of v_2 is arbitrarily chosen. The horizontal plane $v_1 v_2 v_3$ can now be established, since it contains v_2 and its normal vector is u . The 3-space positions of the points v_1 and v_3 can then be computed as the intersections of this plane with the rays through p_1 and p_3 , respectively. Finally, the 3-space position of the point v_4 is computed as the intersection of the ray through p_4



Figure 19: Result of pruning the junction connections in Fig. 17 by determining whether segments in Fig. 16 adequately cover the area between each pair of connected junctions.

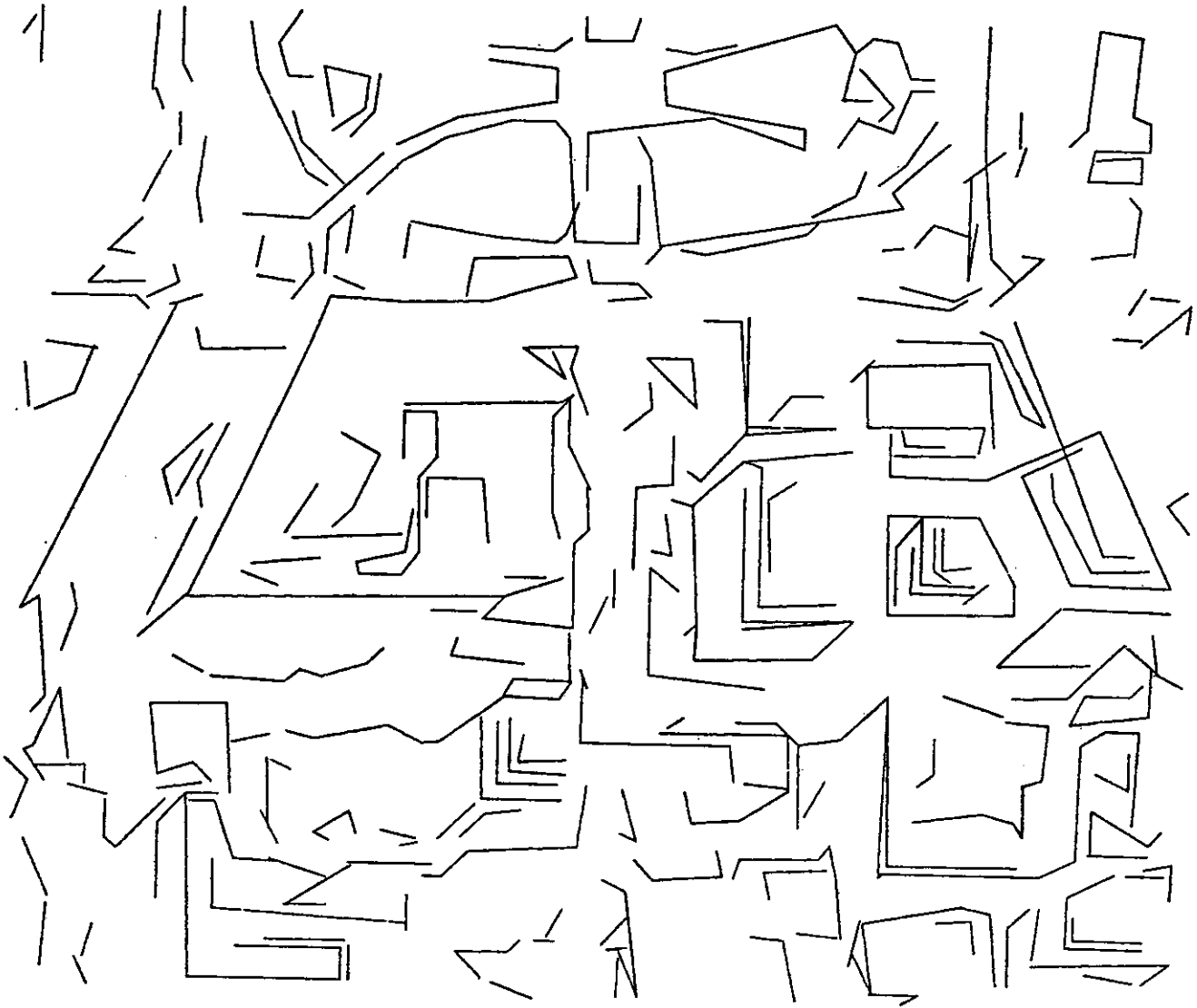


Figure 20: Result of adding to Fig. 19 the junction legs that were originally extracted in the junction finding step, and then deleting extraneous legs.

with the line through v_2 along the vector u .

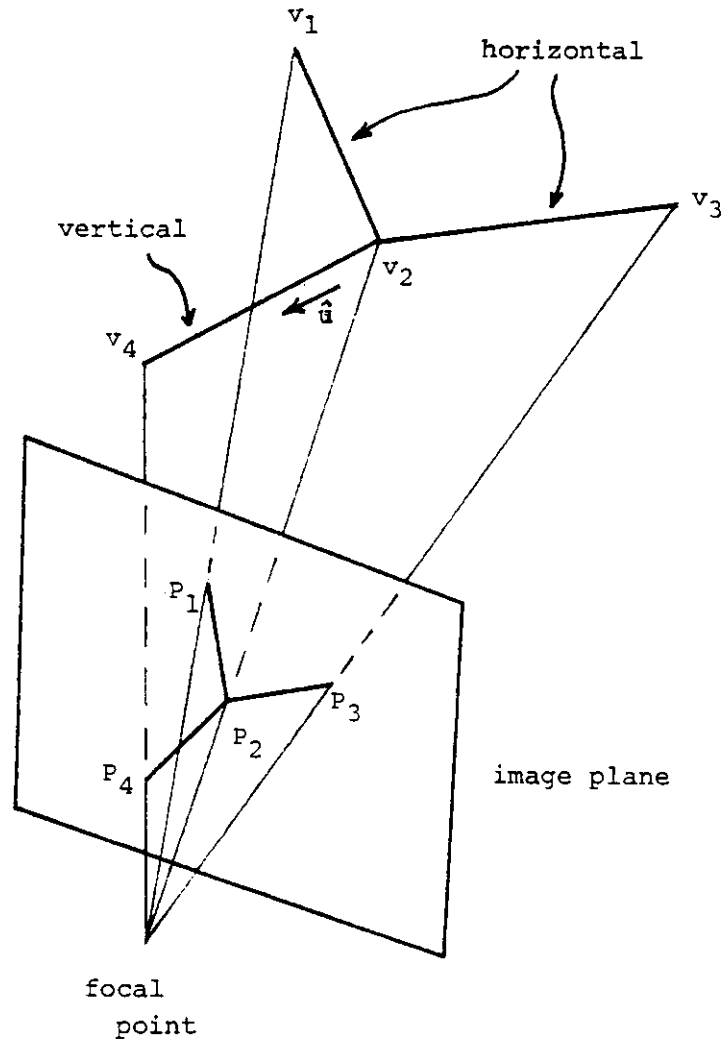


Figure 21: The 3D configuration of the junction $p_1p_2p_3p_4$ can be recovered under assumptions explained in the text.

Although this technique permits us to recover the 3D configuration of any junction relative to some arbitrary depth, it is not useful to apply it directly to the junctions in the original line image (Fig. 16) because the relative heights above the ground plane of the corresponding vertices cannot be determined: the height of each vertex is arbitrarily chosen without relation to the heights of other vertices. It is more useful, however, to apply the technique to the 2D structures in Fig. 20, since the heights of the vertices within each structure can be related. To see how this is done, consider the example in Fig. 22, which shows a 2D structure. The solid lines are part of the extracted structure (while the dashed lines are for the reader's convenience to make the 3D shape more apparent). Suppose lines p_1p_6 and p_3p_4 have been labeled "vertical", while the other solid lines have been labeled "horizontal". Applying our technique to (say) point p_1 , the 3-space positions of the vertices

corresponding to points p_1 , p_2 , and p_6 can be determined relative to some arbitrary depth a for p_1 . If the technique is applied next to point p_2 , the 3-space position of point p_3 can be determined as a function of the depth a . This procedure continues with points p_6 , p_4 , and so on, until the 3D configuration of the whole structure has been determined, relative to some arbitrary depth.

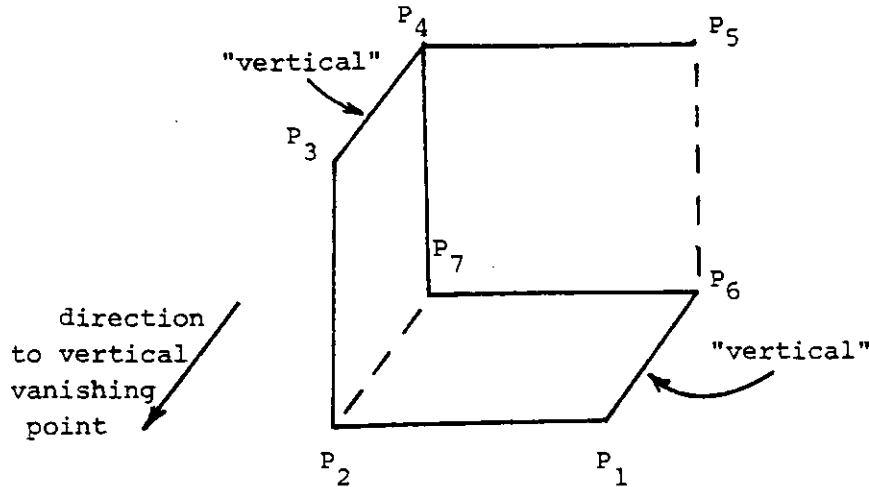


Figure 22: The solid lines represent a connected 2D structure. The dashed lines are for the reader's convenience to make the 3D shape more apparent.

