SCENE ANALYSIS BASED ON IMPERFECT EDGE DATA*

Gilbert Falk
Rutgers University, New Brunswick, N. J.

ABSTRACT

The paper describes a heuristic scene description system. This system accepts as input a scene represented as a line drawing. Based on a set of known object models, the program attempts to determine the identity and location of each object viewed. The most significant feature of the system is its ability to deal with imperfect input data. This ability appears essential in light of our current stock of preprocessing techniques and the variation that is possible in real world data.

INTRODUCTION

A hand-eye system is a problem solving system with an eye (camera) for input and a hand (manipulator) for output. Such a system must have at least 1) a set of scene analysis (perception) programs which interpret the real world in a meaningful way, 2) a set of manipulation programs which control movement of the hand in 3-space, and 3) an executive (problem solver, strategy) program which directs the perceptual and motor processes toward a desired goal. This paper concentrates on the problems of machine perception. In particular, we shall describe some programs that were developed as part of a hand-eye system (4) at the Stanford Artificial Intelligence Laboratory. We shall also contrast our approach with previous work (6,8) and indicate some directions for future work.

Computer perception can be thought of as a large data reduction problem. A matrix of digitized intensity values is read into memory by means of some imaging device. The goal of analysis is a concise description of the scene viewed. The description or interpretation should correspond approximately to the description that a person would give when presented with the same scene. It should contain at least the identity and location of each object. The techniques which we shall describe generate such interpretations based on a single perspective projection.

Any system capable of interpreting its input must in some sense be model-based. The models provide the difference between the input information and the information contained in the interpretation. People seldom need to see the back side of an object which they recognize before "knowing" how it looks from behind. From a purely mathematical point of view, some sort of model is required to produce a three-dimensional description of the world based on a single two-dimensional image.

Consider a set of models (see Fig. 1) which are acceptable as input to a predictor. This means there exists an algorithm, the predictor, which takes the set of models as input and is capable of generating any possible scene comprised of instances of these models as output. The particular scene is specified by a finite set of parameters (variables). These parameters are the names and locations of the objects present in the scene. In this framework the process of interpreting a scene can be viewed as the process of finding values for the variables such that the scene generated by the predicator matches the input scene.

The existence of a predicator suggests the following "hypothesis and test" appraoch to scene analysis. Based on the input data, the models, and a set of heuristics, the parameter values comprising the interpretation are determined. These parameters are then used to generate a prediction of the input. The original input and the predicted input are finally compared and the interpretation is accepted if the two match. Significant discrepancies between the original input and the prediction are used to suggest a new interpretation. Such a scheme can be thought of as a combination model-driven (analysis by synthesis) and data-driven approach.

A SIMPLE VISION SYSTEM

To be more concrete, let us consider the (simplified) analysis of a scene. Analysis consists of transforming and abstracting the information in the input. The picture in Figure 2a displays a scene to be analyzed as seen through the TV camera monitor. This image is read into memory and stored as a 333x256 matrix of intensity values. Each element in the array represents the digitized brightness (0-15) at a point in the field of view. An edge-detector program transforms Figure 2a into a set of edge points shown in Figure 2b by applying a local gradient operator and thresholding at every point in the image. Edge points appear where there is a significant intensity gradient. The final stage of preprocessing transforms Figure 2b into a line drawing shown in Figure 2c. This is accomplished by fitting straight lines to the edge points, extending these lines to form corners, and identifying closed regions and the background. The scene analysis system described in the remainder of this paper is designed to accept such a line drawing as input. We do not discuss the problems of preprocessing. Nevertheless, we are

*This research was supported in part by The Advanced Research Projects Agency of the Department of Defence under Contract SD-183 while the author was at Stanford University.

