# Incremental Reconstruction of 3D Scenes from Multiple, Complex Images*

**Martin Herman**
*Robot Systems Division, National Bureau of Standards, Gaithersburg, MD 20899, U.S.A.*

**Takeo Kanade**
*Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A.*

Recommended by H.H. Nagel

ABSTRACT

*The 3D Mosaic system is a vision system that incrementally reconstructs complex 3D scenes from a sequence of images obtained from multiple viewpoints. 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 updating 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.*

* This research was sponsored partly by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539, and partly by the Air Force Office of Scientific Research under contract F49620-83-C-0100.

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, the Air Force Office of Scientific Research, or the US Government.

*Artificial Intelligence* **30** (1986) 289–341
0004-3702/86/$3.50 

*The model is represented as a graph in terms of symbolic primitives such as faces, edges, vertices, and their topology and geometry. This permits the representation of partially complete, planar-faced objects. Because incremental modifications to the model must be easy to perform, the model contains mechanisms to* (1) *add primitives in a manner such that constraints on geometry imposed by these additions are propagated throughout the model, and* (2) *modify and delete primitives if discrepancies arise between newly derived and current information. The model also contains mechanisms that permit the generation, addition, and deletion of hypotheses for parts of the scene for which there is little data.*

## 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. Our goal in developing the 3D Mosaic system has been to build a full vision system, that is, one that goes all the way from images to symbolic 3D descriptions. Further, we wanted to investigate this process in the context of complex scenes. The result is really a first pass at such a system, and provides us with a better understanding of the components required. 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. Finally, we describe the representation, construction, and updating of the scene model, along with the task-specific knowledge used here.

## 2. Description of 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 [7, 27]. 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 change 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, and (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 at acquiring 3D scene descriptions from multiple views have dealt with relatively simple scenes in controlled environments [2, 8, 9, 22, 25, 28]. This has led, in some cases, to only utilizing occluding contours in the image to form the 3D description [2, 8, 9, 22]. The work of Moravec [23] 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

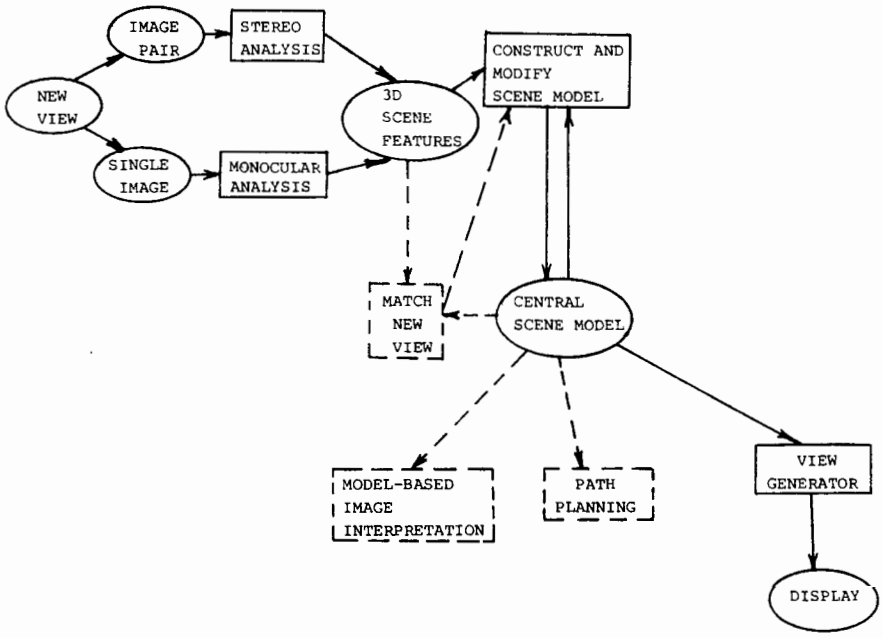


FIG. 1. 3D Mosaic flowchart, showing major modules (boxes) and data structures (ellipses). The dashed lines represent components that have not yet been implemented; the solid lines represent components already implemented.

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. 33. The scene model constructed from these wire frames is shown in Fig. 36. When the monocular analysis component is applied to the image in Fig. 11, the result is the set of wire frames in Fig. 23. These, in turn, are converted into the scene model in Fig. 37. Finally, the result of modifying the model in Fig. 36 with a new view is shown in Fig. 43.

### 3. Stereo Analysis

Most stereo matching methods involve matching low-level image features, such as image intensities [3, 14, 21, 24] or image edge points [3, 13, 24]. Points to be

matched may also be chosen as "interesting points," e.g., those with high variance in all directions [6, 23]. 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 [3, 5, 6, 13, 14, 21].

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 [19].

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 $Q_1$. 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

[1] For a different approach developed for the same domain, see [15].

[2] In 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 [5].

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

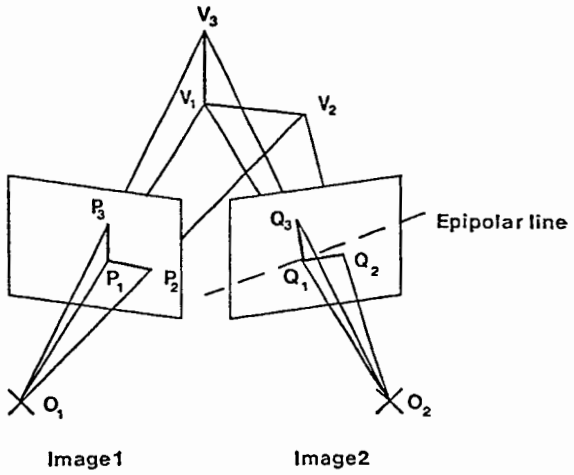


FIG. 2. For junction $P_3P_1P_2$, its apprearance in Image2 can be predicted as a function of position $Q_1$ along the epipolar line. The normal to plane $V_3V_1V_2$ must be known.

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.

Therefore, when an L-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.

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 $P_i$, $Q_i$, and $V_i$ correspond to positions of separate junctions and vertices. For example, if $P_1$ and $P_3$ are two separate junctions in Image1, 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.

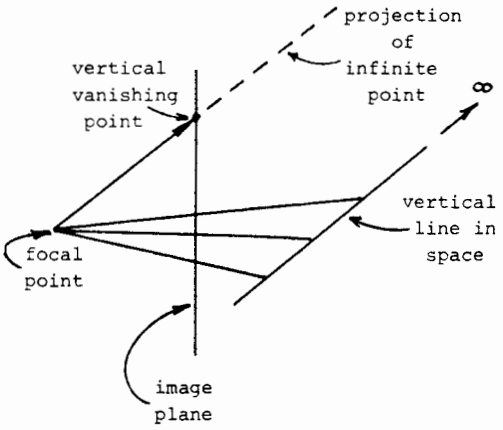


FIG. 3. The vector from the focal point to the vertical vanishing point is a 3-space vector in the vertical direction.

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 [4]. 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.

### 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.

The first few steps in the stereo analysis involve (1) extracting linear features, (2) extracting junctions, and (3) finding potential matches between the junctions in the two images. These steps are described in detail in [17]. Figure 5 shows junctions that have been found in the two images. Notice that many of these junctions correspond to building corners.

The step that involves finding potential junction matches uses the junction prediction technique described earlier. Each L-junction, for example, is initially assumed to lie on a horizontal scene plane. The shape and orientation of its corresponding junction in the other image can therefore be predicted. In this way, each L-junction in the first image is associated with a set of potentially matching junctions in the other image.

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

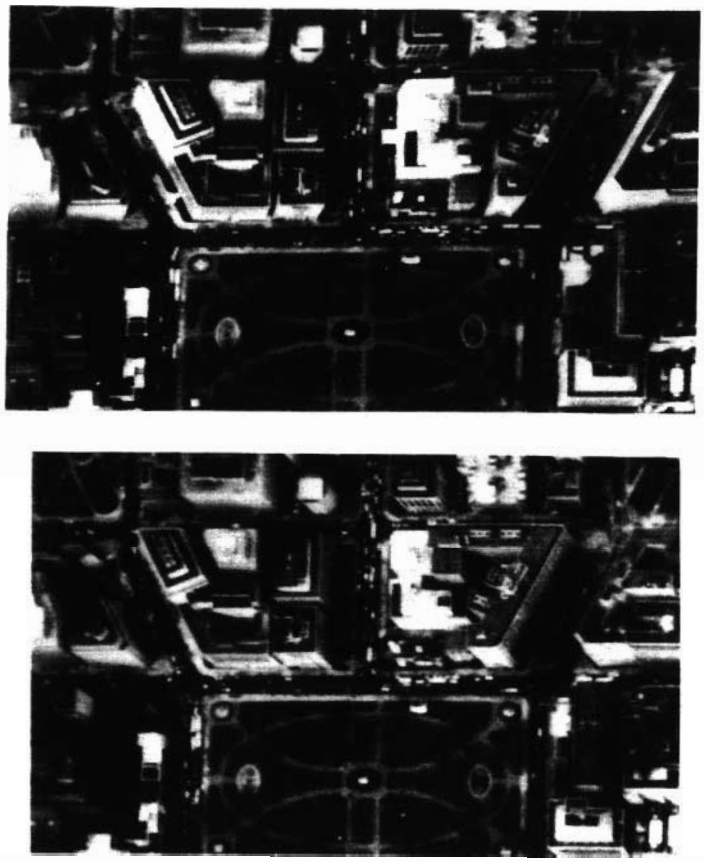

FIG. 4. Gray scale stereo images of a region of Washington, DC.

The next step is to find the best 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. 6, 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

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

Fig. 5. Result of classifying junctions extracted from the images in Fig. 4. Junctions are classified as L, A (arrow), F (fork), or T.

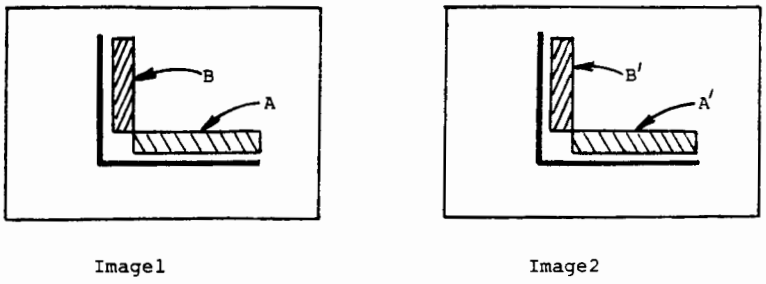


FIG. 6. Intensities of corresponding regions of L-junctions in the two images are used to compute the local matching cost.

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. 7, 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 [27], 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. 8, where each junction in (say) Image1 (i.e., $J$, $K$, $L$, . . .) is paired with each of its potential matches in Image2 (i.e., $J_i'$, $K_i'$, $L_i'$, . . .). 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.

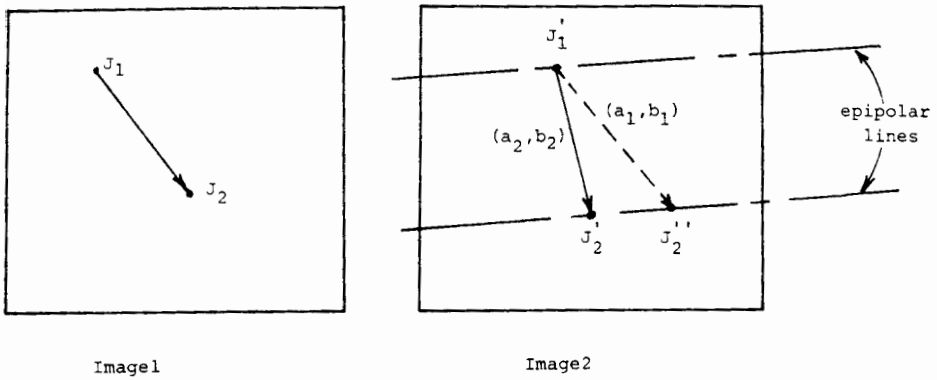


FIG. 7. Positional vectors of predicted and actual positions of two junction matches are used to compute the global matching cost.

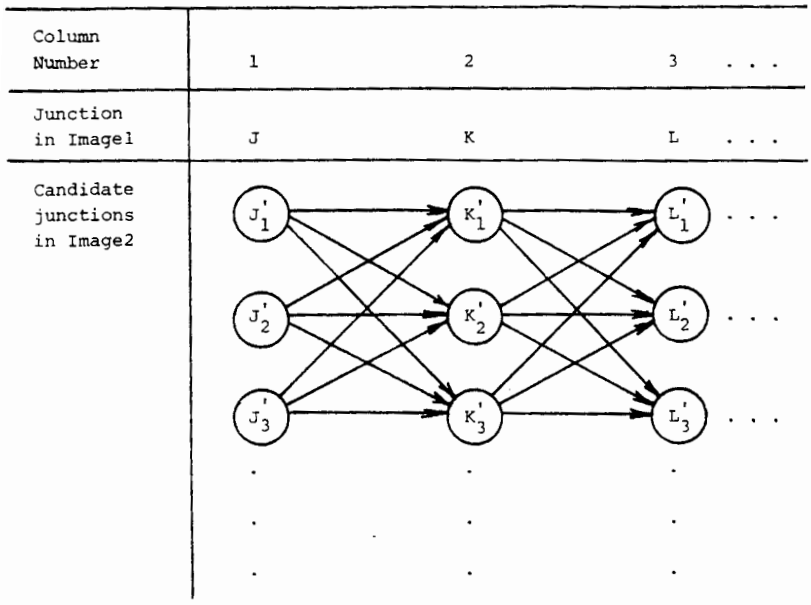


FIG. 8. Each column contains a junction from Image1 and its candidate matches from Image2. The candidates form the nodes of a network which is searched by a beam search.