In order to obtain a coherent scene description, the depths of the different structures in the scene must be related. We use two methods to do this. The first method involves finding structures that lie on the ground plane. Suppose a junction point p of such a structure is hypothesized to arise from a vertex lying on the ground. Then the 3-space position of the vertex may be obtained as the intersection of the ground plane with the ray through p . The normal vector u to the ground plane is known, but the distance d from the focal point to the ground plane is arbitrarily chosen. Since the 3-space position of all junctions arising from ground points can be calculated in this manner, the depths of all structures containing such points can be related to one another through the parameter d .

To hypothesize junctions that arise from vertices lying on the ground plane, we use the observation that if a line labeled "vertical" connects two junctions (e.g., line p_1p_6 in Fig. 22), the line is directed toward the vertical vanishing point with respect to one junction, but away from this vanishing point with respect to the other junction. The latter junction is assumed to represent a vertex lying on the ground plane. Points p_1 and p_3 in Fig. 22 are examples of such junctions. The 3-space positions of these junctions are then calculated, and their values are propagated throughout their structures as described previously. Fig. 23 depicts a perspective view of the 3D wire frames obtained in this manner.

There are many structures in Fig. 20 that do not contain points lying on the ground plane, either because such points are occluded in the scene or because they have not been properly extracted from the image.

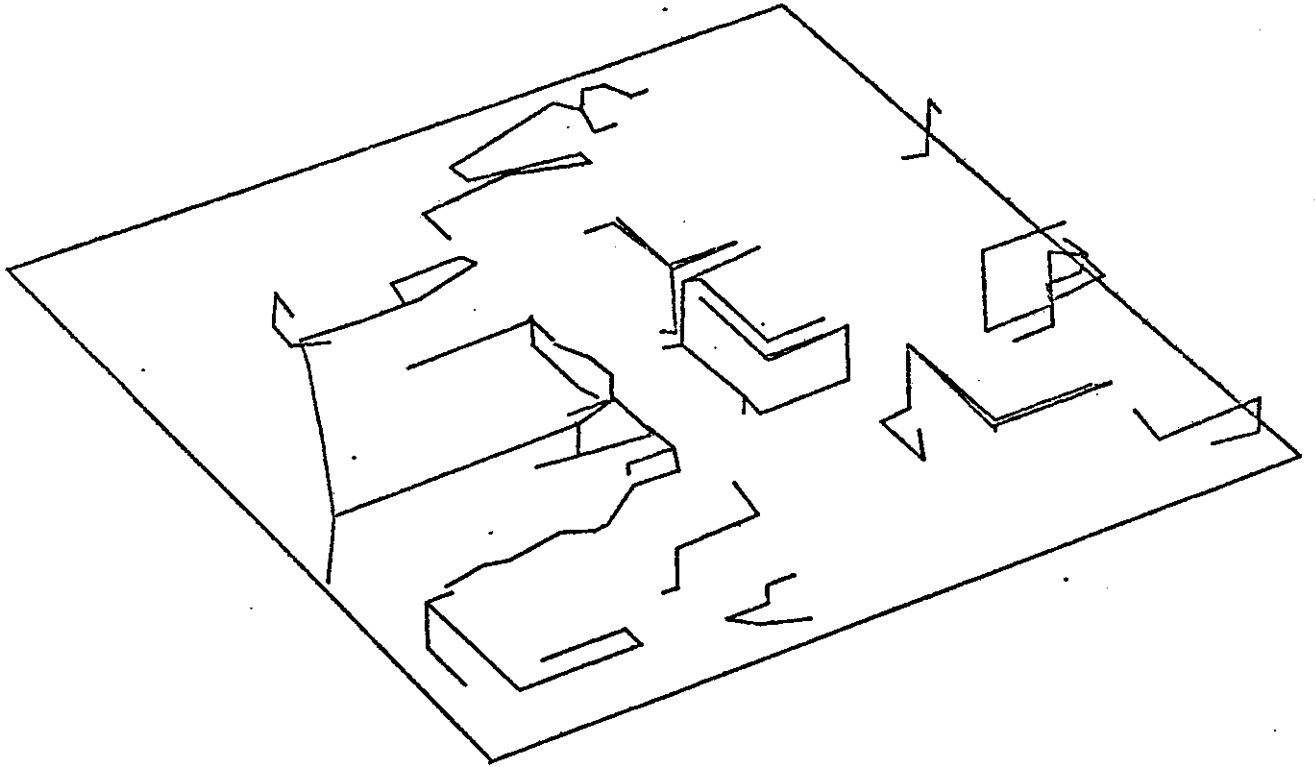


Figure 23: Perspective view of 3D wire frames generated from Fig. 20 using the method of finding junctions arising from vertices lying on the ground plane.

Nevertheless, the heights of some of these structures can be determined using the rule that if two lines are aligned in the image, they are often aligned in 3-space. This rule has been used in other systems [Lowe and Binford 81] and in fact is a restricted version of the parallel line rule [Kanade 81] which states that parallel lines in the image often arise from parallel lines in 3-space. To see how this rule is used, consider Fig. 24. Suppose that points p_1 through p_7 have already been assigned 3D coordinates, and we want to obtain the 3-space position of the 2D structure $p_8p_9p_{10}p_{11}$. Since the lines p_6p_7 and p_8p_{11} are aligned in the image and both are labeled "horizontal", they are assumed to be aligned in the scene and to lie in the same horizontal plane. The 3-space position of (say) point p_8 is therefore determined as the intersection of this plane with the ray through p_8 . The 3D coordinates of this point may then be propagated to points p_9 , p_{10} , and p_{11} as described previously. Note that all 3D positions are functions of the parameter d , which is arbitrarily chosen for the equation of the ground plane.

Fig. 25 depicts a perspective view of the final 3D wire frames obtained using both the methods of hypothesizing points on the ground plane and applying the alignment rule.

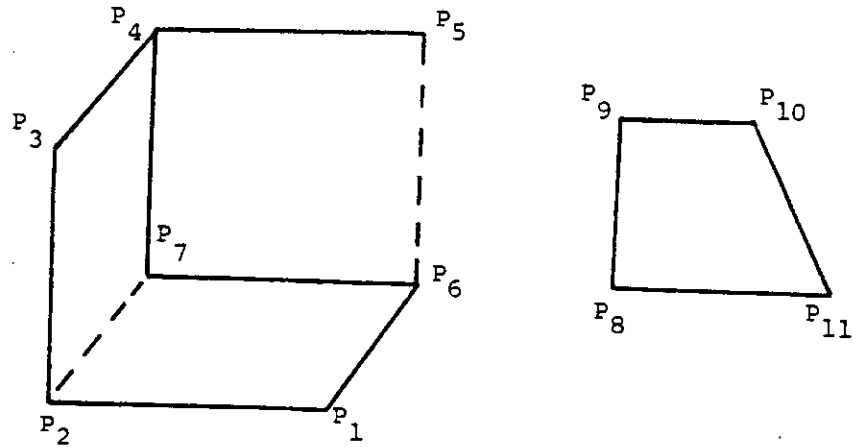


Figure 24: If the 3D configuration of the structure on the left has been determined, the relative 3D position of the structure on the right may also be determined because lines p_6p_7 and p_8p_{11} are aligned.

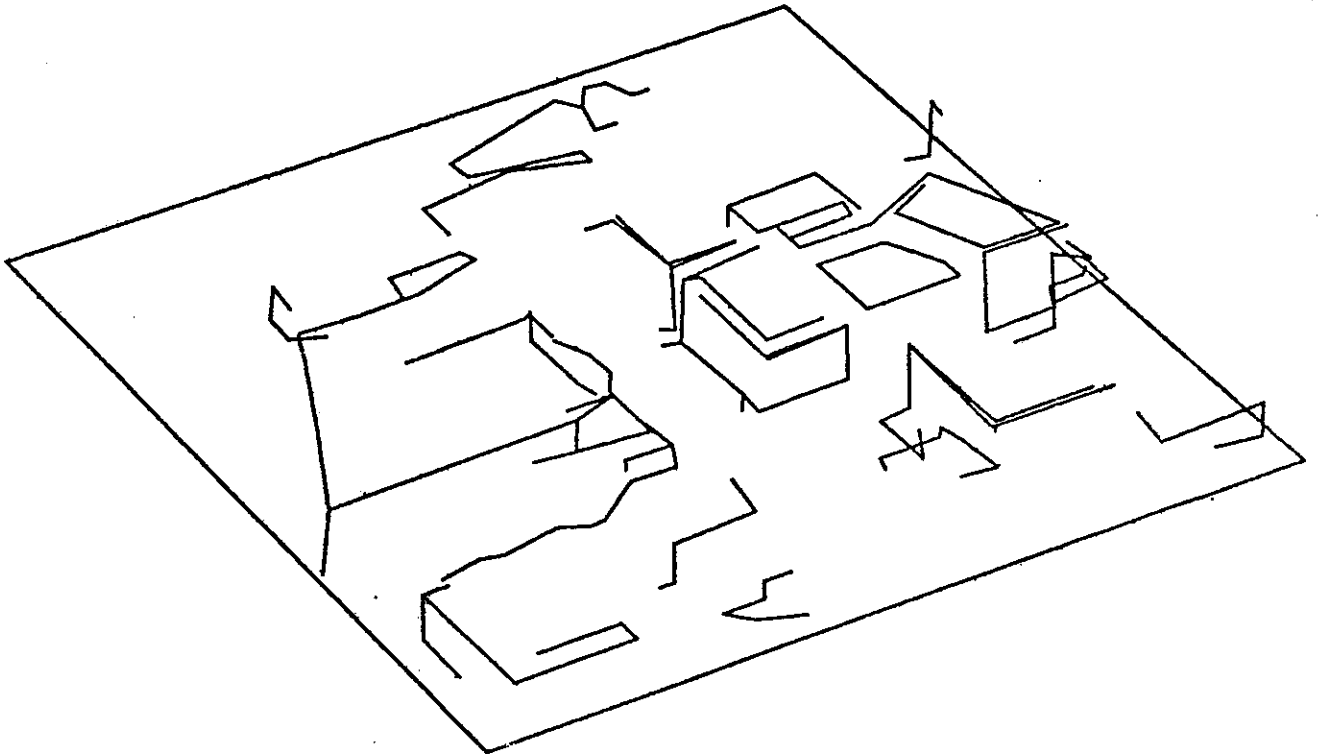


Figure 25: Perspective view of final set of 3D wire frames generated from Fig. 20.