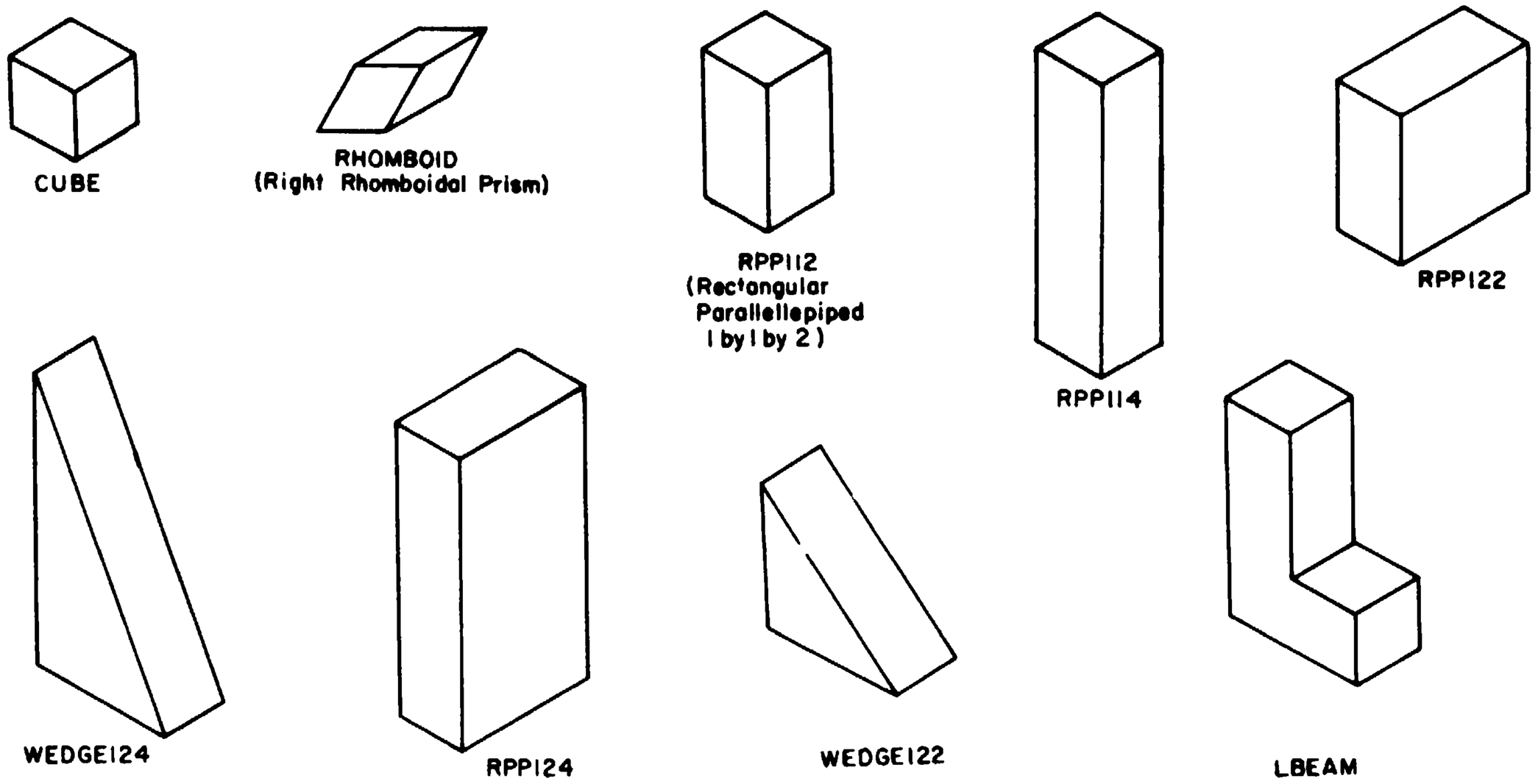


Figure 1

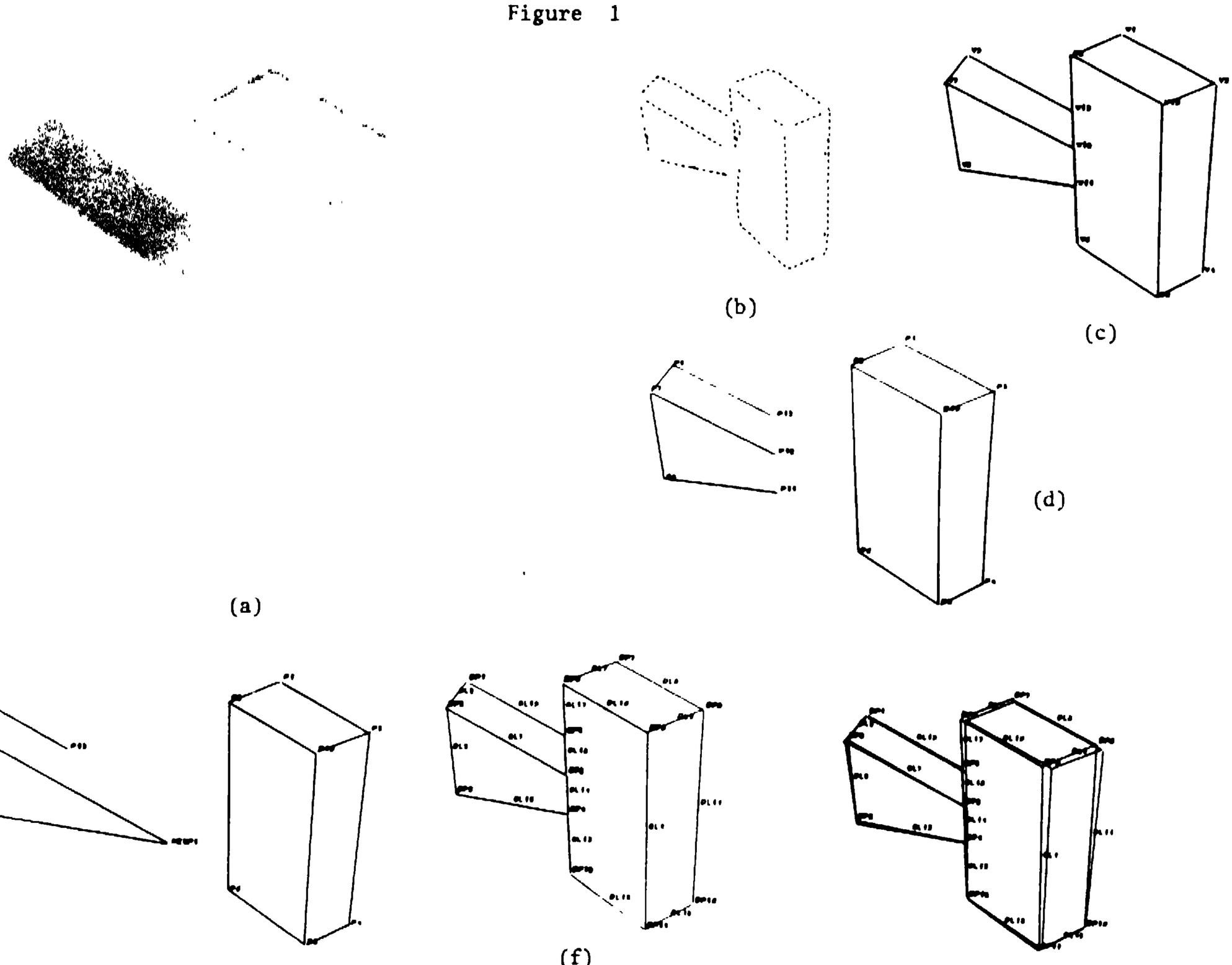


Figure 2

concerned with the quality of output that a processor is likely to produce.