In Fig. 8, 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 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. 8) 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 $K_j'$ 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 $K_j'$, and the local cost of node $K_j'$. Then add a link in the network between nodes $J_i'$ and $K_j'$.

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

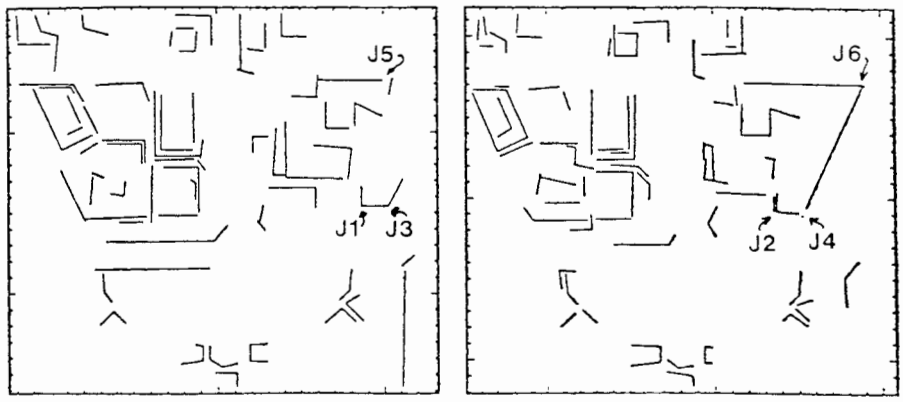


FIG. 9. Matches that have been found for the junction in Fig. 5. Actually, not all matches are correct. For example, although the junction matches $(J_1, J_2)$ and $(J_3, J_4)$ are correct, the match $(J_5, J_6)$ is incorrect.

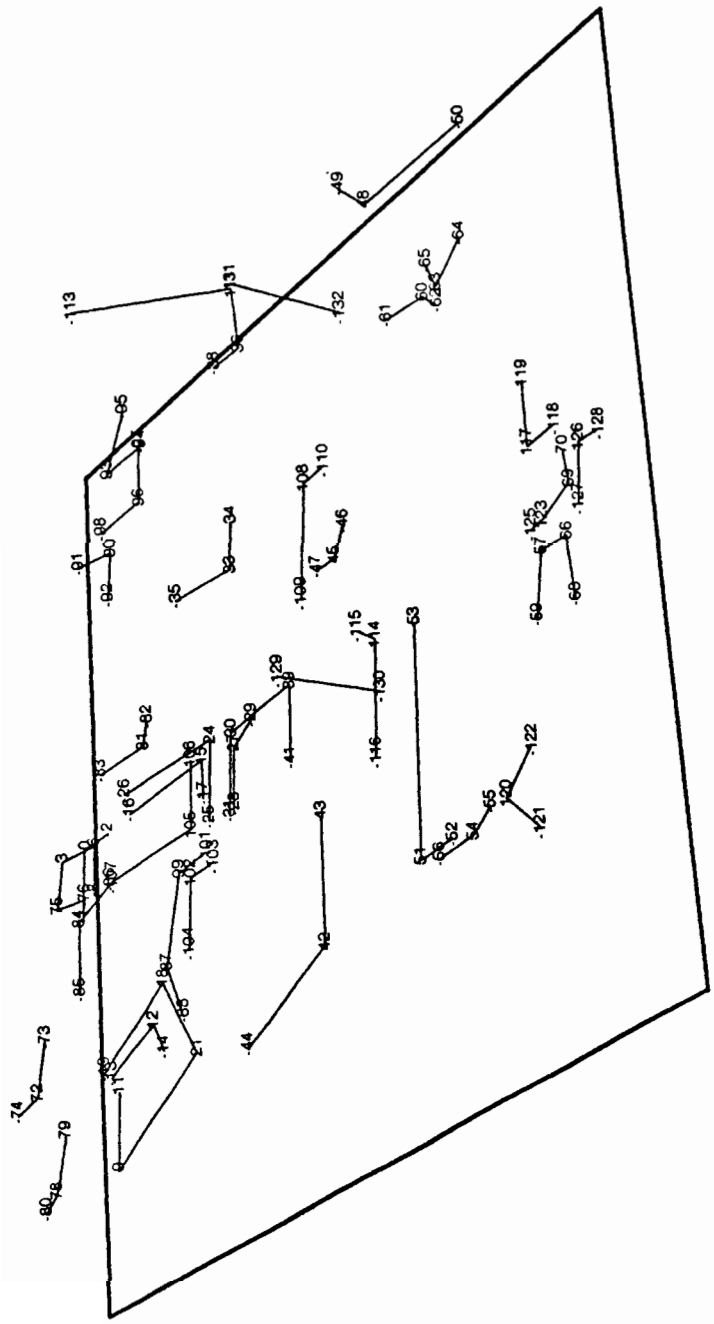

FIG. 10. Perspective view of 3D wire-frame description (i.e., 3D vertices and edges) derived from matches shown in Fig. 9. The numbers represent unique identifiers for the end points of the edges.

path at column $k + j$ if it were extended that far. The results of this procedure are displayed in Fig. 9, which shows junctions in one image that have matches in the other image.

The final steps in the stereo analysis involve (1) finding lines in the images that might be the third leg of matched junctions and that might represent scene edges perpendicular to the ground plane, and (2) using triangulation to derive 3D coordinates of vertices and equations of edges. Figure 10 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. 11. 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

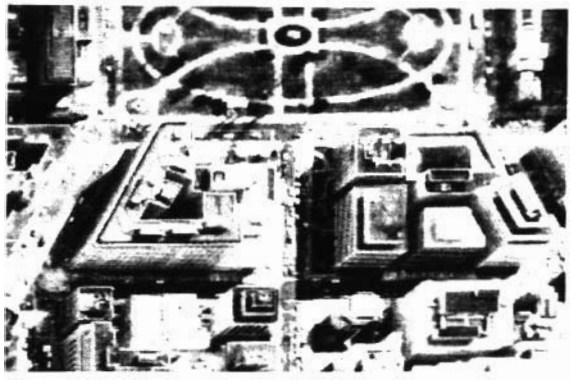

FIG. 11. Aerial photograph showing part of Washington, DC. This is a different view of the same scene as in Fig. 4.

the same as that used during stereo analysis (as previously described). Thinned edge points are shown in Fig. 12, and the result of extracting lines and junctions is shown in Fig. 13.

*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. Second, the two junctions must appear to be connected by line segments in the line image.

The first step involves finding all pairs of junctions such that one has a leg pointing at the other, and proceeds as follows. First, if two junctions share the same leg, they are connected. Next, for each leg of each junction $J_i$, a thin rectangular window is located in the direction along the leg (Fig.14). Of the junctions within this window and within an angle $\alpha$ from the direction of the leg, the one closest to $J_i$ is retained as a candidate for being connected to $J_i$. Figure 15 shows a graph with all candidate connections drawn.

The second step involves determining which connections shown in Fig. 15 appear as connections in the line image (Fig. 13). For each pair of connected

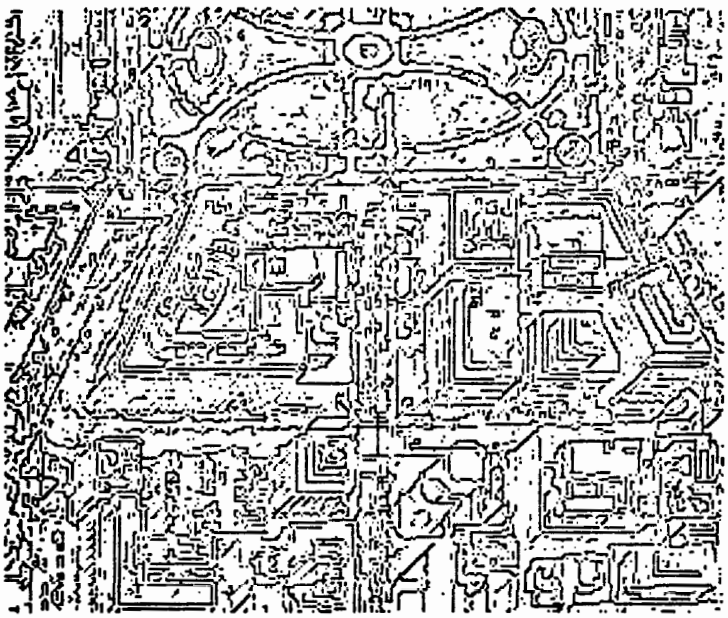

FIG. 12. Result of thinning the edges obtained by applying a Sobel operator to the image of Fig. 11.

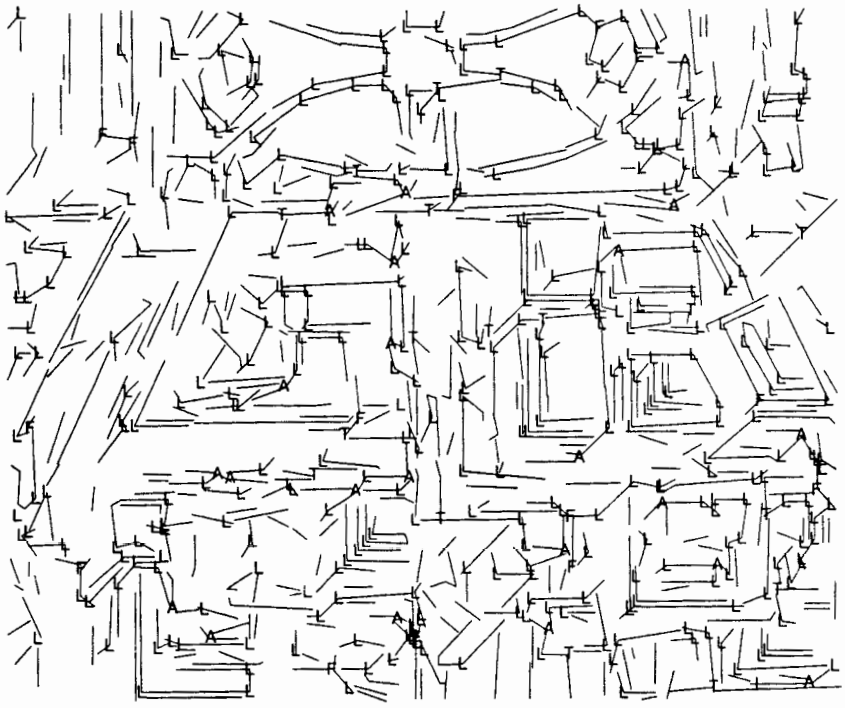

FIG. 13. Lines fitted to the edge points of Fig. 12 after they are linked. Junctions in the image are classified as L, A (arrow), F (fork), or T.

junctions $J_i$ and $J_k$ (Fig. 16), 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 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. 17. 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.

The junction legs originally extracted in the junction finding step are then added to the result of Fig. 17, and extraneous legs are deleted. The final connected structures are displayed in Fig. 18.

*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

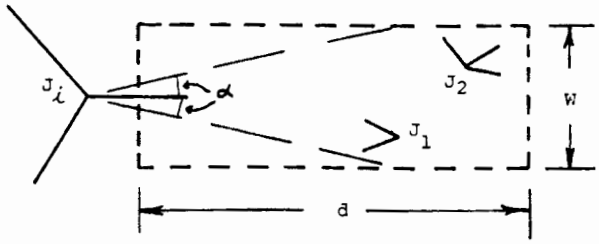


FIG. 14. The closest junction to $J_i$ within the thin rectangular window of length $d$ and height $W$, and within the angle $2\alpha$, is a candidate for being connected to $J_i$.