5. Representing and Manipulating the 3D Scene Model

The representation we have developed for the 3D scene model draws on ideas from geometric modelling used in computer-aided design systems [Baer, Eastman, and Henrion 79, Requicha 80]. In these systems, however, the 3D models are usually derived through interaction with a user. Our case is different in that (1) the 3D models are derived automatically from 2D images, and (2) many portions of the scene are unknown or recovered with errors because of occlusions or unreliable analysis.

The following factors have determined how the scene model is represented and manipulated.

1. Partially complete, planar-faced objects must be efficiently described by the model. It is therefore represented as a graph in terms of symbolic primitives such as faces, edges, vertices, and their topology and geometry. Information is added and deleted by means of these primitives.
2. The model must be easy to use in matching.
3. Because scene approximations are often more useful if they contain reasonable hypotheses for parts of the scene for which there are partial data, we introduce mechanisms that permit hypotheses to be generated, added, and deleted.
4. Because incremental modifications to the model must be easy to perform, we introduce mechanisms to (a) add primitives to the model in a manner such that constraints on geometry imposed by these additions are propagated throughout the model, and (b) modify and delete primitives if discrepancies arise between newly derived and current information.

5.1. Representation of Model

The 3D structure in the scene is 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, their effects are propagated to other parts of the graph so as to obtain globally consistent interpretations.

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 [Baer, Eastman, and Henrion 79, Eastman and Preiss 82]. The geometry involves the physical dimensions and location in 3-space of each component, while the topology involves connections between the components.

Nodes in the structure graph represent either primitive topological elements (i.e., faces, edges, vertices, objects, and edge-groups (which are rings of edges on faces)) or primitive geometric elements (i.e., planes, lines, and points). Face, edge, and vertex nodes are tagged as either *confirmed* or *unconfirmed*. Confirmed means that the element represented by the node has been derived directly from 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 structure 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.

Fig. 26 shows a simple example of a structure graph consisting of two objects, *ob1* and *ob2*. Arrows with single lines represent part-of links, and arrows with double lines represent geometric constraint links. The faces are represented as f_i , the edge-groups as g_i , the edges as e_i , and the vertices as v_i . The graph shows one point node pt and one plane node pl .

5.2. Modifications to 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. 27 shows how a constraint on one node may propagate to others. The arrows in the figure indicate the direction of propagation. Point constraints propagate upward. That is, if a point constrains a vertex, it must also constrain all edges and faces which contain that vertex. Similarly, a point that constrains an edge also constrains all faces containing that edge. Line constraints propagate outward, and plane constraints propagate downward. Whenever a geometric constraint link is added, propagation occurs as indicated in Fig. 27.

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 [Doyle 79], using the notion that when an assertion is deleted, all assertions implying it and all assertions implied by it that have no other support should also be deleted. We obtain assertions that imply a given assertion by following backwards along the arrows in Fig. 27, and we obtain assertions implied by a given assertion by following forward along the arrows.

Consider the simple example in Fig. 28a, 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. 28b), that is, the

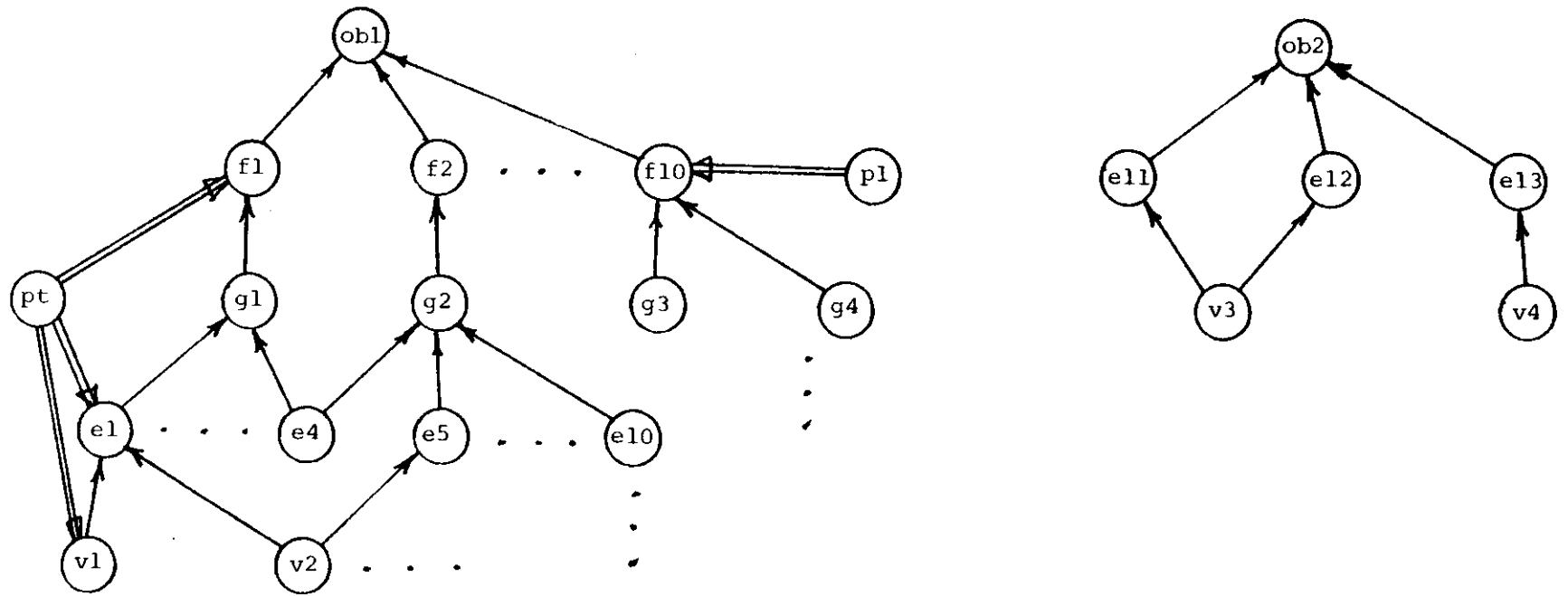


Figure 26: Simple example of a structure graph consisting of two objects, *ob1* and *ob2*. Double line arrows represent geometric constraint links, and single line arrows represent part-of links.

assertion " p constrains e " is deleted. All assertions which have implied this must now be deleted, for if one were to hold, link 4 would also hold. To find these assertions, we locate the box in Fig. 27 that represents a point constraining an edge and follow backwards along the arrow. The result is the box that represents the point constraining any vertex of the edge. In Fig. 28b, this corresponds to 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. 28b. Our intuition tells us that part-of links (link 1) should dominate constraint links (link 3), 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. All these assertions must also be deleted unless they have other support. To do so, we follow forward along the arrow from the box in Fig. 27 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. 28b, this corresponds to the assertion " p constrains f ", which is link 5. This link should therefore be deleted since it has no other support. The resulting structure graph is depicted in Fig. 28c.

6. Generating the 3D Scene Model

The result of image analysis is a 3D wire-frame description that represents 3D vertices and edges which correspond to portions of boundaries of objects in the scene. We construct a surface-based description -- the 3D scene model -- from these boundaries by hypothesizing new vertices, edges, and faces. Both the wire-frame and surface-based descriptions are represented by structure graphs.

Our current techniques for hypothesizing the scene model will be shown next using an example that starts with the output of the stereo analysis component depicted in Fig. 29. These techniques provide a method for hypothesizing parts of the scene for which there are only partial data by exploiting task-specific knowledge. The various thresholds used throughout this example have been manually chosen.

Combine edges. First, if two wire-frame edges are nearly parallel and very close to each other, they are merged into a single edge. This occurs only once in Fig. 29, 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 (Fig. 30a).

Merge partial faces. After all web faces have been created, those that represent portions of a single face are






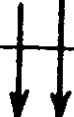
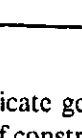

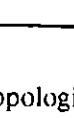
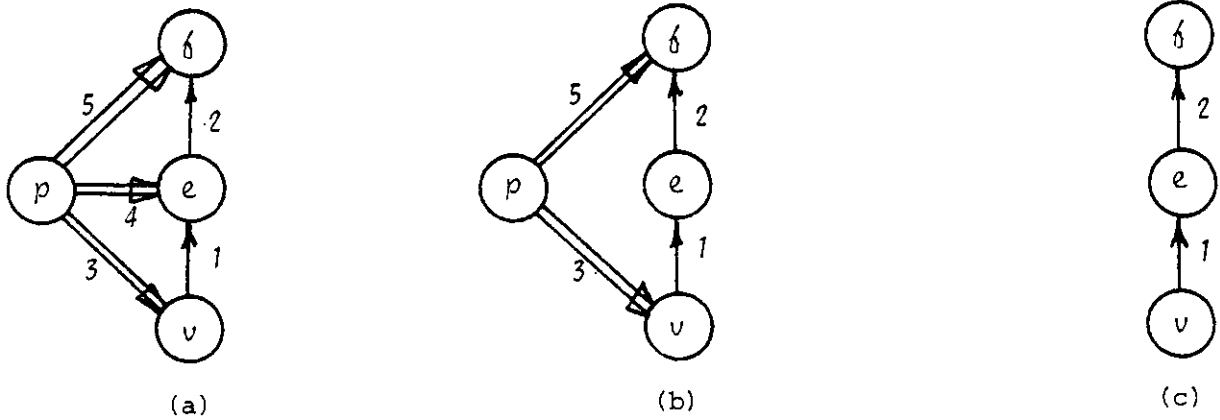
	Point	Line	Plane
face			
edge			
vertex			

Figure 27: 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 28: (a) Initial structure graph. (b) Link 4 is deleted. (c) Resulting structure graph after effects of deletion have been propagated.