It is assumed that the set of objects is completely specified. The scenes are required to consist only of the solids shown in Fig. 1. This particular set of solids was chosen for the following reasons:

1) There are enough different RPPs to build interesting structures(with the hand)•
2) Not all the parallelepipeds are rectangular (e.g. the RHOMBOID).
3) Not all of the solids are parellelepipeds (e.g. the wedges).
4) Not all of the solids are convex (the LBEAM).

Complete structural descriptions referred to as prototypes (models) exist for these solids. We refer to a real world object as an instance of prototype.

Our fixed size prototypes are somewhat less general than the ones described by Roberts (8). Whereas his "cube" model represented the class of all parallelepipeds, we need a separate model for each physically different solid. The discrete size restriction, however, provides an additional set of constraints which can be applied to resolve ambiguities.

Analysis of the line drawing proceeds in several stages. We assume that the camera has initially been calibrated. This means simply that given any point in the image, we can find the ray in 3-space corresponding to it. The line drawing in Figure 2c is first segmented into pieces corresponding to individual bodies (Fig. 2d). In cases where a body can be (partially) completed, the program does this (Figure 2e). In order to identify and locate the corresponding objects in space, features inferred from the projections of the individual bodies are matched against the stored prototypes. To check that the resulting interpretation of the scene is consistent with the original data, the identities and locations of all the objects are used to generate a predicted line drawing. Figure 2f shows this prediction. Finally, the prediction and the original line drawings are compared. If, as in Figure 2g, the two are approximately the same, the program assumes that it has correctly interpreted the scene. If not, the system must try to produce an alternate description of the scene.

In practice, the analysis is considerably more difficult. Acutal edges are not seen due to poor lighting. In the above example, the line V11-V12 was found only by chance. Often.extraneous lines result from shadows and noisein the video system. There is also the inherent ambiguity in inferring three-dimensional information from a single view. These issues receive particular attention in the rest of this paper.

INTERPRETATIONS OF IMPERFECT LINE DRAWINGS

As indicated in the above example, the process of interpreting a line drawing can be thought to consist of five stages: segmentation, completion, recognition (body identification and location), prediction, and verification (comparison and decision). In this section we describe techniques and difficulties that arise in implementing each of them.

SEGMENTATION

Guzman (6) has described some heuristic procedures for segmenting an ideal line drawing into bodies. If our preprocessors were able to produce ideal line drawings, we could simply incorporate Guzman's program, SEE, as part of our scene analysis system. Experience indicates, however, that it is unreasonable to expect such line drawings. Edges are often missed due to poor lighting and spurious lines can result from random noise in the video system and from shadows.

The problems of missing and extra lines can to some extent be traded for one another. Parameters in the preprocessors can often be set so that all the actual edges appear in the line drawing. Such settings, however, generally cause extraneous lines to appear as well. On the other hand, if one sets the preprocessor parameters so that all noise lines will be rejected, some of the true edges are also rejected. In practice (neglecting shadows) one must choose between analyzing an incomplete line drawing and one containing spurious lines. The former seems to be the more tractable. We will assume in the sequel that no shadow lines are present. Techniques for dealing with shadows are described in (3,7). The analysis of scenes with both missing edges and shadow lines remains a problem that needs work.

Given an incomplete line drawing, a reasonable approach is to first call upon a heuristic program referred to as a line proposer. The line proposer has the job of guessing lines that are missing from the line drawing. Such an approach has been investigated by the MIT vision group (2) and Grape at Stanford (5). If the resulting line drawing were guaranteed to be complete, a segmentation procedure like Guzman's could then be applied to it. The procedure which we shall sketch below, on the other hand, is a complementary approach; that is, it tries to segment the original incomplete line drawing into bodies.

Consider the line drawing shown in Fig. 3. What has happened here is that the lighting caused the triangular face of the wedge and the right vertical face of the cube to be identified as the single region R3. Any segmentation pro-

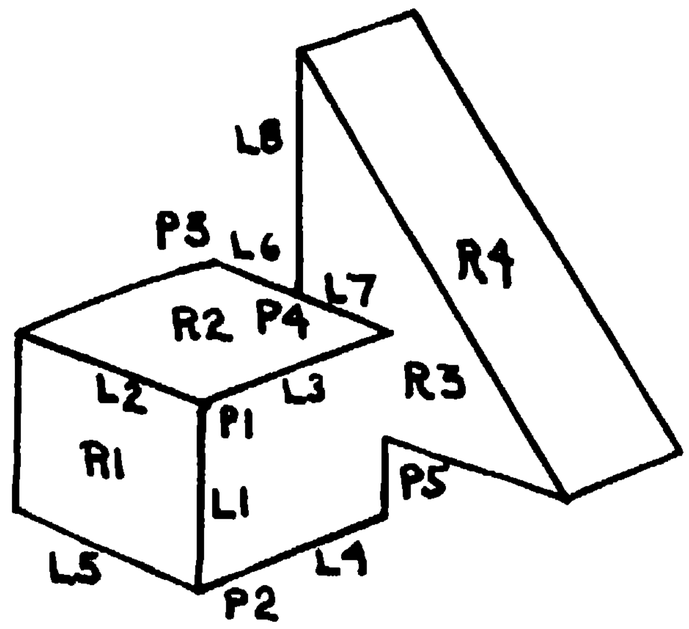


Figure 3

cedure which attempts to partition the scene by region is clearly doomed to failure. Our segmentation proceudre, SEGMENT, first segments the scene into bodies by line. This is accomplished by combining bits of local evidence accumulated at the vertices in a way similar to Guzman. The line "partition" is subsequently used by a region assignment procedures which can detect and split regions such as R3. The resulting set of regions is then partitioned by body.