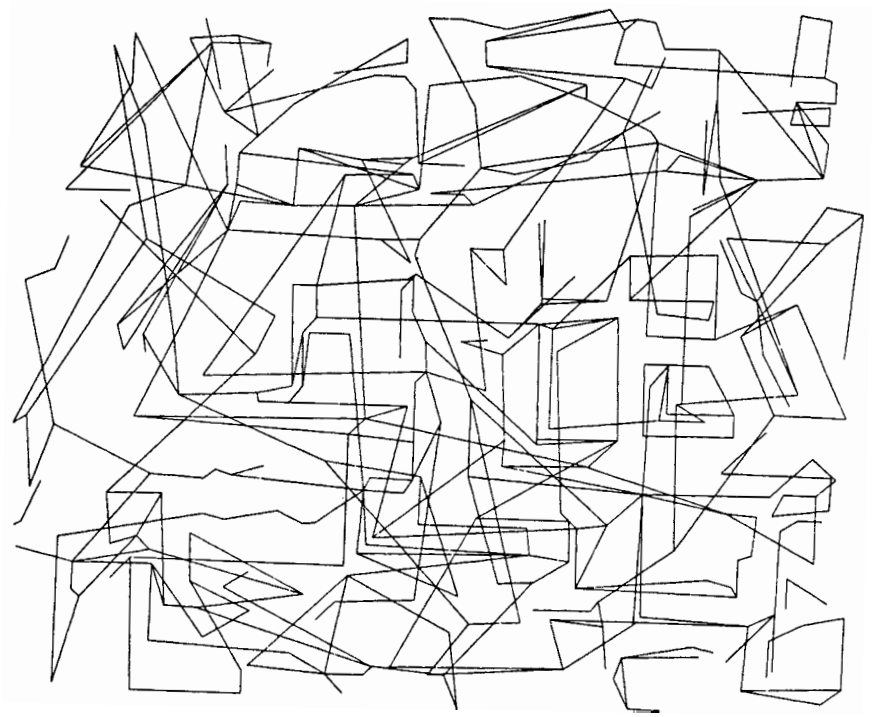

FIG. 15. Each line represents a possible connection between the junctions at its two end points. Each end point corresponds to a junction in Fig. 13.

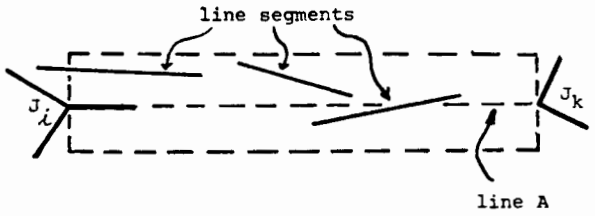


FIG. 16. All line segments within the thin rectangular window connecting junctions $J_i$ and $J_k$ are projected onto line $A$ to determine the amount of coverage.

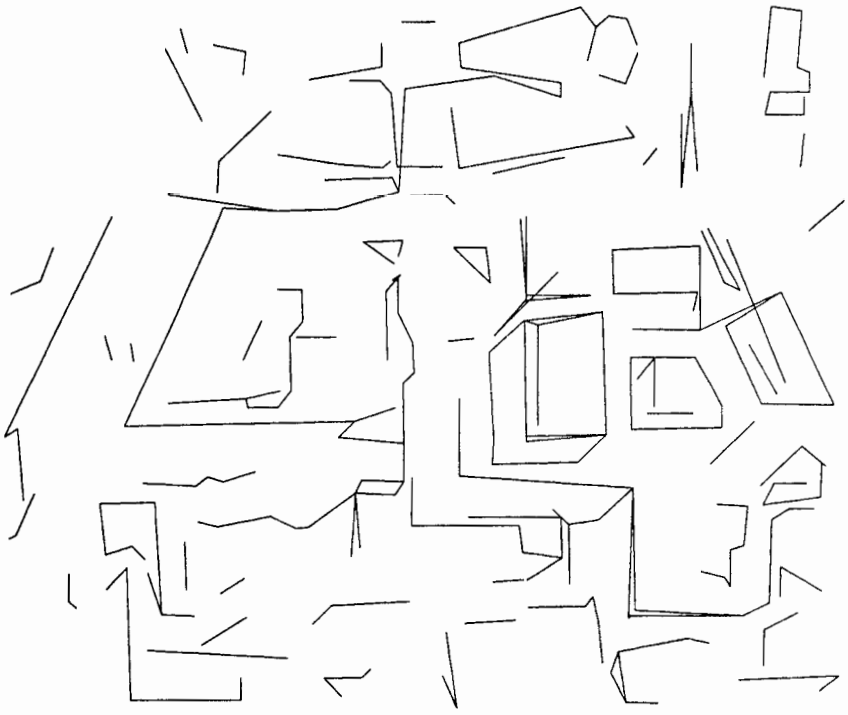

FIG. 17. Result of pruning the junction connections in Fig. 15 by determining whether segments in Fig. 13 adequately cover the area between each pair of connected junctions.

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 [19]. Next, we use the position of the vertical vanishing point to

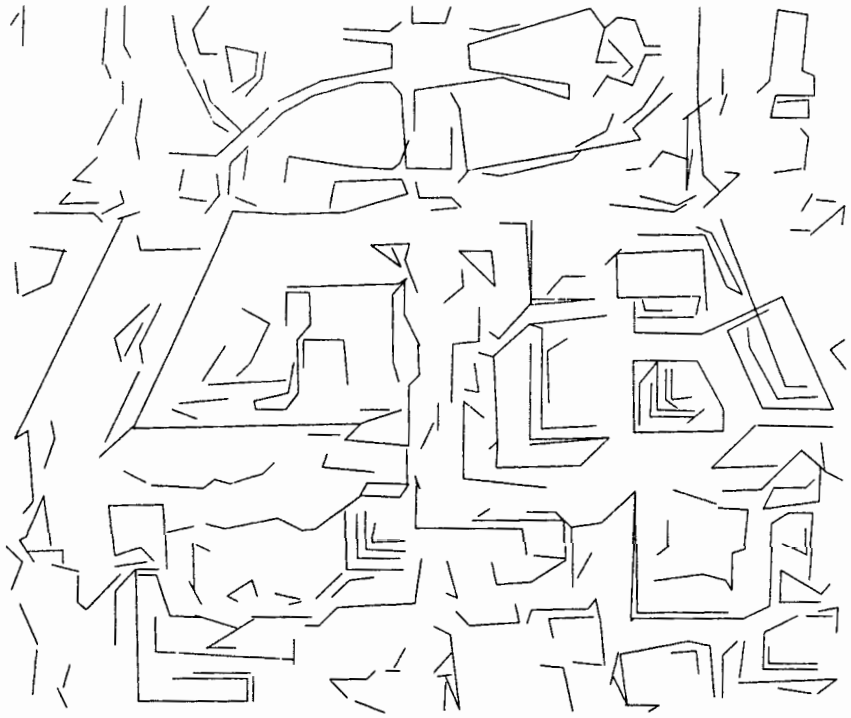

FIG. 18. Result of adding to Fig. 17 the junction legs that were originally extracted in the junction finding step, and then deleting extraneous legs.

calculate the vector in the vertical direction, as described previously. Let us now consider how to recover the 3D configuration of the junction $p_1p_2p_3p_4$ in Fig. 19. Suppose that line $p_2p_4$ has been labeled "vertical" and lines $p_1p_2$ and $p_2p_3$ have been labeled "horizontal." Let $\boldsymbol{u}$ 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 $v_1v_2v_3$ can now be established, since it contains $v_2$ and its normal vector is $\boldsymbol{u}$. 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 $p_4$ with the line through $v_2$ along the vector $\boldsymbol{u}$.

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. 13) because the relative heights

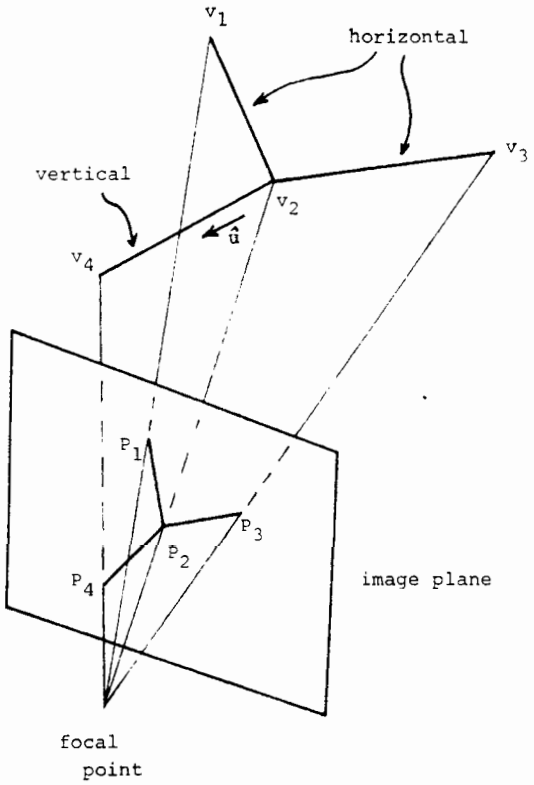


FIG. 19. The 3D configuration of the junction $p_1p_2p_3p_4$ can be recovered under assumptions explained in the text.

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. 18, since the heights of the vertices within each structure can be related. To see how this is done, consider the example in Fig. 20, 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.

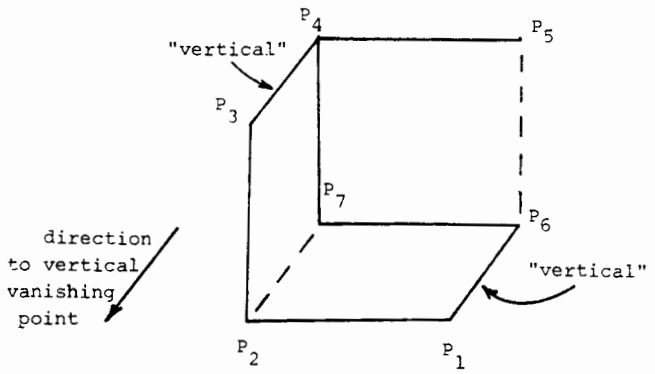


FIG. 20. The solid lines represent a connected 2D structure. The dashed lines are for the reader's convenience to make the 3D shape more apparent.

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 $\boldsymbol{u}$ 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. 20), 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. 20 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. Figure 21 depicts a perspective view of the 3D wire frames obtained in this manner.

There are many structures in Fig. 18 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. 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 [20] and in fact is a restricted version of the parallel line rule [18] which states that parallel lines in the image often arise from

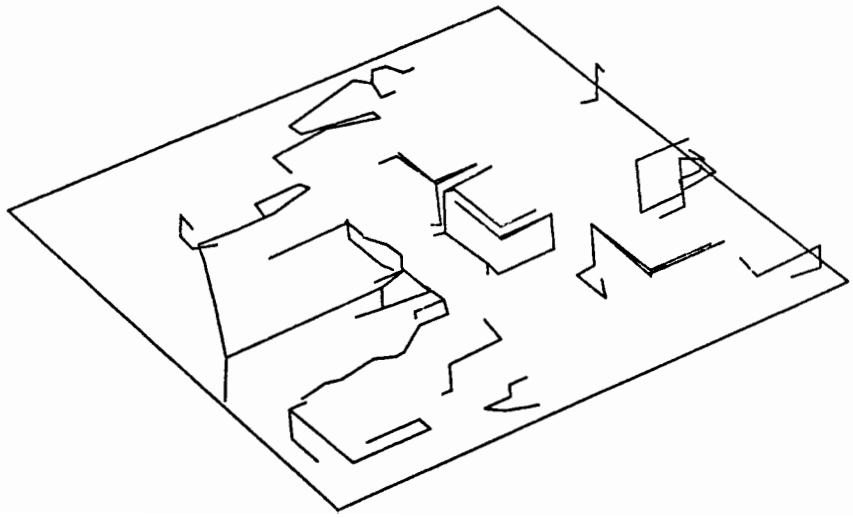

FIG. 21. Perspective view of 3D wire frames generated from Fig. 18 using the method of finding junctions arising from vertices lying on the ground plane.

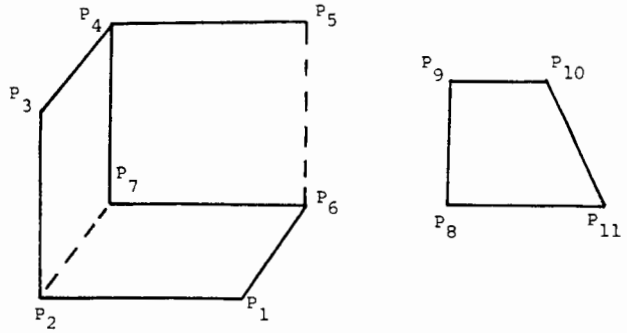


FIG. 22. If the 3D configuration of the structure on the left has been determined, the relative 3D position of the structure on the right may also be determined because lines $p_6p_7$ and $p_8p_{11}$ are aligned.

parallel lines in 3-space. To see how this rule is used, consider Fig. 22. 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

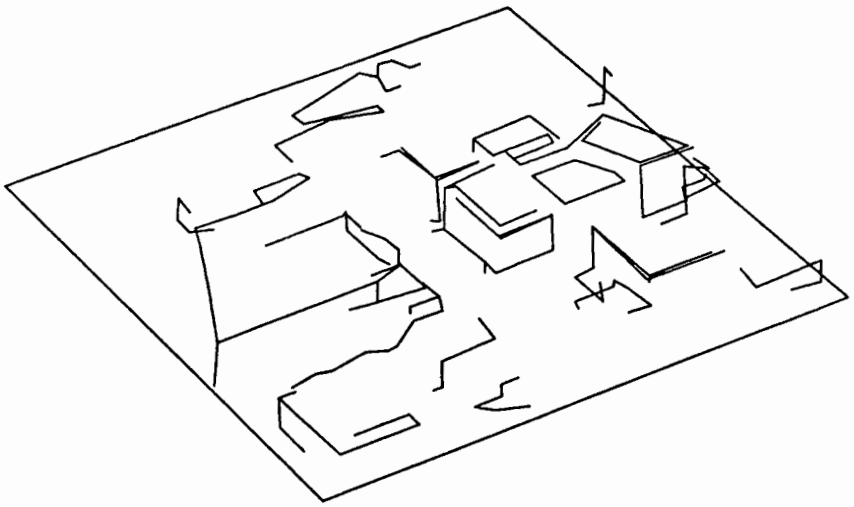

FIG. 23. Perspective view of final set of 3D wire frames generated from Fig. 18.

previously. Note that all 3D positions are functions of the parameter $d$, which is arbitrarily chosen for the equation of the ground plane.