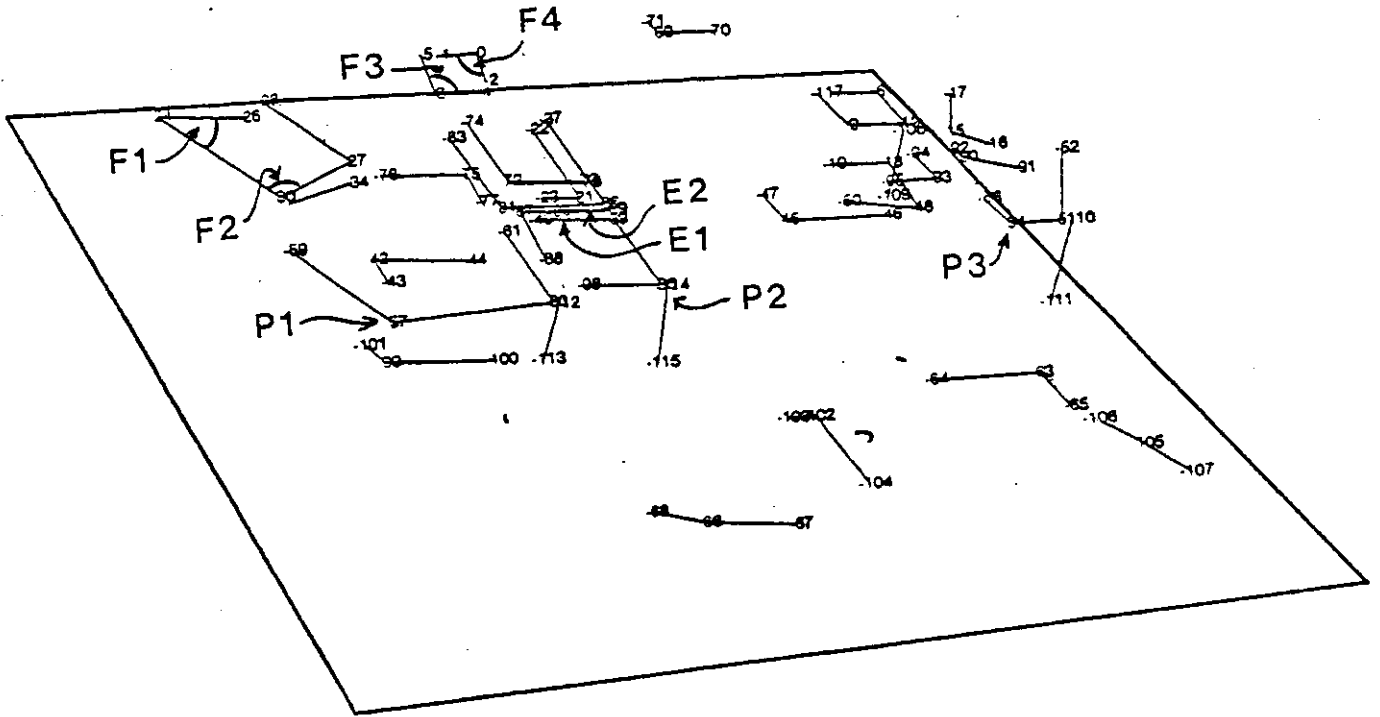


Figure 29: Perspective view of 3D vertices and edges extracted from stereo pair in Fig. 4. This version is different from the one shown in Fig. 13.

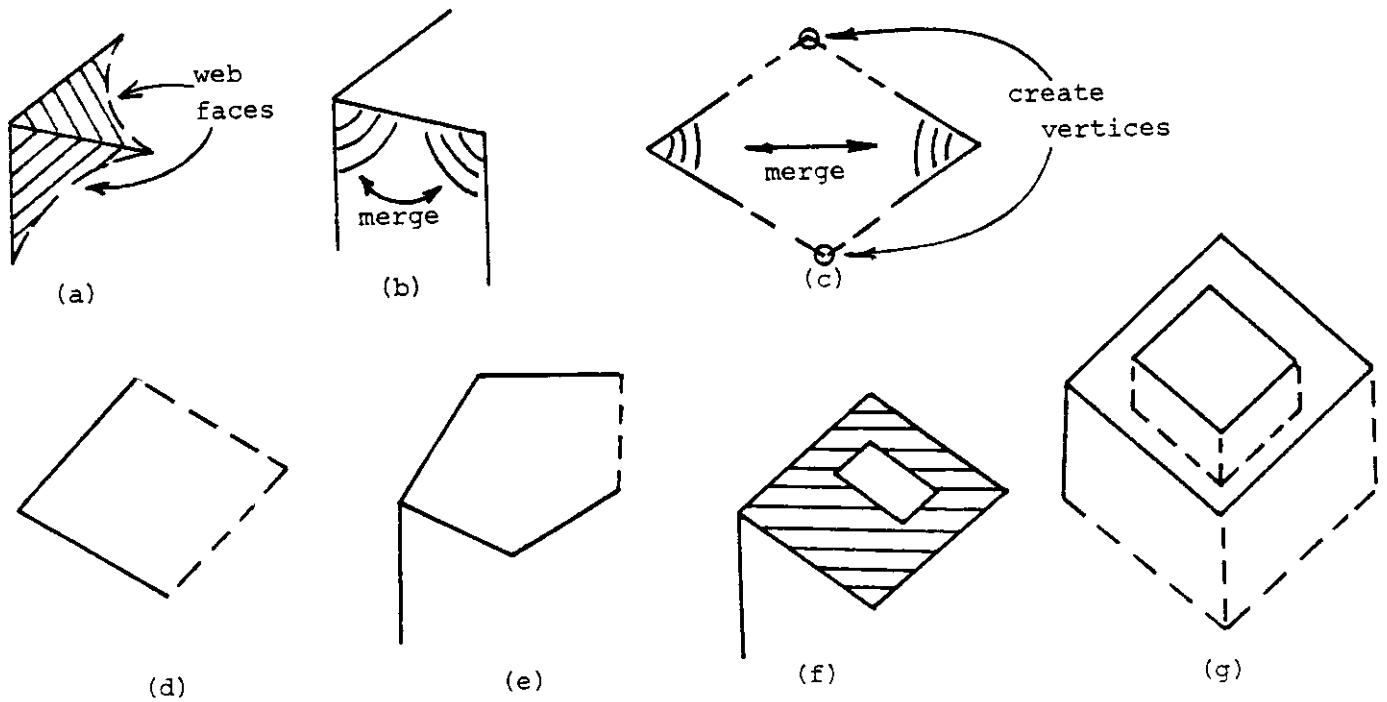


Figure 30: Obtaining a surface-based description from wire frames.

merged. Two partial faces that touch each other (e.g., Fig. 30b, and F1 and F2 in Fig. 29) 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 touch each other (e.g., Fig. 30c, and F3 and F4 in Fig. 29) 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 is 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 to each other. 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 because they do not have closed boundaries. 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 (Fig. 30d), the shape is completed as a parallelogram.
2. If the partial face contains three or more edges connected as a single chain (Fig. 30e), 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 (Fig. 30f). 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. Since we are dealing with urban scenes, faces that lie high enough above the ground are assumed to represent roofs of buildings. A hypothesized vertical wall is dropped towards the ground from each edge of such faces, unless the edge is already part of another face (Fig. 30g). Each wall is dropped either to the ground or to the first face it intersects on the way down. The equation of the ground plane is currently interactively obtained. The 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. 31 shows several perspective views of the resulting scene model. Notice that one of the buildings has a hole in it, through the roof. The planar patches at the "front" of the scene are part of the ground. Because they were not high enough above the ground plane, they were not treated as building roofs. Fig. 32 shows the scene model generated when these techniques are applied to the wire-frame description obtained using monocular analysis (Fig. 25). Note that all vertices, edges, and faces which have been hypothesized by the procedures described above are marked as such, and will be replaced by more correct versions as more information becomes available from new views.

6.1. Comparison with Depth Map

There are several interesting points about the model generated from the stereo output. First, notice that it is a higher level description than a depth map. The product of many stereo analysis systems is a depth map [Baker and Binford 81, Grimson 80, Ohta and Kanade 83] which, like an image, is an array of numbers that will have to be converted into a higher level description. Our approach, on the other hand, has been to extract a set of 3D features using stereo analysis (as shown in Fig. 29) and to use task-specific knowledge to go directly to a higher level 3D description. This description is symbolic and much more compact than one based on surface points, and allows relative sizes and positions of scene objects or their parts to be easily available. This facilitates matching and updating the model with 3D information derived from subsequent views, matching the model with other models, generating and deleting hypotheses for parts of the model, and computing structural features of the model.

6.2. Mapping Gray Scale onto Faces

In order to render more realistic displays, gray scale is added to them [Devich and Weinhaus 80]. This is useful for realistically simulating the appearance of the scene from arbitrary viewpoints. We associate with each face in the model a normalized intensity image patch of the face. These patches are currently extracted from a single image of the scene, but may eventually be extracted from multiple images. For faces that are partially occluded in the image, the intensity patch is associated with the unoccluded portions. 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. Figs. 33 and 34 show the results of adding gray scale to the faces of the models in Figs. 31 and 32, respectively. On a color display, faces and parts of faces that are occluded in the original image are displayed in red. This can be used in a task such as planning flight paths to obtain more images of the scene. The optimal path might be one in which the maximum amount of red portions can be viewed. 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.



Figure 31: Perspective views of buildings reconstructed from wire-frame data in Fig. 29. These buildings correspond to the cluster of buildings at the upper middle parts in the images of Fig. 4.