To indicate roughly how the line partition is generated, we again refer to Figure 3. Consider the "Y" shaped vertex at PI. This is taken to indicate that lines L1, L2, and L3 should be part of the same body. Similarly, the "W" shaped vertex at P2 is taken to indicate that lines L1,L4,and L5 should be assumed to be part of the same body,Since these two triples share the common line LI,it is inferred that the five lines LI, L2, L3, L4, and L5 belong to the same body. The "T" shaped vertex at P4 is taken to indicate that L6 and L7, the collinear segments of the "T", belong to one body and the remaining line, L8 (probably) belongs to a different body. An "L" vertex is one where two lines meet (e.g. P3 and P5). A case like P3 where the background region is associated with an angle greater than 180 degrees is taken to imply that the two lines should be identified as parts of the same body. In a case like P5 where the background is associated with the smaller angle, no such identification is made. There heuristics usually partition the large majority of lines correctly. There are, however, many additional ones that we have not mentioned which deal with special situations.

As indicated previously, the line partition is used to generate a partition on the regions according to body. If all the bounding lines of a REGION X belong to BODY Y then REGION X is assigned to BODY Y. If some of the bounding lines of REGION X belong to BODY Y and some belong to BODY Z, then we must determine whether the region should be assigned to only one of the bodies or should be split (as R3 in Figure 3)and part assigned to each body. We shall not go into the details of this procedure here. The line and region partitions are used to generate descriptions of each (partitially) visible body. These descriptions are the input to the body completion routines.

COMPLETION

Before recognition is attempted, the completion routines try to complete partial line drawings of single objects (see Figure 4). A decision was made to be very conservative about doing this. Partial projections are fixed up only where it is quite clear that no mistake will result. Although the recognition procedure does not demand that the projection it is presented be complete, its chances of success are better when there is no missing data.

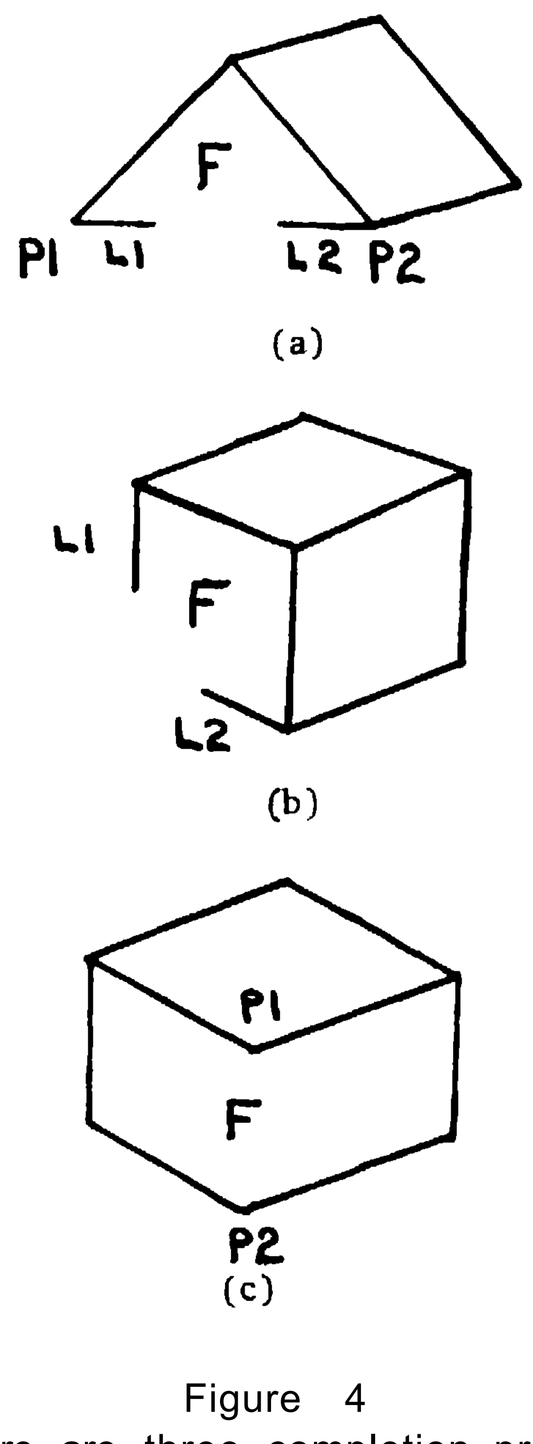


Figure 4

There are three completion procedures:JOIN, ADDCORNER, and ADDLINE. JOIN handles cases similar to that of Figure 4a. It looks for a face F, which 2 hanging collinear lines indicate is incomplete. It replaces these two lines (LI and L2) by a single line between PI and P2.

ADDCORNER handles cases such as Figure 4b. It looks for a face, F, that is incomplete due to two hanging lines which can be extended to form a corner. It completes the face by extend-

ing these lines to the new corner and fixing up the rest of the body description accordingly. One must be careful, however, not to attempt to extend nearly parallel lines.

ADDLINE looks for evidence that an entire line has been missed. Although there are no dangling lines in Figure 4c, something is definitely missing. ADDLINE adds a line between points PI and P2 based on the pair of "L" type vertices there with parallel sides. Face F is plit into 2 faces.