Figure 23 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.

## 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 [1, 26]. 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.

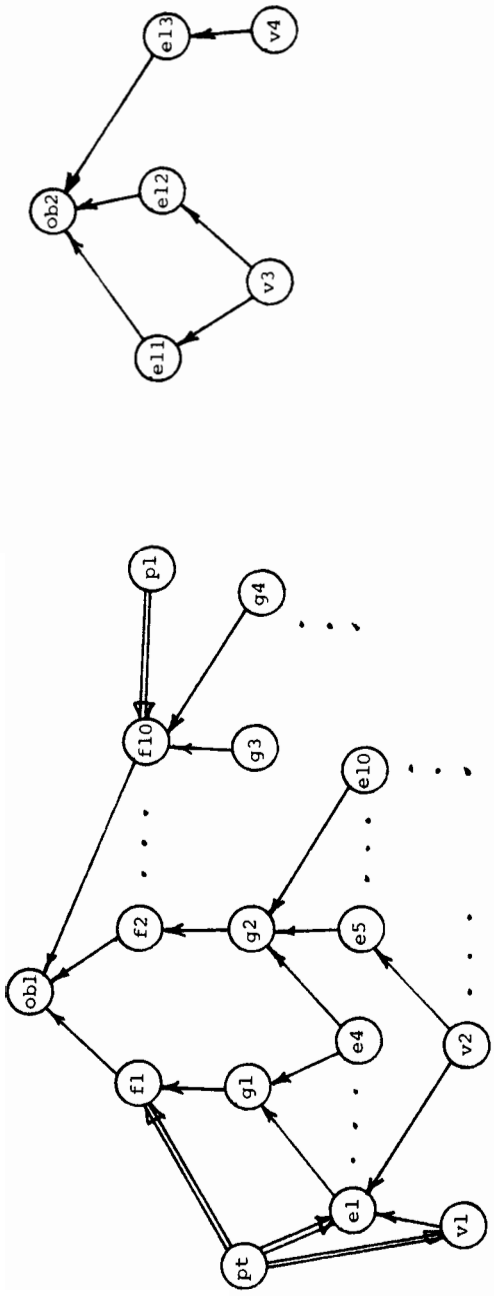


FIG. 24. Simple example of a structure graph consisting of two objects, ob1 and ob2. Double line arrows represent geometric constraint links, and single line arrows represent part-of links.

(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.

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 and deletion 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 [1, 12]. 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, vertex, and point 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. An edge may actually be partially confirmed and partially unconfirmed. For example, a confirmed edge, extracted from the image, may later be hypothesized to extend further in length. In this case, the confirmed portion of the edge will lie between two confirmed points, while the unconfirmed portion will lie between a confirmed and an unconfirmed point.

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.

Figure 24 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.

## 6. Modifications to the 3D Scene 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. These modifications, to be described next, are the basic processes used in constructing the structure graph representation as described in Section 7, and in modifying the structure graph by incorporating a new view, as described in Section 8.

### 6.1. Propagation due to geometric modifications

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). Object and edge group nodes may not be geometrically constrained directly. Figure 25 shows how a constraint on one node may propagate to others. The arrows in the figure indicate the direction of propagation. The tail of an arrow indicates the source constraint; the head indicates the constraint implied by the source constraint.

We see in Fig. 25 that 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 must also constrain all faces containing that edge. Note that when a point constrains an edge, we assume that no constraint is implied for arbitrary vertices that are part of that edge, since the point need not lie on any of these vertices. In one sense, the point may be considered to constrain such vertices since they must lie on a line going through the point. This constraint, however, is not useful until another constraint on the line is derived, such as another point that lies on the edge. In this case, our system generates the equation of the line that constrains the edge and propagates the line constraint down to the vertex, as explained in the next paragraph. A more direct and useful constraint is thus imposed on the vertex.

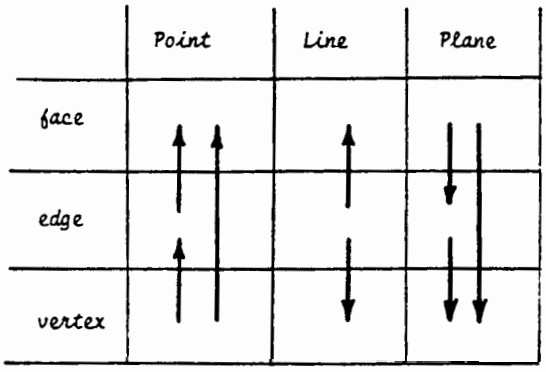


FIG. 25. Rectangular boxes indicate geometric constraints on topological modes. Arrows indicate direction of propagation of constraints.

Similarly, when a point constrains a face, no useful constraint is implied for arbitrary edges or vertices that are part of the face.

As indicated in Fig. 25, line constraints propagate outward. A line that constrains an edge must also constrain all faces containing the edge and all vertices that are part of the edge. Finally, plane constraints propagate downward. A plane that constrains a face must also constrain all edges and vertices that are part of the face. Similarly, a plane that constrains an edge must also constrain all vertices that are part of the edge. Whenever a geometric constraint link is added, propagation occurs as indicated in Fig. 25.

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 [10], 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. To see this, consider Fig. 26. Let $\{x_1, x_2, \ldots, x_m\}$ be a set of assertions, each of which independently implies the assertion $y$. The assertion $(y \wedge v_1 \wedge v_2 \wedge \cdots)$, in turn, implies each assertion in the set $\{z_1, z_2, \ldots, z_n\}$. Furthermore, for each $i$, $z_i$ is independently implied by each assertion in the set $\{w_{ij}\}$. Now suppose the assertion $y$ is deleted, i.e., it is declared false. Then:

(1) Since each assertion $z_i$ depends on the truth of $y$, $z_i$ is deleted unless it has other support $w_{ij}$.

(2) All assertions $x_i$ are made false. None of them can be true, for if one were, $y$ must be true. Since $x_i$ may consist of a conjunction of assertions, at least one of them is deleted to make $x_i$ false.

We obtain assertions that imply a given assertion by following backwards along the arrows in Fig. 25, and we obtain assertions implied by a given assertion by following forward along the arrows.

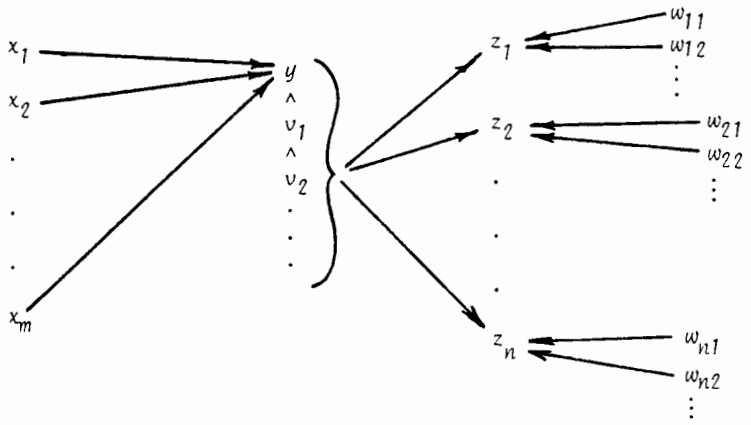


FIG. 26. The assertion $y$ is independently implied by each $x_i$. Each assertion $z_i$ is independently implied by $(y \wedge v_1 \wedge v_2 \wedge \cdots)$ and $w_{ij}$.

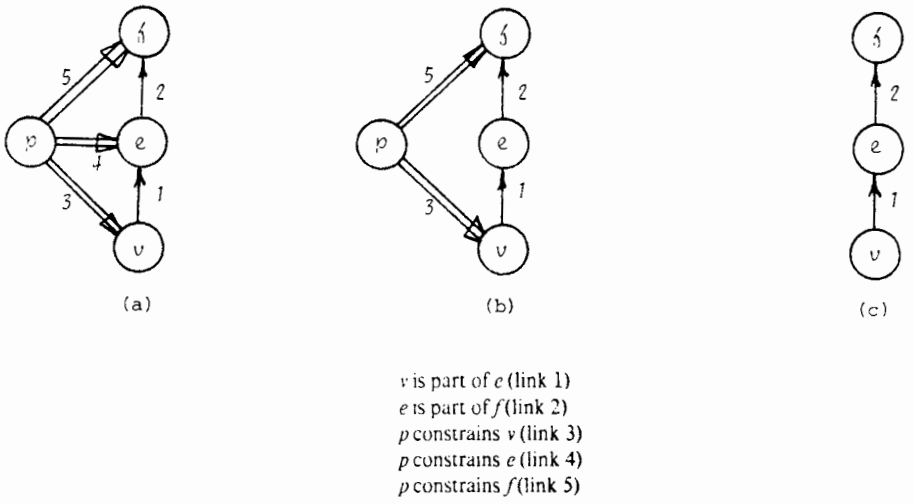


FIG. 27. (a) Initial structure graph. (b) Link 4 is deleted. (c) Resulting structure graph after effects of deletion have been propagated.

Consider the simple example in Fig. 27(a), 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. 27(b)), that is, the 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. 25 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. 27(b), 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. 27(b). 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. 25 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. 27(b), this corresponds to the assertion "$p$ constrains $f$," which is link 5. This link should therefore be deleted since it has no other support. One possible source of other support is external to the structure graph. Link 5 may have been derived, for example, directly from image data, rather than through structure graph propagation. We rule out the possibility that links 4 and 5 are unrelated, and thus delete link 5. The resulting structure graph is depicted in Fig. 27(c).

### 6.2. Propagation due to topological modifications

When a topological part-of link between two topological nodes is added or deleted, the effects are propagated to other parts of the structure graph. In the following, we will consider both geometric and topological effects.

#### 6.2.1. *Geometric effects*

When a topological part-of link is added between two topological nodes, the geometric constraints on each node must be propagated to the other node in accordance with the chart in Fig. 25. There are three main cases to consider: (1) adding a part-of link between a vertex and edge node, (2) between an edge and face node, and (3) between a vertex and face node. These three cases are explicitly covered in Fig. 25. The remaining cases fall into two classes: (a) adding a part-of link between some topological node and an object node, and (b) between some topological node and an edge-group node. Since object nodes cannot be geometrically constrained directly, actions in class (a) have no geometric effects. Since geometric constraints can be propagated through edge-group nodes, actions in class (b) do have geometric effects. These effects, however, can be reduced to the three cases above, as explained in the next paragraph.

Consider the example of adding a part-of link between an edge node $E$ and a face node $F$. From Fig. 25, we see that all point and line constraints on $E$ must be propagated to $F$, while all plane constraints on $F$ must be propaged to $E$. Plane constraints propagated to $E$ are, in turn, propagated to vertices of $E$. As another example, consider adding a part-of link between an edge-group node $G$ and a face node $F$. This situation results in the same geometric propagation as the following two cases: (1) add a part-of link from each edge of $G$ to $F$, and (2) from each vertex of $G$ to $F$. Similar rules can be established for the other two situations involving edge-group nodes (i.e., adding a link between a vertex and edge-group node, and between an edge and edge-group node).

When a part-of link between two topological nodes is deleted, an attempt is made to nullify any geometric propagation that occurred through the link. This is done by deleting, from the two nodes connected by the link, all geometric constraints that have propagated through the link. The effects of deleting these geometric constraint links are, in turn, propagated to the rest of the graph in the manner described in the previous section.

As an example, consider deleting a part-of link between an edge node $E$ and a face node $F$. As seen in Fig. 25, all point and line constraints on $F$ that also constrain $E$ were either (1) propagated up from $E$, (2) propagated up from another edge or vertex of $F$, or (3) derived from an external source. We rule out the possibility that the same constraints on $E$ and $F$ are unrelated, thus ruling out the external source. Therefore, points and lines that constrain both $F$ and $E$, but do not also constrain another edge or vertex of $F$, are deleted from

$F$ since we just cut off the only path through which they could have propagated to $F$. The effects of deleting the point and line constraints from $F$ are, in turn, propagated to the rest of the graph. Similarly, all plane constraints on $E$ that also constrain $F$ are deleted from $E$ unless they also constrain another face that contains $E$ (which would be unusual). The effects of deleting plane constraints from $E$ are then propagated.

An example of a link with more than one source of support is shown in Fig. 28(a). Suppose the part-of link between $e_1$ and $f$, link 4, is deleted (Fig. 28(b)). According to the chart in Fig. 25, link 8 is a candidate for deletion since the point node $p$ constrains both $e_1$ and $f$. However, since $p$ also constrains the edge $e_2$, which is part of $f$, link 8 is still valid.