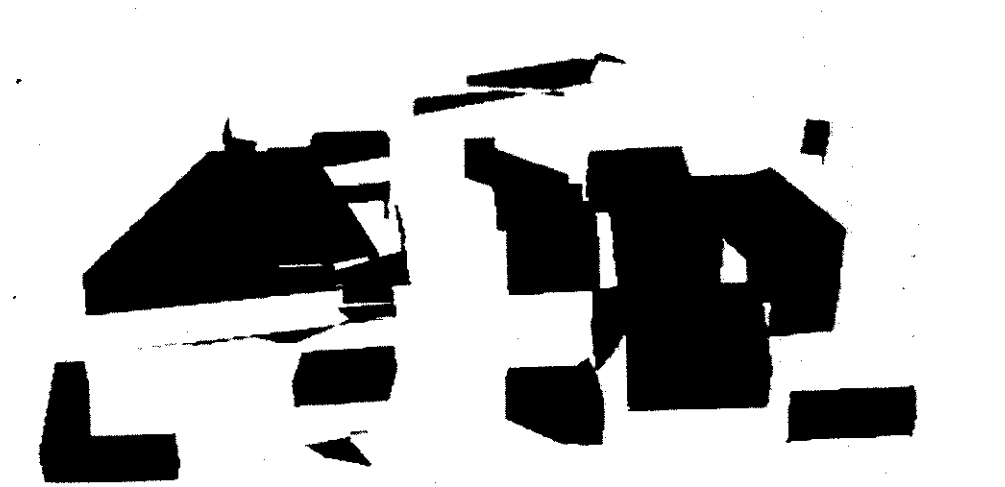
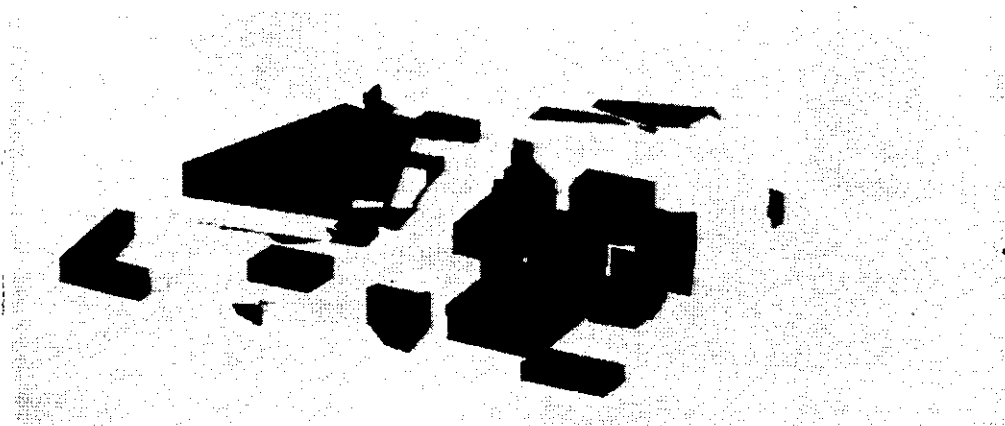
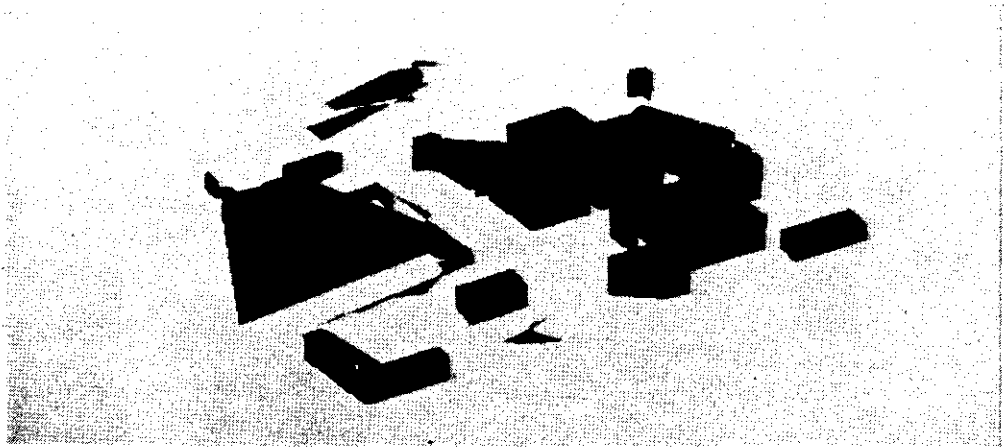


Figure 32: Perspective views of buildings reconstructed from wire-frame data in Fig. 25.

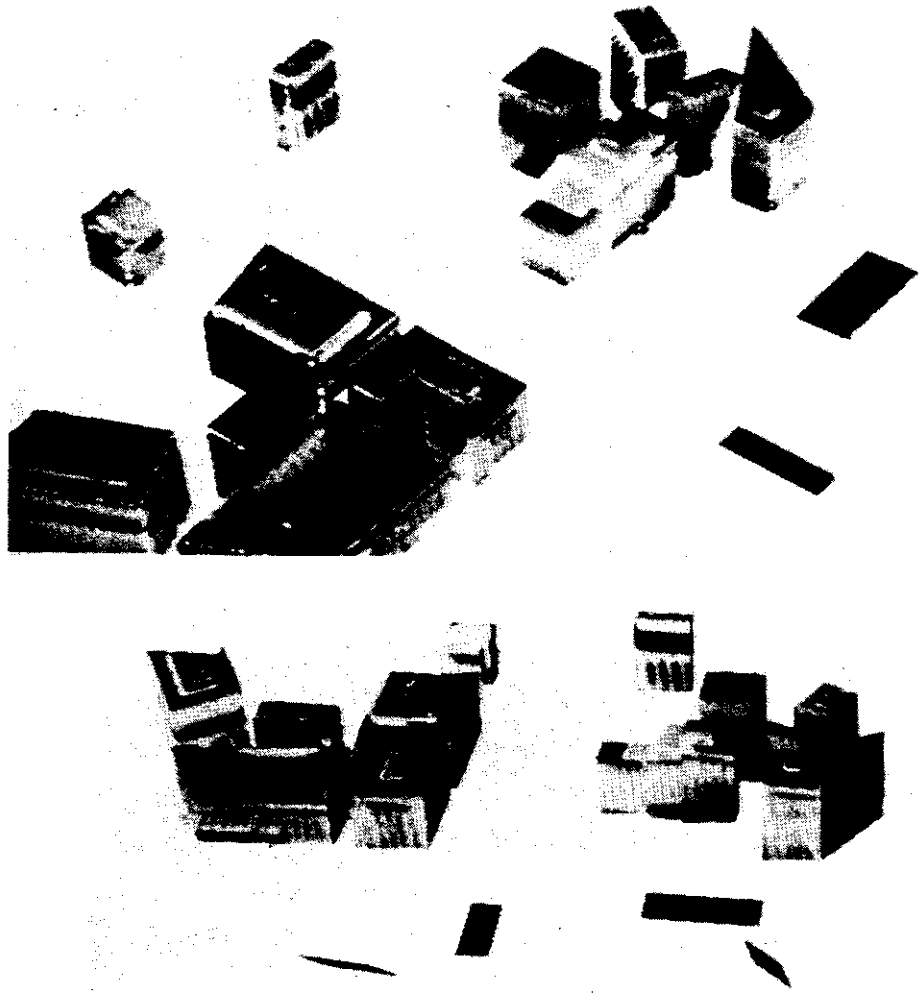


Figure 33: Reconstructed buildings of Fig. 31 with gray scale, derived from the left image in Fig. 4, mapped onto faces. On a color display, faces and portions of faces occluded in the original image are colored red.

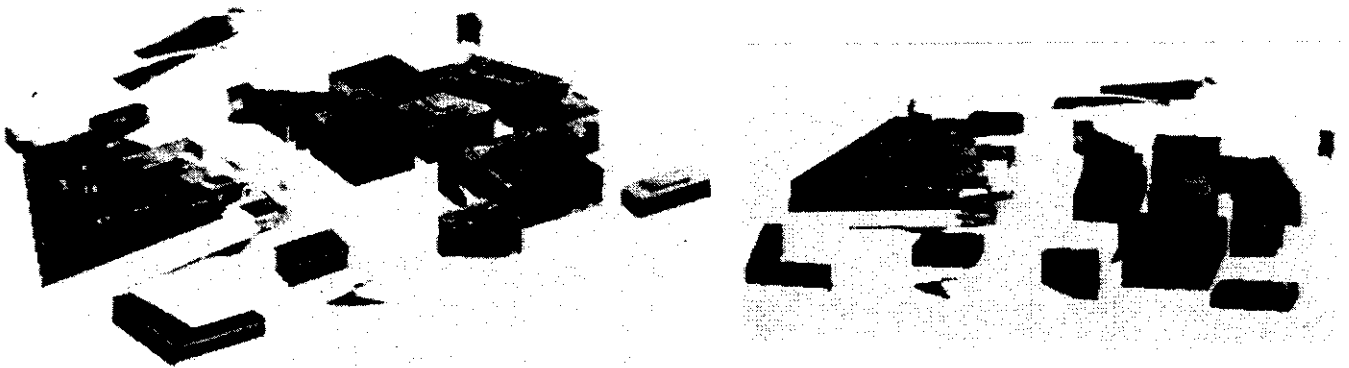


Figure 34: Reconstructed buildings of Fig. 32 with gray scale, derived from Fig. 14, mapped onto faces.

7. Combining New Views with Current Model

The process of incorporating a 3D wire-frame description extracted from a new view into the current scene model can be divided into three main steps:

1. The wire-frame data must first be matched to the current model. This process provides (a) the scale transformation and coordinate transformation from the wire-frame data to the model, and (b) corresponding elements (i.e., vertices and edges) in the two.
2. The new wire-frame data is then merged with the current model. This process includes (a) merging pairs of corresponding elements, and (b) adding to the model wire-frame elements for which no correspondences were found. The latter procedure is aided by knowledge of the scale and coordinate transformations. During the merging process, hypothesized parts of the model that are inconsistent with the new wire-frame data are deleted.
3. At this point, many objects in the model may be incomplete because (a) new wire-frame data has been added, and/or (b) some hypothesized elements have been deleted. These objects are completed using the techniques described in section 6.