RECOGNITION

Recognition consists of identifying the object as an instance of a prototype and locating the object in space.

The identification procedure tries to name a given object by extracting features from its line drawing projection, P, and matching these against the properties of the 3-D prototypes. If we define PROTOTYPES to be the set containing all existing prototypes, we can represent this process schematically as:

$$\text{PROTOTYPES} \rightarrow F_1(P) \rightarrow S_1 \rightarrow F_2(P) \rightarrow \ldots\ldots \rightarrow F_n(P) \rightarrow S_n.$$

Feature F_1 is extracted from the projection and those prototypes which might possibly have this feature are put in S_1 ($S_1 \subseteq$ PROTOTYPES). A fixed number, n-1, of other features are similarly used in an attempt to further restrict the possible interpretations of projection P. Hopefully, for some i, $1 \le i \le n$, S_i is a singleton and the object is identified. If this is not the case, the identification procedure can only say that the object is one of the prototypes in Sn.

The set of features consist of geometric properties such as face shape (triangular, square. hexagonal), angle, and edge lengths. A problem arises, however, from the fact that the projection features are two-dimensional whereas the prototype features are three-dimensional. A solution to this problem is to infer 3-D features from the line drawing and then match these against the prototypes. A number of heuristic techniques based on the notion of support hypothesis are used to do this. Roughly speaking, support hypothesis says that if one knows a plane (the support plane) in which a given image point lies, then this plus the ray to the point (known since the camera was initially calibrated) determines the 3-space location of the point precisely. Although stereo ranging (10) and focus ranging (11) provide a more direct way to locate a given image point in space, these procedures can be quite clostly to apply. We currently plan to resort to these techniques only when the heuristic ones fail. A thorough investigation of how and when binocular cues should be utilized remains to be done. To simplify the analysis based on support hypothesis, we shall assume that objects not supported by the table are resting on the top horizontal faces of other objects.

In order to apply support hypothesis one must determine not only the base points of each object, but also binary relations of the form SUPPORTER (BODY X, BODY Y) for all X and Y such that BODY Y supports BODY X on its top face. The technique which we use to do this is quite similar to the one described by Winston (13). A difference in approach, however, is that we use a partial 3-D scene description as well as information contained in the image to resolve support ambiguities. This point will be made clear in one of the examples which follow.

After deciding to which model a particular object in the scene corresponds, the recognition procedure proceeds to determine its location in space. The most obvious way to specify this is to find the location of the center of mass of the object and the angles that 3 known orthogonal axes in the body make with the axes of the table coordinate system. The techniques described below provide a convenient way to compute and record these six numbers.

Consider a prototype with its center of mass at the origin of the table system and an arbitrary orientation. Now imagine an instance of this prototype translated and rotated to some other location in space. If we represent the vertices of the unmoved prototype in homogeneous coordinates (1) as $Pj = (Xj, Yj, Zi, 1)^T$ and those of the instance as $Pj' = (Xj', Yj', Zj', 1)^T$ then we can write $Pi' = T\,Pi$ where

$$T = \begin{bmatrix} & & & DX \\ & R & & DY \\ & & & DZ \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The upper left hand corner of T is a 3x3 rotation matrix, R, and the first 3 components of the fourth column describe the translation of the center of mass of the instance in the table system. By finding the location and orientation of an object we shall mean finding the matrix T which moves the identified prototype to the position of the instance.

There are six parameters needed to specify T;DX, DY, DZ, and the three angles implicit in R. Knowing the 3-space locations of three corresponding points on the object and model will permit T to be computed. We shall not go into the actual procedure used to set up these correspondences .

PREDICTION

After having identified and located each of the objects in the scene, the predictor is in a

position to say how the described scene would appear in the image. This problem has been investigated rather extensively in recent years by workers in the area of computer graphics. A major goal of their research has been to find efficient algorithms for solving the so-called "hidden-line (surface) problem" (9,12). For the extremely complex objects and scenes with which they deal, an efficient algorithm is a necessity. Our vision system, on the other hand, is currently not able to analyze scenes of such complexity and is not as strictly constrained by the need for efficiency.

The algorithm currently used by the predictor is a modified brute force approach. First, the predictor generates a prediction with none of the hidden lines removed. Next, lines on the "back faces" of individual objects are deleted from the prediction. Back faces can be determined by comparing the face normals with the line of sight vector. Finally, the remaining hidden lines are removed by computing all the line intersections in the image plane and determining which of the resulting line segments are behind visible faces. It is this last step which is time consuming and a number of heuristics are used to speed up the process. The predictor is totally independent of the rest of the scene description system and could, therefore, easily be replaced by one employing a more efficient algorithm.

VERIFICATION

Verification is the process of comparing the predicted line drawing with the original input and making a decision based on the result of this comparison. There are two distinct interpretations for "original input" here. First, the predicted line drawing should (approximately) match the original line drawing from which it was derived. This can be confirmed by checking that for each line of the prediction there is a corresponding line in the original line drawing. If such a correspondence cannot set up, the verifier concludes that either it has failed in its interpretation of the scene or the original line drawing was incomplete, i.e., some lines were missing. Quite often such missing lines can be detected by a statistical line verification operator (11) applied to the intensity data in the vicinity of the predicted edge. While such an operator is too costly to apply everywhere in the original intensity matrix, it is reasonable to apply selectively at this stage of the analysis.