### 6.2.2. *Topological effects*

A topological modification sometimes implies topological changes elsewhere in the structure graph. This is best illustrated through an example. Figure 29(a) shows the graph representing the situation in Fig. 29(b). The edge $e$ has two vertices, $v_1$ and $v_2$, and $v_1$ is known to be part of the face $f$. Now suppose a part-of link is added between $v_2$ and $f$ (link 4 in Fig. 29(c)). Since both vertices of $e$ are now part of $f$, $e$ must also be part of $f$, as shown in Fig. 29(d). Therefore link 5 in Fig. 29(c) is added.

Another kind of topological effect results from the desire to eliminate redundant part-of links. Part-of links serve as paths in the structure graph along which effects of geometric changes are propagated. In order to simplify this process, the number of paths between each pair of topological nodes is minimized using the following rule: Two topological nodes may not be directly

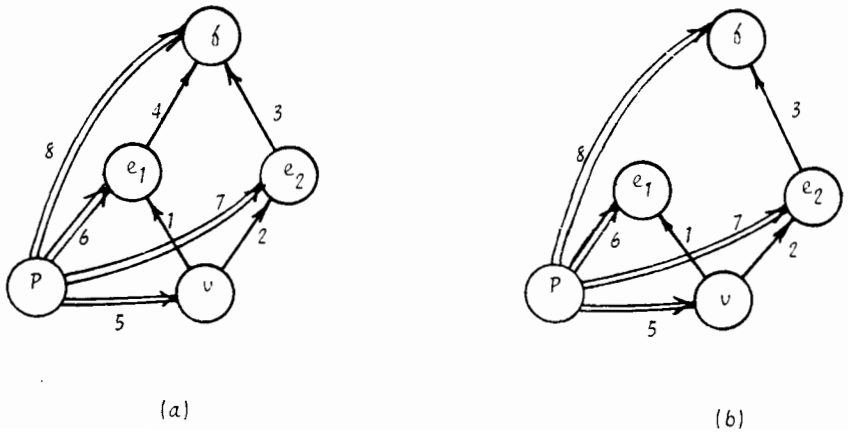


FIG. 28. Example of a link with more than one source of support. (a) Initial structure graph. (b) Link 4 is deleted, but link 8 remains because of support from links 3 and 7.

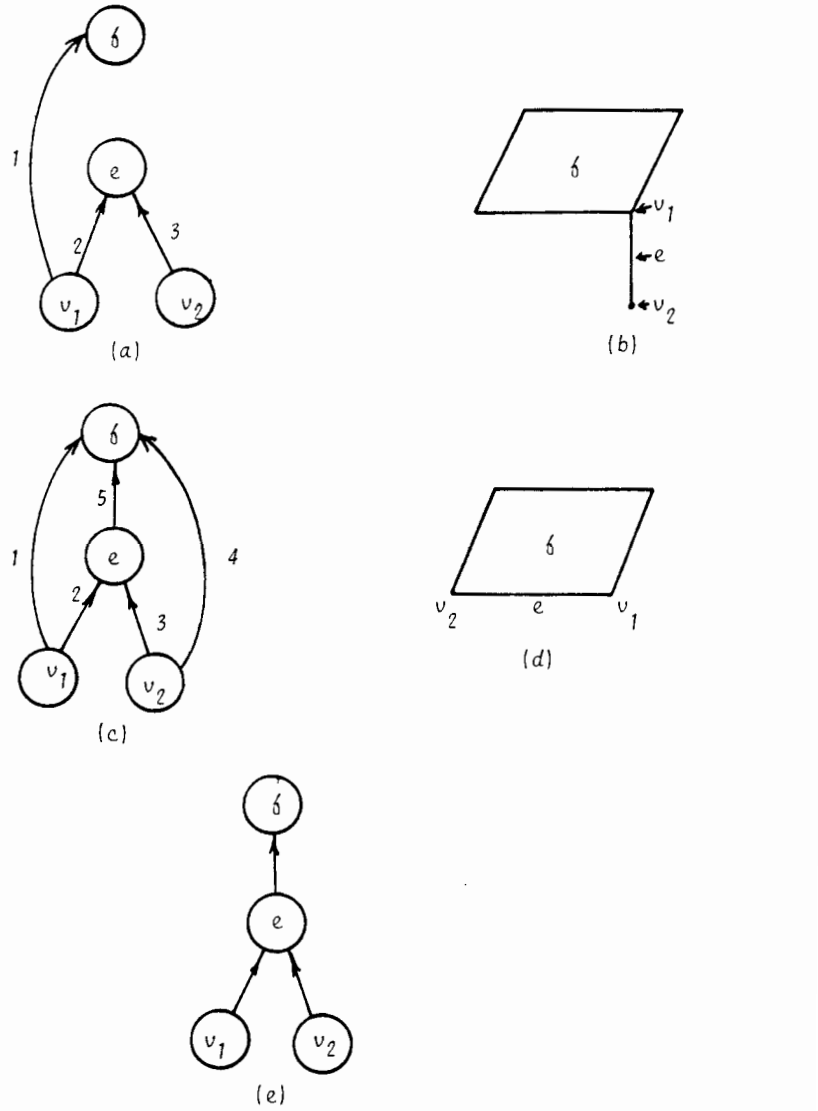


FIG. 29. Topological propagation. (a) and (b) Initial situation. (c) and (d) Link 4 is added, resulting in addition of link 5. (e) Redundant links are eliminated.

connected (i.e., by means of a part-of link) if they are also connected through one or more intermediate topological nodes. For example, suppose a part-of link is added between the edge node $e$ and the face node $f$ in Fig. 30(a). To avoid redundancy, all links connecting vertex nodes of $e$ and the node $f$ (link 1 in Fig. 30(a)) and vertex nodes of $e$ and object nodes containing $f$ (link 2) are

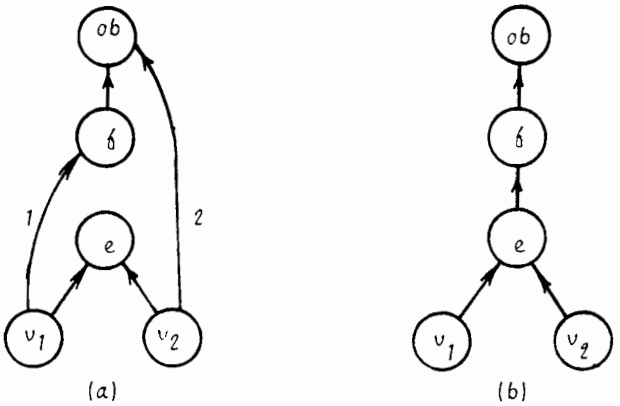


FIG. 30. (a) Initial configuration. (b) When a link is added from $e$ to $f$, links 1 and 2 are deleted to eliminate redundancy.

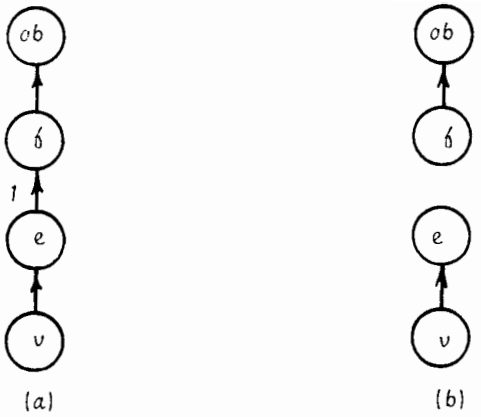


FIG. 31. (a) Initial configuration. (b) Final result after link 1 is deleted.

deleted. In addition, if there were any links between $e$ and object nodes containing $f$, they would also be deleted. The final configuration is shown in Fig. 30(b). In the example of Fig. 29, the graph in (c) has redundant links. Links 1 and 4 are therefore deleted, resulting in the graph of Fig. 29(e).

Although adding a part-of link can result in topological changes elsewhere in the graph, deleting a part-of link does not change the topology anywhere else. No attempt is made to recover previous states of topological connections. Figure 31(b) shows the result of deleting link 1 from the graph in Fig. 31(a). This technique seems to work well in our experiments.

## 7. Constructing and Updating the 3D Scene Model

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 representing 3D vertices and edges corresponding to portions of boundaries of objects in the scene. The goal of the updating process is to merge the wire-frame description with the current model. In general, this process will result in a partial 3D model which may consist of surfaces at some places but only portions of boundaries at other places. This partial 3D model must then be converted into a full surface-based description by hypothesizing new vertices, edges, and faces. Our current techniques for making such hypotheses exploit task-specific knowledge that falls into two categories: (1) knowledge of planar-faced objects, and (2) knowledge of urban scenes.

Both the wire frames and scene models are represented by structure graphs. The wire-frame description extracted from the first view forms the initial state of the scene model, and all of its edges, vertices, and points are tagged as confirmed. This wire-frame model is then converted into a full surface-based model using task-specific knowledge. All elements of the model that were not present in the initial state are hypothesized and tagged as unconfirmed.

When a wire-frame description is extracted from a new view, all of its edges, vertices, and points are tagged as confirmed. This description is then matched to the current model (in order to find corresponding elements in the two and the scale and coordinate transformation from one to the other) and merged with the current model. In the merging process, confirmed elements in the wire frames and model that match are "averaged' together, resulting in new confirmed elements. The parts of the wire frames that have no match in the model are then added to the model. Hypothesized elements in the model that are no longer consistent with confirmed parts are deleted. At this point, task-specific knowledge is again used to fill out the model and to form a full surface-based description.

### 7.1. Knowledge of planar-faced objects

Since the structure graph has been designed for scenes that can be modelled as collections of planar-faced objects, knowledge of such objects is inherent in the representation and propagation rules, as described above. We now discuss how knowledge of such objects is used to construct a scene model from wire frames.

When new wire-frame information (derived either from the first or a subsequent view) is added to the model, many object descriptions will be incomplete. A goal of the model construction process, of course, is to complete these object descriptions using task-specific knowledge. The notion of an object description being complete is best expressed in the context of the structure graph. An object node in the structure graph is considered complete if it meets certain requirements, which may be expressed in terms of complete

nodes contained by the object node. Each type of node in the graph, therefore, must meet certain requirements to be considered complete. Even though these requirements are only implicitly followed during the model construction process, it is useful to state them explicitly.

(1) An *object node* is complete if it is closed, i.e., each edge node of the object is part of two face nodes, both of which are complete.

(2) A *face node* is complete if it is constrained by a plane node and contains one or more complete edge-group nodes. One of these edge-group nodes must represent a bounding ring of edges on the face. The other, optional edge-group nodes represent inner edge rings, which would be holes in the face. In addition, each edge node of the face must be part of an edge group of the face.

(3) An *edge-group node* is complete if it contains a single, connected, closed ring of complete edges on a face.

(4) An *edge node* is complete if it is constrained by a line node and contains two complete vertex nodes.

(5) A *vertex node* is complete if it is constrained by a point node.

The following techniques applicable to planar-faced objects are used in constructing the model (see [17] for more details): (1) combining edges (Fig. 32 and $E_1$ and $E_2$ in Fig. 33), (2) generating a partial face, called a *web* face, for each adjacent pair of legs ordered around a vertex (Fig. 34(a)), (3) merging partial faces, and (4) finding and constructing holes in faces (Fig. 34(b)).

The procedure that merges partial faces distinguishes those that touch each other from those that do not. Two partial faces that touch each other (e.g., Fig. 34(c), and $F_1$ and $F_2$ in Fig. 33) 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.

The procedure for merging two touching faces provides a good example of how the basic modification techniques described in Section 6 are used to construct the structure graph representation. The merging of two touching

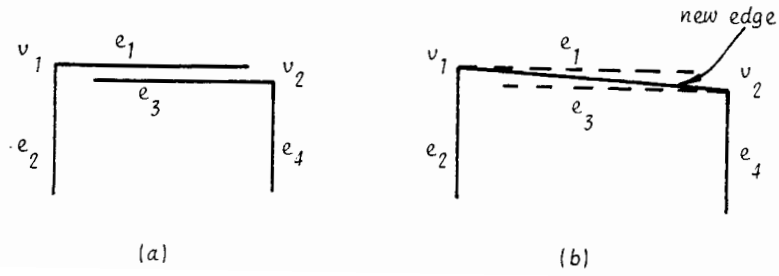


FIG. 32. Combining edges. (a) Edges $e_1$ and $e_3$ are very close to each other, and each has a confirmed vertex ($v_1$ and $v_2$, respectively). These vertices are on opposite ends of each other. (b) The new edge is shown as the result of merging $e_1$ and $e_3$.