To see how these steps are carried out, consider the example of incorporating the information from a second view into the scene model of Fig. 31. This scene model was constructed from the set of wire frames (Fig. 29) automatically extracted from a "front" view of the scene (Fig. 4). The second set of wire frames, shown in Fig. 35, was manually generated to simulate information available from an opposing point of view (viewing the scene from the "back"). The viewpoint for the perspective drawing of Fig. 35 is chosen to be similar to that of Fig. 29 to allow easier comparison by the reader. Notice that the information in Fig. 29 emphasizes edges and vertices facing the front of the scene, while those facing the back of the scene are emphasized in Fig. 35.

7.1. Matching

We assume in this example that the scale and coordinate transformations from the new wire-frame data to the current model is known; the data and model may therefore be described in the same coordinate system. We have not yet implemented a general matcher that provides these transformations between the two.

The next step is to determine corresponding edges and vertices in the data and model. First we label each connected group of edges in the wire-frame data as a distinct wire-frame object. Next, wire-frame objects are matched with model objects. Two objects are said to match if they have confirmed parts that match. Matches are sought only for edges and vertices, since these constitute the only confirmed parts of a wire-frame object. The requirements for two confirmed vertices, one from each object, to match are: (1) they must be very close to each other, or (2) they must be part of matching edges whose other two vertices match. The requirements for two confirmed edges, one from each object, to match are: (1) the two confirmed vertices of one edge must match the two of the other, or (2) one confirmed vertex on one edge matches one on the other, and the two

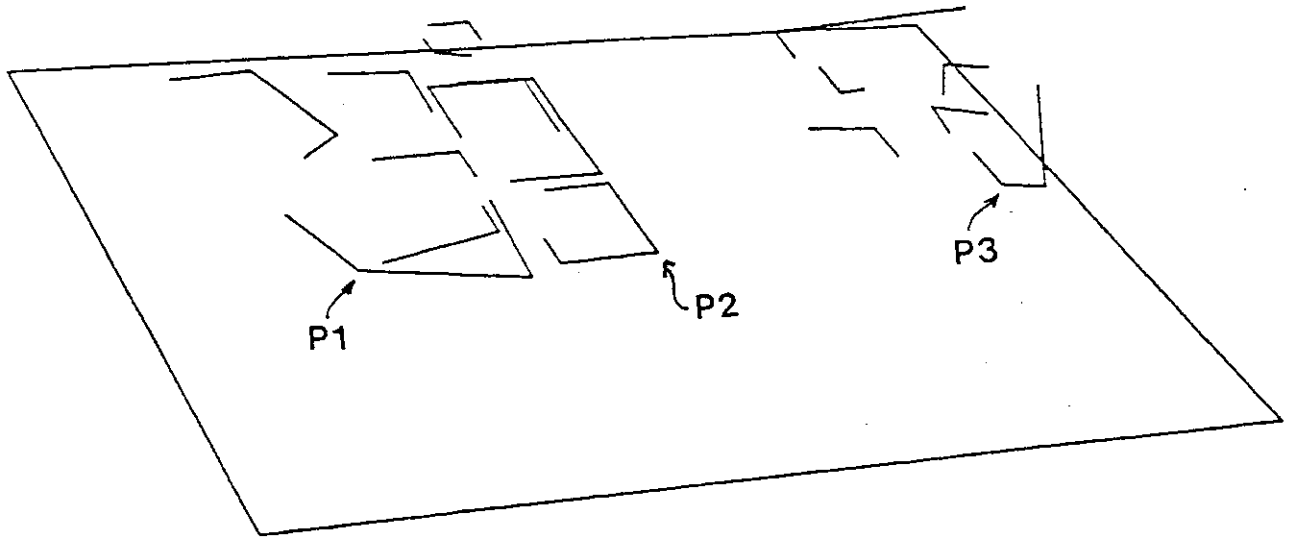


Figure 35: Perspective view of manually generated vertices and edges which simulate information available from images showing an opposite point of view from that shown in Fig. 4. The viewpoint for this drawing is chosen to be similar to Fig. 29. Points P1, P2, and P3, for example, correspond to points P1, P2, and P3 in Fig. 29.

edges are close together and overlap in their lengths. These rules are used in a relaxation algorithm to obtain matching vertices and edges.

7.2. Discrepancies

We must now merge the new wire-frame data into the model. An important issue here is how to handle discrepancies between the two. We consider the following two types of discrepancies:

1. After the coordinate system of the wire-frame data has been transformed to that of the model and scale adjustments have been made, corresponding pairs of confirmed vertices and edges may not register perfectly in 3-space. In order to merge them into single elements, we perform a "weighted averaging" of their positions.
2. Hypothesized elements in the model may be inconsistent with newly obtained elements. We handle this by deleting such hypothesized elements.

To determine whether or not hypotheses are still valid when confirmed elements in the model are modified or deleted, we consider the elements which gave rise to the hypotheses. A hypothesis is dependent on all elements whose existence directly resulted in the creation of the hypothesis. If one of these elements is modified or deleted, the hypothesis must also be modified or deleted since the conditions under which it was created are no longer valid. The dependency relationships for hypothesized elements are explicitly recorded at the time of their creation using dependency pointers [Doyle 81].

The following examples show how some of these relationships are recorded:

1. When two non-touching partial faces are merged, (Fig. 36a) each face has two edges which are intersected with their counterparts in the other face. The intersection points form two new hypothesized vertices, each of which is dependent on the two edges whose intersection gave rise to it. In Fig. 36a, the arrows indicate the dependencies. Vertex $v1$ is dependent on edges $e1$ and $e3$, and vertex $v2$ is dependent on edges $e2$ and $e4$. If one of the edges were to be modified (e.g., if its position were to be displaced), the vertex that depends on that edge would no longer be a valid hypothesis, and would therefore be deleted. A new vertex might then be hypothesized.
2. When a face is completed by connecting its two end points (Fig. 36b), two new vertices and one new edge are hypothesized. The new edge $e4$ is dependent on both $e1$ and $e3$, while the new vertices $v1$ and $v2$ are dependent on the edges on which they lie.
3. When a vertical wall is dropped from a face, the first step is to drop hypothesized edges from vertices of the face. Such edges are dependent on the vertices from which they are dropped. In Fig. 36c, the new edges $e1$ and $e2$ are dropped from, and are dependent on, the vertices $v1$ and $v2$, respectively. A dropped edge is constrained to be perpendicular to the ground plane, and would therefore no longer be a valid hypothesis if the vertex it depends on, which is one of its end points, were to be displaced.

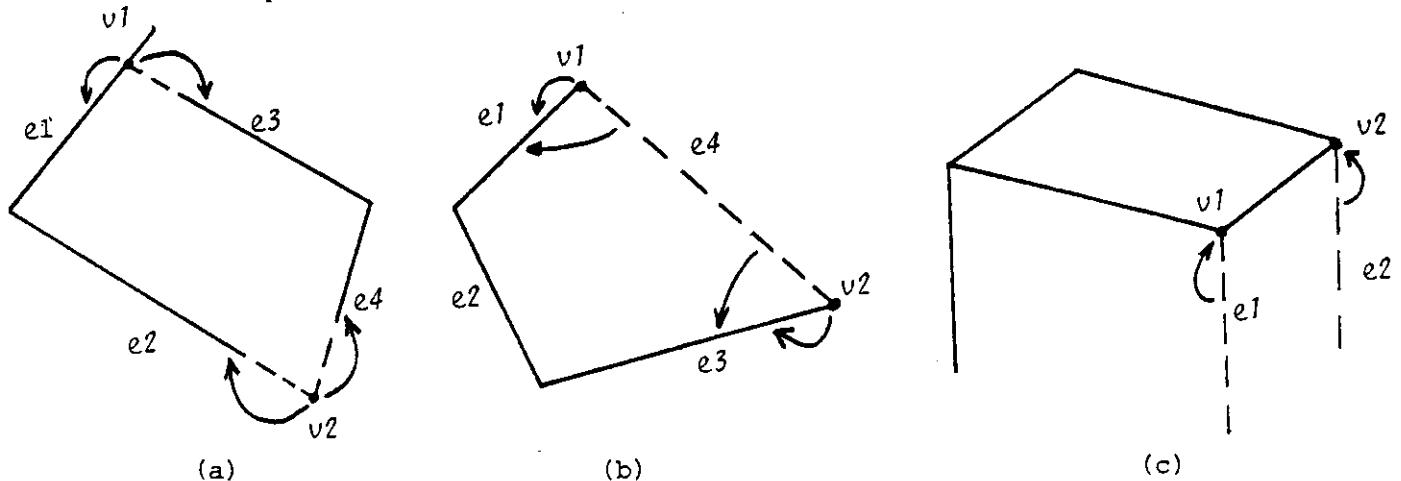


Figure 36: Generating dependencies for hypothesized edges and vertices. The dependence of an element on another is depicted as an arrow from the former to the latter. (a) Two non-touching partial faces are merged. (b) A face is completed. (c) Vertical edges are dropped from a floating face.

When a confirmed edge or vertex in the model is modified or deleted, the set of all hypothesized elements that depend on it are deleted. Recursively, elements depending on deleted ones are also deleted. When hypothesized vertices and edges are deleted in this manner, it is possible for hypothesized faces to lose minimal support, i.e., they may no longer be constrained by at least three non-colinear points. Such faces are also deleted.