If all of the predicted lines match lines in the original line drawing, the verifier accepts the interpretation of the scene. For any predicted line that cannot be matched, the statistical verification operator is called to check for the edge (line) in the original TV data. If for a particular body no more than N of its predicted edges remain unconfirmed (N is currently set equal to 3), the verifier assumes that the body has been recognized correctly. If the prediction of a body does not match the input data, the verifier assumes that the body has been recognized incorrectly and asks the recognizer to try again for this body.

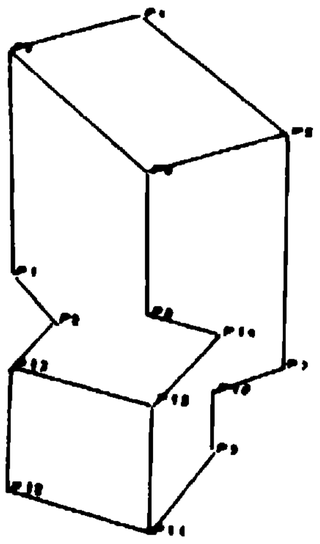
This procedure handles the case where the recognizer is unable to identify an object uniquely. Unfortunately, the scene analysis programs at present are unable to revise decisions made during the segmentation and completion stages. Such a capability is certainly one which would be desirable to add in the future. The examples which follow indicate how the system performed on some real scenes and will hopefully clarify some of the points made above.

SCENE I

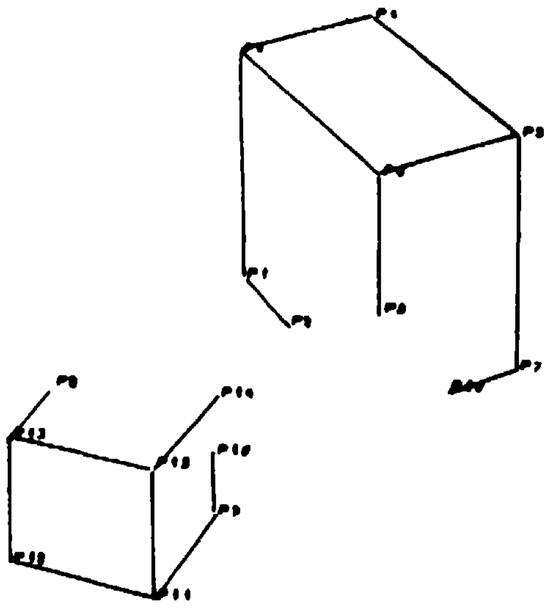
The scene of Figure 5a consists of an RPP122 with a cube in front of it. The intensity distribution was such that in two cases regions that actually correspond to 2 faces result. Three visible edges are not present in the line drawing. Note that P8 and P14 are "L" vertices of a type we have not mentioned previously, i.e. neither associated region is the background. To avoid making a segmentation error at points such as P8, we have chosen not to infer anything from such "L" vertices. Because of this decision, we see in Figure 5b that the line P8-P14 is not present in the segmented line drawing. Figure 5c shows the situation after completion of the individual bodies has been attempted. For the CUBE, lines P9-P10 and P14-P15 have been extended to the corner NEWP1 by ADDCORNER. Application of ADDCORNER twice to the RPP122 results in NEWP2 and a completed line drawing for the body. Since neither body is found to support the other, they both must be supported by the table. Identification of points PII, P9, P12, PI, P7 and NEWP2 as table (base) points allows recognition using 3-D information inferred from the line drawing. Figure 5d shows the predicted line drawing. As is clear from Figure 5e, the prediction and original input match quite well except for the three missing edges. Most probably these could be found in the TV data by the verification operator. Even if they could not, enough of the lines match so that the hypothesized scene description is acceptable.

SCENE II

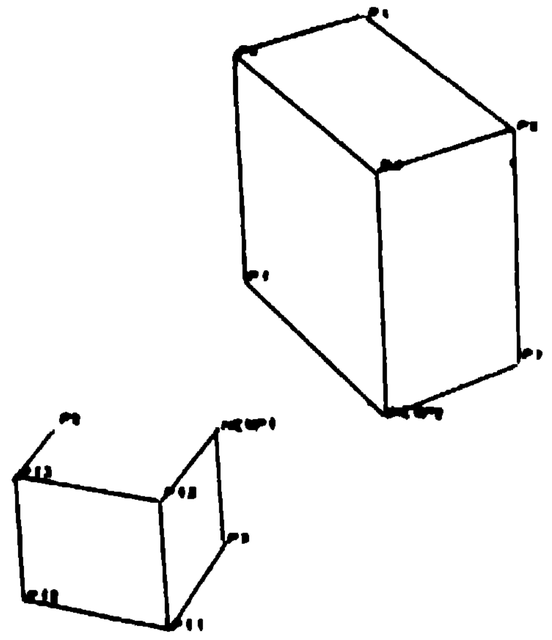
In contrast to the previous scene where both blocks rested on the table, this scene (Figure 6a) illustrates the complications arising from other forms of support. As in the previous example, there are two vertices whose type was not considered in our abbreviated discussion of the segmentation procedure. The "Y" type vertices at PI3 and P5 are not used to infer any line groupings since one of the 3 regions neeting thene is the background. Figure 6b illustrates the situation after segmentation and Figure 6c shows



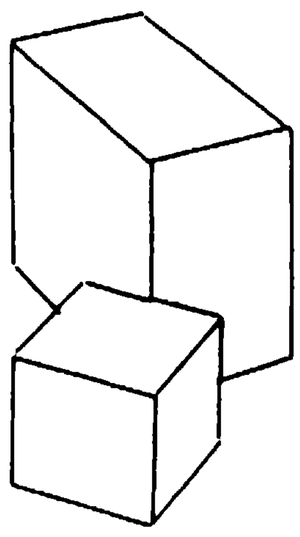
(a)