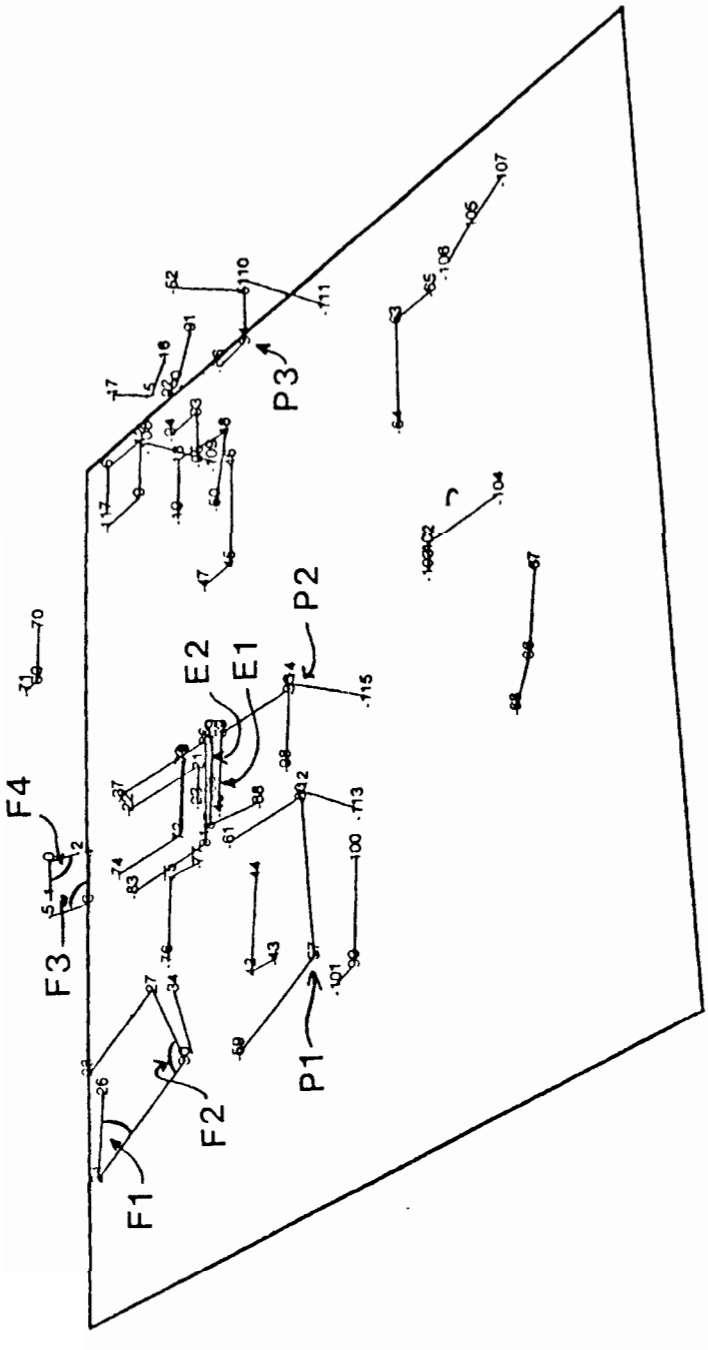


Fig. 33. Perspective view of 3D vertices and edges extracted from stereo pair in Fig. 4. This version is different from the one shown in Fig. 10.

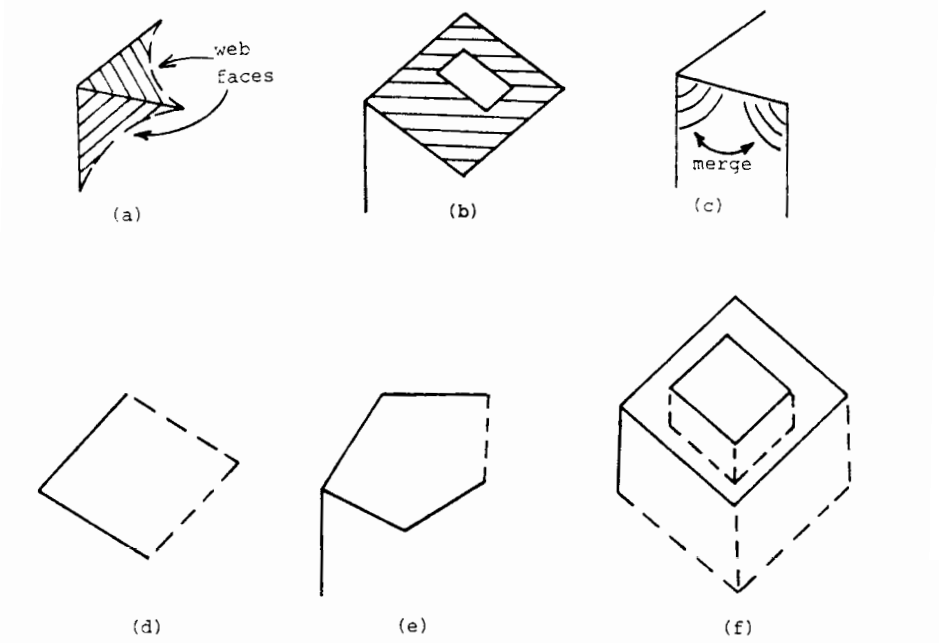


FIG. 34. Obtaining a surface-based description from wire frames.

faces $F_1$ and $F_2$ involves (1) finding the two edge groups $G_1$ of $F_1$ and $G_2$ of $F_2$ that contain the shared edge, (2) subtracting edges and vertices from $G_1$ (i.e., deleting part-of links in the structure graph) and adding them to $G_2$ (adding appropriate part-of links), (3) subtracting edge groups, edges, vertices, lines, and points from $F_1$ and adding them to $F_2$, and (4) recalculating the plane equation of $F_2$ as a least-squares fit to all the points now constraining $F_2$. As elements are added or subtracted from the structure graph, their effects are propagated to other parts of the graph, as described in Section 6.

Two partial faces that do not touch each other (e.g., Fig. 35(a), and $F_3$ and $F_4$ in Fig. 33) 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 (in Fig. 35(a), $p_1$ uniquely matches with $p_3$, and $p_2$ with $p_4$), and (3) the planes of the faces are nearly parallel and very close to each other. When merging two faces that do not touch, the two edges on which each matching pair of end points lie are intersected. The intersection points form two new vertices on the resulting face (vertices $v_1$ and $v_2$ in Fig. 35(b)). Notice in Fig. 35(b) that the edge $e_1$ has been shortened in the process, while the other edges have been extended.

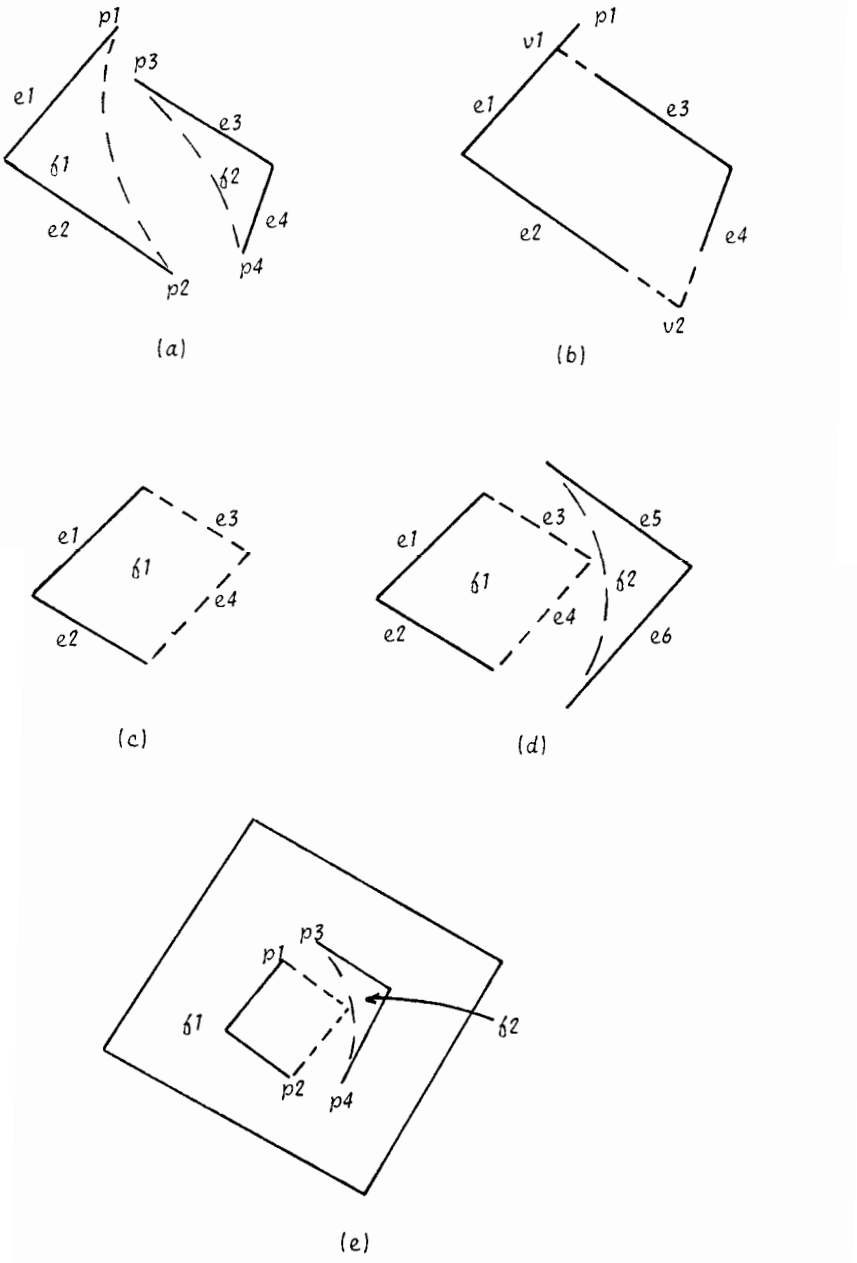


FIG. 35. Merging of nontouching faces. (a) $f_1$ and $f_2$ satisfy the conditions for merging. (b) Result of merging $f_1$ and $f_2$. (c) and (d) The complete face $f_1$ is merged with the partial face $f_2$. (e) The complete face $f_1$, which contains a hole, is merged with the partial face $f_2$.

Up till now, we have only discussed the merging of partial faces. However, if the confirmed parts of two faces, each of which may be partial or complete, satisfy the conditions for merging nontouching faces, then the faces may be merged. For example, suppose the face $f_1$ in Fig. 35(c) contains the confirmed edges $e_1$ and $e_2$ and the hypothesized edges $e_3$ and $e_4$. Now suppose that the web face $f_2$ in Fig. 35(d) is new information that becomes available, say, from a new view. The confirmed parts of $f_1$ may then be merged with $f_2$ if they satisfy the conditions for merging. In the process, hypothesized parts of $f_1$ must be deleted. The mechanisms for doing this will be discussed later.

Another interesting example is depicted in Fig. 35(e), whose situation is similar to that in Fig. 35(d) except that the web face $f_2$ is merged with confirmed parts of the face $f_1$, which has a hole in it. Notice that the condition that the confirmed parts of each face must have two end points which are uniquely matched to those of the other face is satisfied by $p_1$ and $p_3$, and by $p_2$ and $p_4$. As a result of merging, $f_2$ aids in completing the boundary of the hole in $f_1$.

### 7.2. Knowledge of urban scenes

Because the wire-frame data extracted from images represent a partial and sparse description of the scene, knowledge of planar-faced objects by itself is generally not adequate for completing many of the objects in the model. Knowledge of urban scenes that contain block-shaped objects has been useful for this task. This knowledge is described in [17], and involves (1) completing the shapes of faces in the form of a parallelogram (Fig. 34(d)) or other polygon (Fig. 34(e)), and (2) hypothesizing vertical faces for incomplete objects (Fig. 34(f)).

### 7.3. Examples of generating the 3D scene model

When the techniques described above are applied to the output of the stereo analysis component depicted in Fig. 33, we obtain the scene model shown in Fig. 36. 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. When these techniques are applied to the output of the monocular analysis component shown in Fig. 23, we obtain the scene model shown in Fig. 37. 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.

Figures 38 and 39 show the result of adding gray scale to the faces of the models in Figs. 36 and 37, respectively. The technique for doing this is described in [17].

FIG. 36. Perspective views of buildings reconstructed from wire-frame data in Fig. 33. Compare with Fig. 4.