7.3. Merging

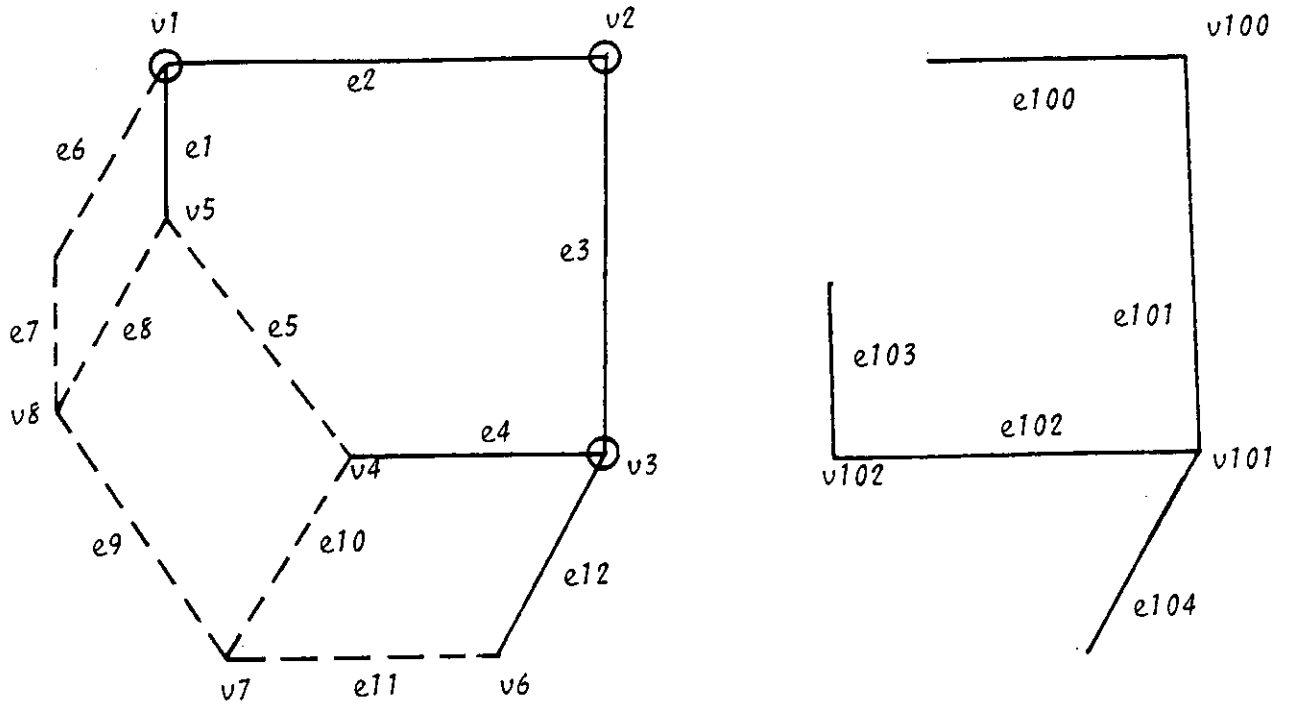
The procedure that merges corresponding wire-frame and model objects takes into account the fact that the 3-space positions of end points of edges that are confirmed vertices are generally much more accurate than the positions of non-vertex end points. Therefore, confirmed vertices are given more weight during merging. As an example, consider Fig. 37. Suppose the wire-frame object in (b) is to be merged with the model object in (a), and the corresponding edges and vertices are as follows: $(v2, v100)$, $(v3, v101)$, $(e2, e100)$, $(e3, e101)$, $(e4, e102)$, $(e12, e104)$. We assume the wire-frame object has been transformed to register with the model object. The solid lines in the model represent confirmed edges; the dashed lines represent hypothesized edges.

The merging procedure starts by merging corresponding vertices. Pairs of vertices ($(v2, v100)$ and $(v3, v101)$ in Fig. 37) are combined into single vertices with coordinates of the midpoint between them. If the distance between an initial pair of vertices exceeds a threshold, all hypothesized elements that recursively depend on the initial model vertex are deleted. At this point, all corresponding pairs of edges will share at least one vertex. The corresponding edges are merged next as follows:

1. If the two edges share both their vertices ($(e3, e101)$ in Fig. 37), the new edge connects the two new vertices already generated.
2. If one edge has two confirmed vertices but the other does not ($(e2, e100)$ and $(e4, e102)$ in Fig. 37), the new edge is the same as the former. Notice that the non-vertex end point in this case is given zero weight.
3. If the two edges share one vertex and the other end points are not confirmed ($(e12, e104)$ in Fig. 37), the new edge is the "average" of the two edges.

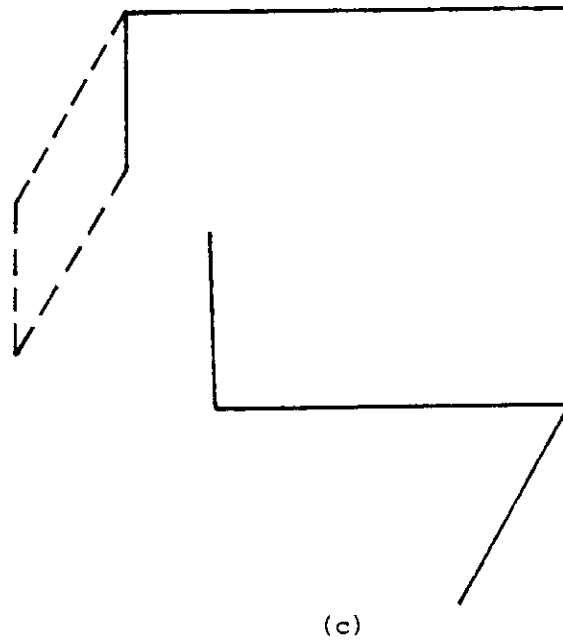
If a model edge to be merged contains only one confirmed vertex (e.g., $e4$ and $e12$ in Fig. 37), then all hypothesized elements that recursively depend on this edge are deleted. For example, the hypothesized elements that recursively depend on $e4$ in Fig. 37 are the vertices $v4$ and $v7$, and the edges $e5$, $e10$, $e9$, and $e11$. If a model edge to be merged contains two confirmed vertices (e.g., $e2$ and $e3$ in Fig. 37), no hypothesized elements need be deleted since all necessary deletions were done when the vertices of the edge were merged.

After all corresponding elements of the two objects have been merged, the edges and vertices remaining in the wire-frame object that were not merged ($e103$ in Fig. 37) are added to the model object. The final configuration after merging is shown in Fig. 37c. This object is incomplete and must be completed using the techniques described in section 6.



MODEL OBJECT
(a)

WIRE-FRAME OBJECT
(b)



(c)

Figure 37: The wire-frame object in (b) is to be merged with the model object in (a). The confirmed edges of the model object (indicated by solid lines) are $e1$, $e2$, $e3$, $e4$, and $e12$; the confirmed vertices (indicated by circles) are $v1$, $v2$, and $v3$. Dashed lines represent hypothesized edges. (c) The result after merging.

7.4. Results of Merging

When these procedures are applied to the wire-frame data in Fig. 35 and the scene model in Fig. 31, we obtain the updated scene model shown in Fig. 38. The updated version has two important improvements over the initial version. First, the updated model contains more buildings since new wire-frame data, some of which represent new buildings, have been incorporated into the initial model. Second, for many buildings described in both versions of the model, the positions of vertices and edges are more accurate in the updated version. This is because many hypothesized vertices and edges are replaced by accurate ones obtained from the new data, and many confirmed vertices and edges are merged with corresponding ones in the data by "averaging" their positions, generally decreasing the amount of error.

The shape of the large hole in the roof of one of the buildings has changed from a rectangle in the initial model to an almost triangular quadrilateral in the updated version. When compared with the source images in Fig. 4, the rectangular shape would seem more accurate. However, the positions of the edges and vertices that form the hole are more accurate in the updated model in the sense that they are more faithful to the wire-frame descriptions derived from the images.

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

8. Summary

The 3D Mosaic system acquires an understanding of the 3D configuration of surfaces and objects in a scene. The system encompasses several levels of the vision process, starting with images and ending with symbolic scene descriptions. Because the scenes considered are highly complex, we use multiple views so that more information can be extracted than from a single view. This has led to an incremental approach for acquiring the scene model. As a result, the following capabilities are required:

1. Image analysis must extract as much scene information as possible from input images.
2. Partial scene descriptions must be represented and manipulated.
3. Incremental modifications and updates to the scene model must be easy to perform.
4. Mechanisms for generating, manipulating, and deleting hypotheses from the model must be introduced.

A view of the scene may be either a single image or a stereo pair. Two separate system components for extracting 3D information from images have been described: stereo analysis and monocular analysis. Both of these components extract sparse 3D wire-frame descriptions from the images. A component that converts these wire frames into a surface-based description has also been described.