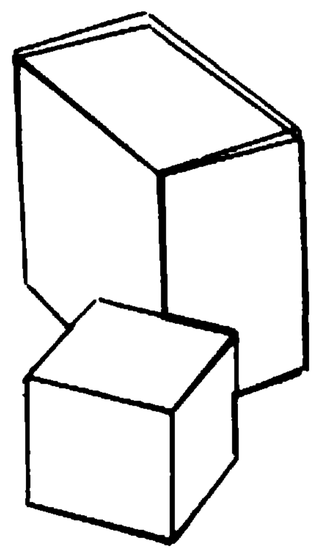
(b)



(c)

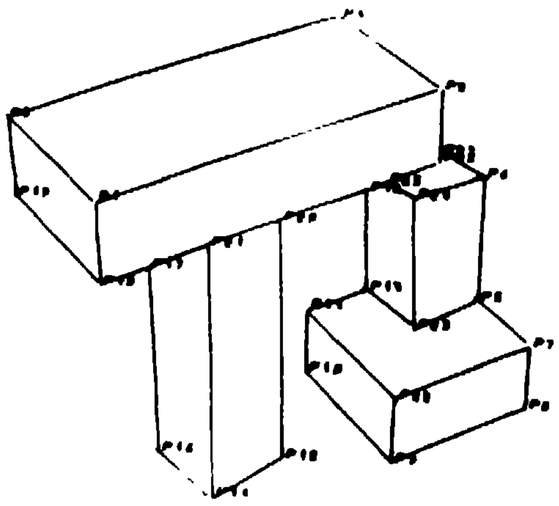


(d)

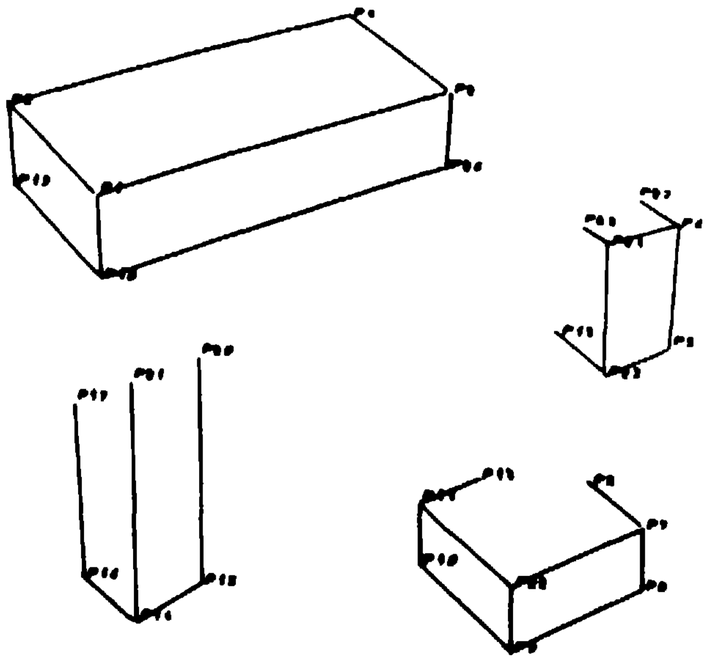


(e)

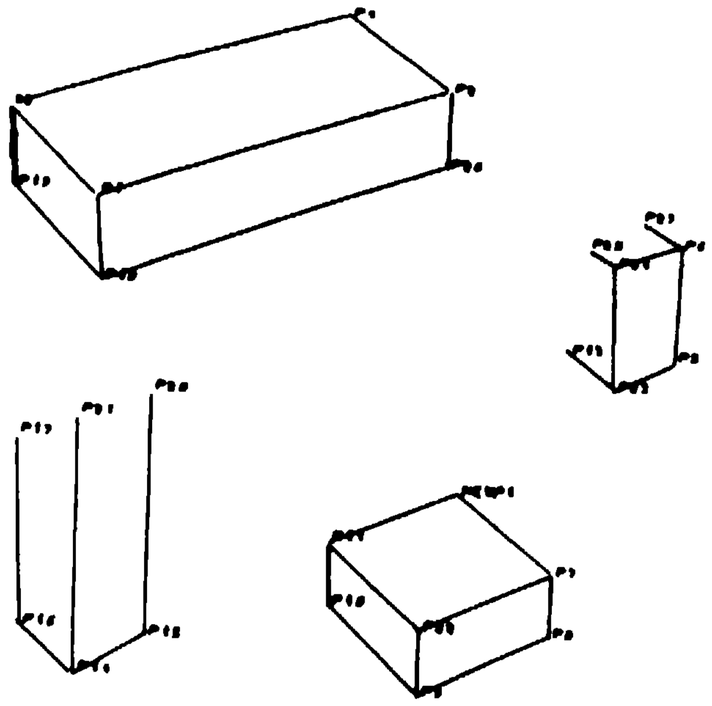
Figure 5



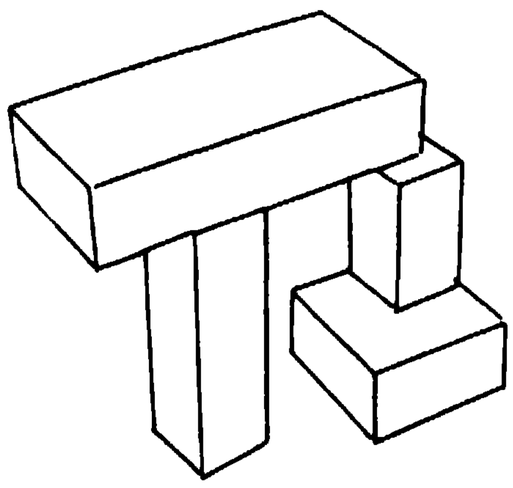
(a)