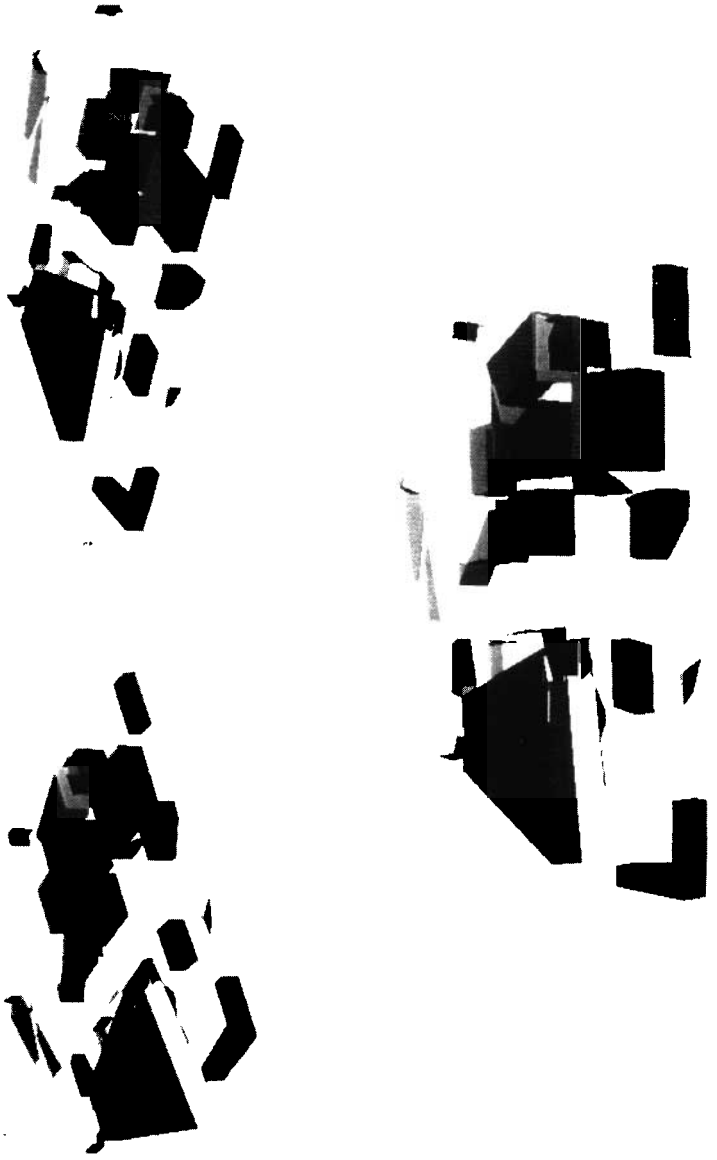

FIG. 37. Perspective views of buildings reconstructed from wire-frame data in Fig. 23. Compare with Fig. 11.

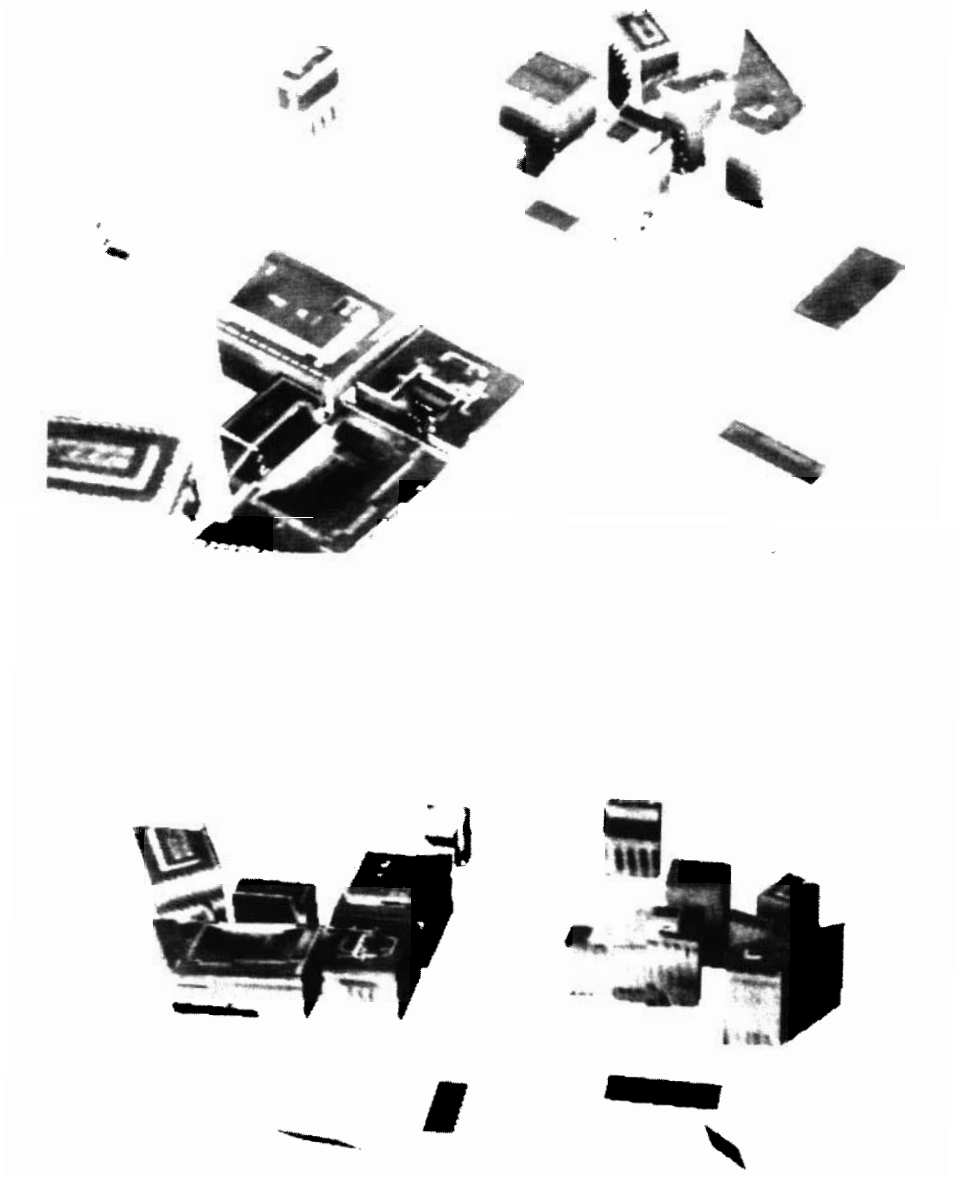

FIG. 38. Reconstructed buildings of Fig. 36 with gray scale, derived from the top image in Fig. 4, mapped onto faces. On a color display, faces and portions of faces occluded in the original image are colored red.

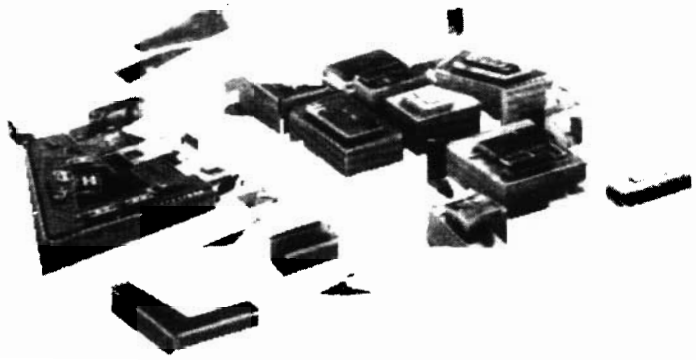

FIG. 39. Reconstructed buildings of Fig. 37 with gray scale, derived from Fig. 11, mapped onto faces.

Although the stereo, monocular, and model generation components of the system have been applied to several other images of downtown Washington, DC, the results presented in this paper are the best we have obtained. We are currently working on improving the system so as to make it more robust.

## 8. Combining New Views with the 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 the previous section.

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. 36. This scene model was constructed from the set of wire frames (Fig. 33) automatically extracted from a "front" view of the scene (Fig. 4). The second set of wire frames, shown in Fig. 40, was manually generated to simulate information available from an opposing point of view (viewing the scene from the "back"). Notice that the information in Fig. 33 emphasizes edges and vertices facing the front of the scene, while those facing the back of the scene are emphasized in Fig. 40.

### 8.1. 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 (but see [16]).

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

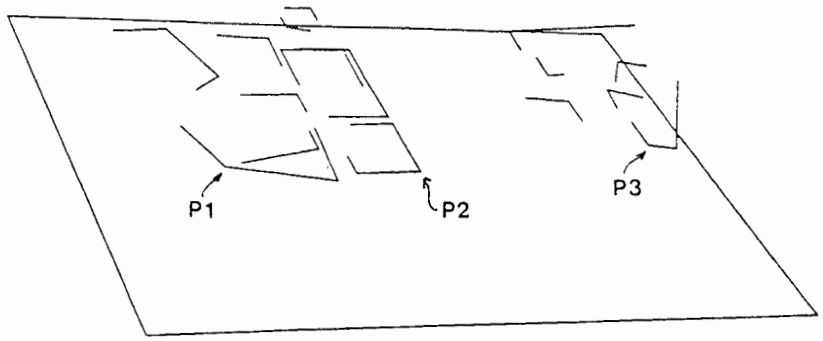


FIG. 40. Perspective view of manually generated vertices and edges which simulate information available from images showing an opposite point of view from that shown in Fig. 4. The viewpoint for this drawing is chosen to be similar to Fig. 33. Points $P_1$, $P_2$, and $P_3$, for example, correspond to points $P_1$, $P_2$, and $P_3$ in Fig. 33.

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

As an example, consider Fig. 42. Suppose the object in Fig. 42(a) is part of the model. Suppose also that the wire-frame object in Fig. 42(b) has been derived from a new view, and it has been transformed to register with the model object. The following algorithm is used to match the two.

*Step* 1. Find pairs of confirmed vertices that match by determining which ones lie within a threshold distance of one another. The vertices $v_2$ and $v_{100}$ are found to match, but let us suppose the distance between $v_3$ and $v_{101}$ exceeds the threshold.

*Step* 2. Find pairs of confirmed matching edges that contain previously found matching vertices. The edges $e_2$ and $e_{100}$, and $e_3$ and $e_{101}$, contain matching vertices and, using the distance and overlap tests, are found to match.

*Step* 3. For each new matching pair of edges found, if they contain a single pair of matching vertices, match their other vertices (if they exist and are confirmed). The vertices $v_3$ and $v_{101}$ match because $e_3$ and $e_{101}$ match. No new matching vertices result from the matching edges $e_2$ and $e_{100}$, since $e_{100}$ has only one vertex.

*Step* 4. Proceed by repeating Step 2, i.e., find new pairs of confirmed matching edges that contain previously found matching vertices. The edges $e_4$ and $e_{12}$ are compared with $e_{102}$ and $e_{104}$, respectively. Using the distance and overlap tests, $e_4$ and $e_{102}$, as well as $e_{12}$ and $e_{104}$, are found to match.

*Step* 5. Next, Step 3 is repeated. New matching vertices are sought that lie on newly found matching pairs of edges. The matching edges found in Step 4 contain no new matching vertices, since $v_4$ and $v_6$ are unconfirmed. The algorithm therefore halts at this point; it would have continued with Step 2 if new matching vertices had been found. The following pairs of matches are returned:$(v_2, v_{100})$, $(v_3, v_{101})$, $(e_2, e_{100})$, $(e_3, e_{101})$, $(e_4, e_{102})$, $(e_{12}, e_{104})$.

### 8.2. 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 using the structure graph modification techniques described in Section 6.

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 [11].

We currently record these relationships for the following situations:

(1) When two nontouching partial faces are merged (Fig. 41(a)), each face has two partial 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. 41(a), the arrows indicate the dependencies. Vertex $v_1$ is dependent on edges $e_1$ and $e_3$, and vertex $v_2$ is dependent on edges $e_2$ and $e_4$. 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 an incomplete face is completed in the shape of a parallelogram

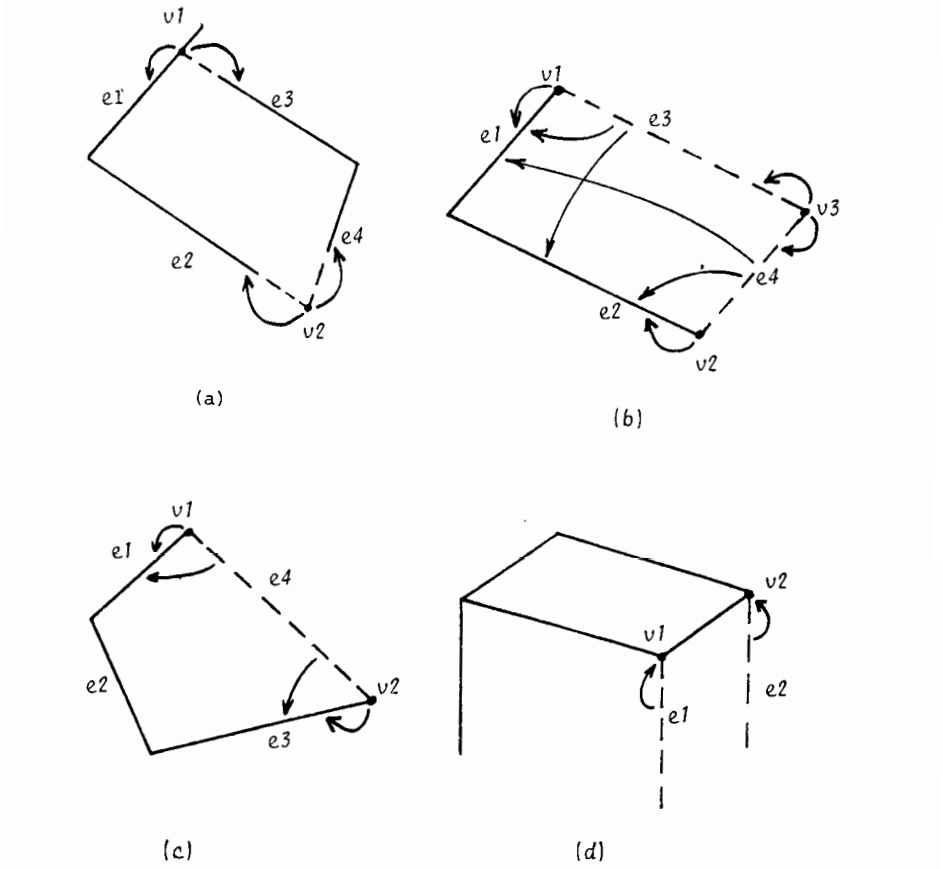


FIG. 41. Generating dependencies for hypothesized edges and vertices. The dependence of an element on another is depicted as an arrow from the former to the latter. (a) Two nontouching partial faces are merged. (b) A face is completed in the shape of a parallelogram. (c) A face is completed by connecting its two end points. (d) Vertical edges are dropped from a floating face.