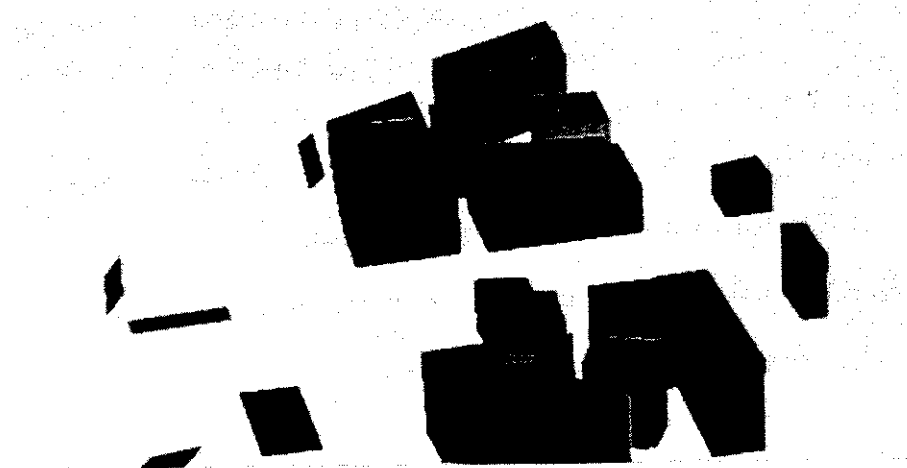
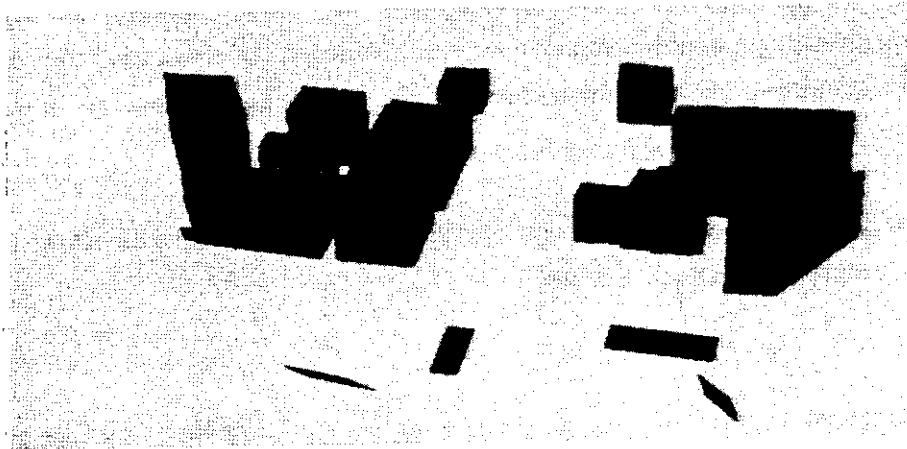
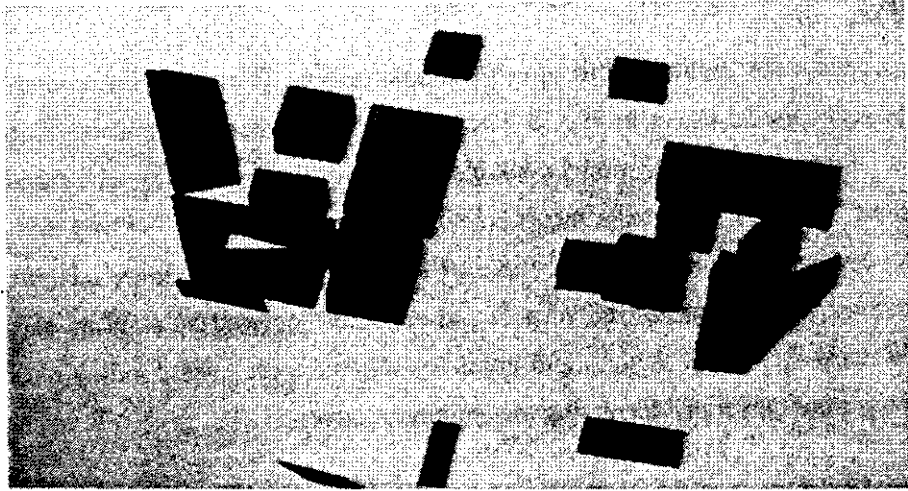


Figure 38: Perspective views of buildings derived by incorporating the wire-frame data in Fig. 35 into the model in Fig. 31.

We have demonstrated that task-specific knowledge is very useful for interpreting complex images. Knowledge of block-shaped objects in an urban scene is used for stereo analysis, monocular analysis, and reconstructing shapes from the wire frames. Our techniques have been demonstrated on complex aerial photographs of urban scenes.

There are several extensions and improvements to the system that we will pursue in the future:

1. Incorporating depth map data. Currently, our stereo analysis extracts a sparse set of wire frames from the images. We would also like to include a stereo algorithm that extracts depth maps [Baker and Binford 81, Grimson 80, Ohta and Kanade 83]. The depth map from a new view would have to be segmented into surfaces, edges, and vertices, and merged into the current model.
2. Improving the 3D matching. The algorithm that matches new 3D information with the current model should be improved so that it can provide the scale and coordinate transformations between the two. In addition, the current algorithm, which considers only edges and vertices when performing the matching, will have to be extended to include faces that may be directly obtained from a depth map.
3. Using the current model to interpret a new view. Currently, 3D information is extracted from a new view without using any information available in the model generated from previous views. Making use of the current model may aid in segmenting a depth map, or extracting 3D information from a single image.
4. Improving the monocular analysis by using other monocular cues, such as shadows [Shafer and Kanade 82] and texture.

Acknowledgement

Shigeru Kuroe worked extensively on the stereo system and low-level edge detection, linking, and line fitting algorithms. Duane Williams has provided much programming support, especially with the program that maps gray scale onto the model faces. Fumi Komura explored and experimented with initial concepts dealing with this project. Yuichi Ohta and Steve Shafer have provided useful comments and criticism.

References

- [Baer, Eastman, and Henrion 79]
 Baer, A., Eastman, C., and Henrion, M.
 Geometric Modelling: a Survey.
Computer-Aided Design 11:253-272, September, 1979.
- [Baker 77] Baker, H. H.
 Three-Dimensional Modelling.
Proc. IJCAI-77 :649-655, August, 1977.
- [Baker and Binford 81]
 Baker, H. H., and Binford, T. O.
 Depth from Edge and Intensity Based Stereo.
Proc. IJCAI-81 :631-636, 1981.
- [Barnard 82] Barnard, S. T.
 Methods for Interpreting Perspective Images.
Proc. ARPA Image Understanding Workshop , September, 1982.
- [Barnard and Fischler 82]
 Barnard, S. T. and Fischler, M. A.
 Computational Stereo.
Computing Surveys 14(4), December, 1982.
- [Barnard and Thompson 80]
 Barnard, S. T. and Thompson, W. B.
 Disparity Analysis of Images.
IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-2(4):333-340, July, 1980.
- [Barrow, Bolles, et al. 77]
 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 :696, August, 1977.
- [Baumgart 74] Baumgart, B. G.
Geometric Modeling for Computer Vision.
 Technical Report STAN-CS-74-463, Department of Computer Science, Stanford University, Stanford, CA, October, 1974.
- [Bourne, Milligan, and Wright 82]
 Bourne, D. A., Milligan, R., Wright, P. K.
 Fault Detection in Manufacturing Cells Based on Three Dimensional Visual Information.
Proc. SPIE , May, 1982.
- [Devich and Weinhaus 80]
 Devich, R. N., and Weinhaus, F. M.
 Image Perspective Transformations.
Proc. SPIE 238:322-332, July, 1980.

- [Doyle 79] Doyle, J.
A Truth Maintenance System.
Artificial Intelligence 12 :231-272, 1979.
- [Doyle 81] Doyle, J.
Three Short Essays on Decisions, Reasons, and Logics.
Technical Report STAN-CS-81-864, Department of Computer Science, Stanford University, Stanford, CA, May, 1981.
- [Duda and Hart 73]
Duda, R.O. and Hart, P. E.
Pattern Classification and Scene Analysis.
John Wiley and Sons, New York, 1973.
- [Eastman and Preiss 82]
Eastman, C. M., and Preiss, K.
A Unified View of Shape Representation and Geometric Modeling.
Israel Conference on CAD, January, 1982.
- [Grimson 80] Grimson, W.E.L.
A Computer Implementation of a Theory of Human Stereo Vision.
Phil. Trans. Roy. Soc. Lond. B292 :217-253, 1980.
- [Hannah 74] Hannah, M. J.
Computer Matching of Areas in Stereo Images.
Technical Report AIM-239, Stanford University, July, 1974.
- [Henderson, Miller, and Grosch 79]
Henderson, R. L., Miller, W. J., and Grosch, C. B.
Automatic Stereo Reconstruction of Man-made Targets.
Proc. SPIE 186(6):240-248, August, 1979.
- [Kanade 81] Kanade, T.
Recovery of the Three-dimensional Shape of an Object from a Single View.
Artificial Intelligence 17(1-3):409-460, August, 1981.
- [Kender 83] Kender, J. R.
Environmental Labelings in Low-Level Image Understanding.
Proc. IJCAI-83 :1104-1107, August, 1983.
- [Lowe and Binford 81]
Lowe, D. G., and Binford, T. O.
The Interpretation of Three-dimensional Structure from Image Curves.
Proc. IJCAI-81, August, 1981.
- [Lucas and Kanade 81]
Lucas, B. D., and Kanade, T.
An Iterative Image Registration Technique With an Application to Stereo Vision.
Proc. IJCAI-81 :674-679, August, 1981.

- [Martin and Aggarwal 83] Martin, W. N. and Aggarwal, J. K.
Volumetric Descriptions of Objects from Multiple Views.
IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-5(2):150-158, March, 1983.
- [McKeown 83] McKeown, D. M.
MAPS: The Organization of a Spatial Database System Using Imagery, Terrain, and Map Data.
Technical Report CMU-CS-83-136, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, July, 1983.
- [Moravec 80] Moravec, H.P.
Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.
Technical Report CMU-RI-TR-3, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, September, 1980.
- [Nevatia and Babu 80] Nevatia, R. and Babu, K.R.
Linear Feature Extraction and Description.
Computer Graphics and Image Processing 13:257-269, July, 1980.
- [Ohta and Kanade 83] Ohta, Y., and Kanade, T.
Stereo by Intra- and Inter-scanline Search Using Dynamic Programming.
Technical Report CMU-CS-83-162, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, October, 1983.
- [Potmesil 83] Potmesil, M.
Generating Models of Solid Objects by Matching 3D Surface Segments.
Proc. IJCAI-83 :1089-1093, August, 1983.
- [Requicha 80] Requicha, A. A. G.
Representations for Rigid Solids: Theory, Methods, and Systems.
Computing Surveys 12(4):437-464, December, 1980.
- [Rubin 80] Rubin, S.
Natural Scene Recognition Using Locus Search.
Computer Graphics and Image Processing 13:298-333, 1980.
- [Shafer and Kanade 82] Shafer, S. A., and Kanade, T.
Using Shadows in Finding Surface Orientations.
Technical Report CMU-CS-82-100, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, January, 1982.
- [Underwood 75] Underwood, S. A. and Coates, C. L.
Visual Learning from Multiple Views.
IEEE Transactions on Computers (6):651-661, June, 1975.