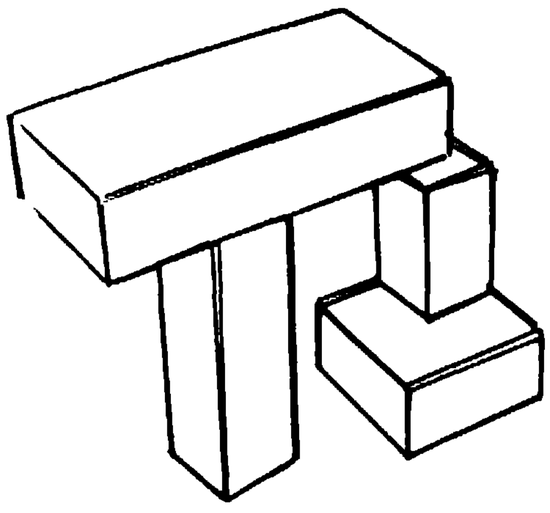
(b)



(c)



(d)



(e)

Figure 6

the partial body projections that result after completion.

Recognition proceeds in a bottom-to-top fashion. To begin with the recognizer might decide to identify either the RPP114 or the RPP122 which are both resting on the table. Assume that it recognizes the RPP114. It then tries to recognize the RPP124 above the RPP114 but this fails since it has not yet recognized the RPP-112. This must be done in order to determine which object the RPP124 is actually resting on. The next body processed, therefore, must be the RPP122 resting on the table. After it is identified and located in 3-space, its top face can be used as the support plane for the RPP112 which is processed next. Now knowing the positions of the RPP114 and the RPP112, the system can determine the true support plane of the RPP124. Finally, this last body is recognized.

The prediction is shown in Figure 6d and with the original line drawing superimposed in Figure 6e. Again the two match and the interpretation of the scene is accepted.

CONCLUSION

We have sketched the operation of a large heuristic program capable of interpreting line drawings as a three-dimensional scene. We have tested the program on approximately 30 different line drawings containing between one and six objects. Failures arose only where a body was too occluded by another, many links were missed by the preprocessors, or one body was leaning on another. Such scenes are often confusing to humans as well. The only previous scene description system comparable to ours is the one described by Roberts. Guzman's recognition programs did not operate on real data nor were they concerned with the problem of locating an object in space. Our scene description techniques are able to deal with more complex scenes than was the Roberts system. Objects can be supported by one another as well as by the table. The most distinctive feature of the system, however, is its ability to interpret a scene based on imperfect data. Tn addition to tolerating inaccuracies in its input, it can analyze degemerate views of objects, objects which appear partially occluded, and line drawings in which edges are totally missing. The potential to cope with these situations is a consequence of the basic organization we employed. The system uses its line drawing input and a set of models to suggest and test hypotheses. We believe that this approach to machine perception will prove to be a fruitful one.

ACKNOWLEDGEMENTS

I wish to thank my co-workers at The Stanford Artificial Intelligence Project for suggestions and subroutines supplied. Particular thanks to J. A. Feldman for guidance and encouragement during the course of this work.

REFERENCES

1. Ahuja, D.V. and Coons, "Geometry for Construction and Display',' IBM Systems Journal, Vol.7 Nos. 3 & 4, pp. 188-206, 1968,

2. Binford, T., "A Visual Preprocessor," Internal Report, Project MAC, Massachusetts Institute of Technology, Cambridge, Mass. Apr.1970.

3. Falk, G.,"Computer interpretation of imperfect line data as a three-dimensional scene," Computer Science Departmental Report, No. CS180, Stanford University, Stanford, Calif.,August 1970.

4. Feldman, J.A., Feldman, G.M., Falk, G., Grape, G., Pearlman, J., Sobel, I., Tenenbaum, J.M., The Stanford hand-eye project. Proceedings of the First International Joint Conference on Artificial Intelligence, pp. 521-526,May 1969.

5. Grape, G.R., "On predicting and verifying missing elements in line-drawings, based on brightness discontinuity information from their initial TV-images," Course Project for CS225 and CS382, Stanford University, Stanford, Calif., March 1970.

6. Guzman, A./'Computer recognition of three-dimensional objects in a visual scene," Ph.D. Thesis, Project MAC TR-59, Massachusetts Institute of Techology, Cambridge, Mass. December 1968.

7. Orban, R.,"Removing shadows in a scene," Project MAC, Artificial Intelligence Memo No.192, Massachusetts Institute of Technology, Cambridge, Mass., April 1970.

8. Roberts, L.G., "Machine perception of three-dimensional solids," Technical Report No.315, Lincoln Laboratory, M.I.T., May 1963. Also in Optical and Electro-Optical Information Processing, Tippett, J.T. et al.

9. Ronuey, G. W. , "Computer assisted assembly and rendering of solids," Rome Air Development Center Technical Report 69-365, University of Utah, September 1969.

10. Sobel, I., "Camera models and machine perception," Ph.D. Thesis, Artificial Intelligence Memo No. 121, Stanford University,Stanford, Calif., May 1970.

11. Tenenbaum, J.M., "Accommodation in computer vision," Ph.D.Thesis, Computer Science Department Report No. CS182, Stanford University, Stanford, Calif., October 1970.

12. Warnock, J.E.,"A hidden line algorithm for halftone picture representation," University of Utah Computer Science Technical Report 4-15, May 1968.