(Fig. 41(b)), two new edges and three new vertices are hypothesized. Each of the new edges $e_3$ and $e_4$ is dependent on both of the old edges $e_1$ and $e_2$. The edge $e_3$, for example, is dependent on $e_1$ in the sense that its end point is constrained by the end point of $e_1$. It is dependent on $e_2$ in the sense that it is constrained to be parallel to $e_2$. The new vertex $v_3$ is dependent on the two hypothesized edges $e_3$ and $e_4$, while the new vertices $v_1$ and $v_2$ are dependent on the confirmed edges on which they lie.

(3) When a face is completed by connecting its two end points (Fig. 41(c)), two new vertices and one new edge are hypothesized. The new edge $e_4$ is dependent on both $e_1$ and $e_3$, while the new vertices $v_1$ and $v_2$ are dependent on the edges on which they lie.

(4) 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. 41(d), the new edges $e_1$ and $e_2$ are dropped from, and are dependent on, the vertices $v_1$ and $v_2$, 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. After edges are dropped from all vertices of the face, vertical faces are generated. This results in more hypothesized edges and vertices. The situations under which these are created fall under categories (2) and (3) above.

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 noncolinear points. Such faces are also deleted.

### 8.3. 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 nonvertex end points. Therefore, confirmed vertices are given more weight during merging. As an example, consider again Fig. 42, where the wire-frame object in Fig. 42(b) is to be merged with the model object in Fig. 42(a).

The merging procedure starts by merging corresponding vertices. Pairs of such vertices ($(v_2, v_{100})$ and $(v_3, v_{101})$ in Fig. 42) are combined into single vertices with coordinates of the midpoint between them. If the distance between an initial pair of such vertices exceeds a threshold, all hypothesized edges and vertices that recursively depend on the initial model vertex are deleted. Hypothesized faces that have lost minimal support are also 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 ($(e_3, e_{101})$ in Fig. 42), the new edge connects the two new vertices already generated.

(2) If one edge has two confirmed vertices but the other does not ($(e_2, e_{100})$ and $(e_4, e_{102})$ in Fig. 42), the new edge is the same as the former. Notice that the nonvertex 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 vertices ($(e_{12}, e_{104})$ in Fig. 42), the new edge is the "average" of the two edges, obtained using a least-squares fit.

Before merging, a model edge may contain either one or two confirmed vertices. If it contains one confirmed vertex, then all hypothesized edges and

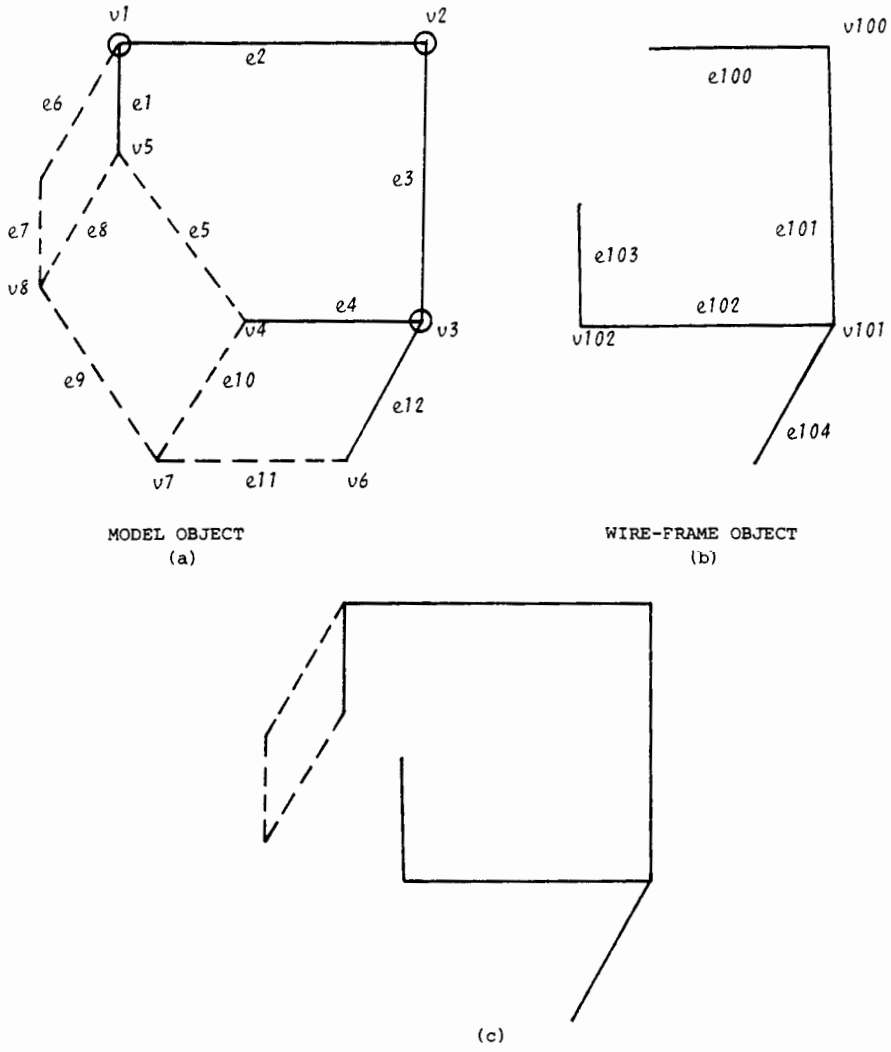


FIG. 42. The wire-frame object in (b) is to be merged with the model object in (a). The confirmed edges of the model object (indicated by solid lines) are $e_1$, $e_2$, $e_3$, $e_4$, and $e_{12}$; the confirmed vertices (indicated by circles) are $v_1$, $v_2$, and $v_3$. Dashed lines represent hypothesized edges. (c) The result after merging.

vertices in the model that recursively depend on this edge are deleted. Hypothesized faces that have lost minimal support are also deleted. In Fig. 42, this occurs for the edges $e_4$ and $e_{12}$. The hypothesized elements in the figure that recursively depend on, say, $e_4$ are the vertices $v_4$ and $v_7$, and the edges $e_5$, $e_{10}$, $e_9$ and $e_{11}$. If a model edge to be merged contains two confirmed vertices

FIG. 43. Perspective views of buildings derived by incorporating the wire-frame data in Fig. 40 into the model in Fig. 36.

(e.g., $e_2$ and $e_3$ in Fig. 42), no hypothesized elements need be deleted since all necessary deletions were made 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 ($e_{103}$, in Fig. 42) are added to the model object. The final configuration after merging is shown in Fig. 42(c). This object is incomplete and must be completed using the techniques described in Section 7.

### 8.4. Results of merging

When these procedures are applied to the wire-frame data in Fig. 40 and the scene model in Fig. 36, we obtain the updated scene model shown in Fig. 43. 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.

## 9. Conclusions

We set out to develop an entire vision system to interpret complex images, one that goes all the way from images to symbolic 3D descriptions. The following are some conclusions we can draw from this project.

(1) Complex images usually cannot be fully interpreted. Difficulties in interpretation arise not only from occlusions, but also from variations in surface texture and reflectance, variations in shape, and complex lighting conditions. Computer vision systems must therefore have the capability to deal with approximate, imperfect scene descriptions when performing tasks such as matching, path planning, or model-based image interpretation.

(2) Incremental reconstruction of complex scenes will often be necessary. Multiple views are required to effectively reconstruct complex scenes. A

system that moves about and interacts with its environment in order to obtain the multiple views will be able to gradually add more information to its scene model at the same time that it carries out its other tasks.

(3) Scene descriptions are often more useful if they contain reasonable hypotheses for parts of the scene for which there are only partial or no data. For example, path planning cannot be done for occluded regions of the scene without a good guess about what lies in these regions. If the hypotheses turn out to be incorrect, they should eventually be modified. Computer vision systems must therefore have mechanisms for intelligently generating hypotheses, verifying them, and modifying them.

(4) Task-specific knowledge is very useful at all levels of complex image interpretation, from low-level image analysis to high-level formation of symbolic descriptions. Knowledge of block-shaped objects in an urban scene is used in the 3D Mosaic system for stereo analysis, monocular analysis, and reconstructing shapes from the wire frames.

(5) Stereo matching of 2D structural features (such as junctions) may be important for complex images and should be further investigated.

## ACKNOWLEDGMENT

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

1. Baer, A., Eastman, C. and Henrion, M., Geometric modelling: A survey, *Comput.-Aided Des.* **11** (1979) 253–272.
2. Baker, H.H., Three-dimensional modelling, in: *Proceedings Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA (1977) 649–655.
3. Baker, H.H. and Binford, T.O., Depth from edge and intensity based stereo, in: *Proceedings Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC (1981) 631–636.
4. Barnard, S.T., Methods for interpreting perspective images, in: *Proceedings ARPA Image Understanding Workshop*, 1982.
5. Barnard, S.T. and Fischler, M.A., Computational stereo, *Comput. Surveys* **14**(4) (1982) 553–572.
6. Barnard, S.T. and Thompson, W.B., Disparity analysis of images, *IEEE Trans. Pattern Anal. Machine Intell.* **2**(4) (1980) 333–340.
7. 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, in: *Proceedings Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA (1977) 696.
8. Baumgart, B.G., Geometric modeling for computer vision, Tech. Rept. STAN-CS-74-463, Department of Computer Science, Stanford University, Stanford, CA, 1974.
9. Bourne, D.A., Milligan, R. and Wright, P.K., Fault detection in manufacturing cells based on three-dimensional visual information, *Proc. SPIE* **336** (1982) 29–36.

10. Doyle, J., A truth maintenance system, *Artificial Intelligence* **12** (1979) 231–272.
11. Doyle, J., Three short essays on decisions, reasons, and logics, Tech. Rept. STAN-CS-81-864, Department of Computer Science, Stanford University, Stanford, CA, 1981.
12. Eastman, C.M. and Preiss, K., A unified view of shape representation and geometric modeling, in: *Proceedings Israel Conference on CAD*, 1982.
13. Grimson, W.E.L., A computer implementation of a theory of human stereo vision, *Phil. Trans. Roy. Soc. Lond.* **B292** (1980) 217–253.
14. Hannah, M.J., Computer matching of areas in stereo images, Tech. Rept. AIM-239, Stanford University, Stanford, CA, 1974.
15. Henderson, R.L., Miller, W.J. and Grosch, C.B., Automatic stereo reconstruction of man-made targets, *Proc. SPIE* **186**(6) (1979) 240–248.
16. Herman, M., Matching three-dimensional symbolic descriptions obtained from multiple views of a scene, in: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, 1985.
17. Herman, M., Kanade, T. and Kuroe, S., Incremental acquisition of a three-dimensional scene model from images, *IEEE Trans. Pattern Anal. Machine Intell.* **6**(3) (1984) 331–340.
18. Kanade, T., Recovery of the three-dimensional shape of an object from a single view, *Artificial Intelligence* **17**(1–3) (1981) 409–460.
19. Kender, J.R., Environmental labelings in low-level image understanding, in: *Proceedings Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, F.R.G. (1983) 1104–1107.
20. Lowe, D.G. and Binford, T.O., The interpretation of three-dimensional structure from image curves, in: *Proceedings Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, 1981.
21. Lucas, B.D. and Kanade, T., An iterative image registration technique with an application to stereo vision, in: *Proceedings Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC (1981) 674-679.
22. Martin, W.N. and Aggarwal, J.K., Volumetric descriptions of objects from multiple views, *IEEE Trans. Pattern Anal. Machine Intell.* **5**(2) (1983) 150-158.
23. Moravec, H.P., Obstacle avoidance and navigation in the real world by a seeing robot rover, Tech. Rept. CMU-RI-TR-3, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, 1980.
24. Ohta, Y. and Kanade, T., Stereo by intra- and inter-scanline search using dynamic programming, Tech. Rept. CMU-CS-83-162, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1983.
25. Potmesil, M., Generating models of solid objects by matching 3D surface segments, in: *Proceedings Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, F.R.G. (1983) 1089–1093.
26. Requicha, A.A.G., Representations for rigid solids: Theory, methods, and systems, *Comput. Surveys* **12**(4) (1980) 437–464.
27. Rubin, S., Natural scene recognition using locus search, *Comput. Graph. Image Process.* **13** (1980) 298–333.
28. Underwood, S.A. and Coates, C.L., Visual learning from multiple views, *IEEE Trans. Comput.* **24**(6) (1975) 651–661.

*Received May 1985; revised version received April 1986*