STANFORD ARTIFICIAL INTELLIGENCE PROJECT
MEMO AIM-132

COMPUTER SCIENCE DEPARTMENT REPORT
NO. CS180

COMPUTER INTERPRETATION OF IMPERFECT LINE DATA AS A THREE-DIMENSIONAL SCENE

BY

GILBERT FALK

AUGUST 1970

COMPUTER SCIENCE DEPARTMENT
STANFORD UNIVERSITY

STANFORD ARTIFICIAL INTELLIGENCE PROJECT
MEMO AIM-132

August 1970

COMPUTER SCIENCE DEPARTMENT REPORT
NO. CS180

COMPUTER INTERPRETATION OF IMPERFECT LINE DATA AS A THREE-DIMENSIONAL SCENE

by

Gilbert Falk

ABSTRACT: The major portion of this paper describes a heuristic scene description program. This program 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 program is its ability to deal with imperfect input data.

We also present some preliminary results concerning constraints in projections of planar-faced solids. We show that for a restricted class of projections, 4 points located in 3-space in additon to complete monocular information are sufficient to specify all the visible point locations precisely.

This research was supported in part by the Advanced Research Projects Agency, of the Department of Defense under contract SD-183.

COMPUTER INTERPRETATION OF IMPERFECT LINE DATA AS A THREE-DIMENSIONAL SCENE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Gilbert Falk

August 1970

ACKNOWLEDGMENTS

I would like to thank Professor Jerome A. Feldman, my thesis adviser, for his guidance and constant encouragement throughout the course of my work. I want to express my appreciation to Professor John McCarthy, Professor William F. Miller, and Dr. Alan C. Kay for their constructive readings of this thesis. Professor George Dantzig, Dr. Peter Hart, Richard Paul, and S. Ganapathy have also provided helpful comments, suggestions, and criticisms.

For programs made available to me, I owe a special debt to a number of my co-workers at the Stanford Artificial Intelligence Laboratory. D. Swinehart and R. Sproull implemented the SAIL language in which the programs described in this thesis are written. Routines belonging to L. Quam, K. Pingle, M. Hueckel, and G. Grape were used to produce my line drawing input. A program for conveniently generating prototype descriptions was provided by R. Paul.

I dedicate this work to my wife, Ronnye, who has been so understanding over the past two years.

BLANK PAGE

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF ILLUSTRATIONS

CHAPTER 1
INTRODUCTION

This thesis deals with various aspects of three-dimensional scene description by computer. The Stanford Artificial Intelligence Project has provided the environment and facilities for this work. A number of people at the project have been investigating the general problem of coordinating perceptual and motor processes under computer control.

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 must contain at least the identity and location of each object. Our work has largely been concerned with techniques and programs for generating such descriptions from a single view. The programs are designed to function as part of a larger Hand-Eye system. A Hand-Eye system is a problem solving system with an eye (camera) for input and a hand (manipulator) for output. The visual portion of this system will be described briefly in this chapter and in more detail in Chapter 4.

Our work was originally motivated by an interest in the research of Roberts [33] and Guzman [12,13]. Although many of our techniques are similar to those which they have described, our view of machine perception as a process of generating and testing hypotheses is significantly different. Our primary goal has been to understand how various constraints and models can be applied in the interpretation of line drawing data.

Analysis and Synthesis Based on Models:

Any system capable of interpreting its input data 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 actually see the back 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 perspective projection.

Consider a set of models which are acceptable as input to a predictor. This means that 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.

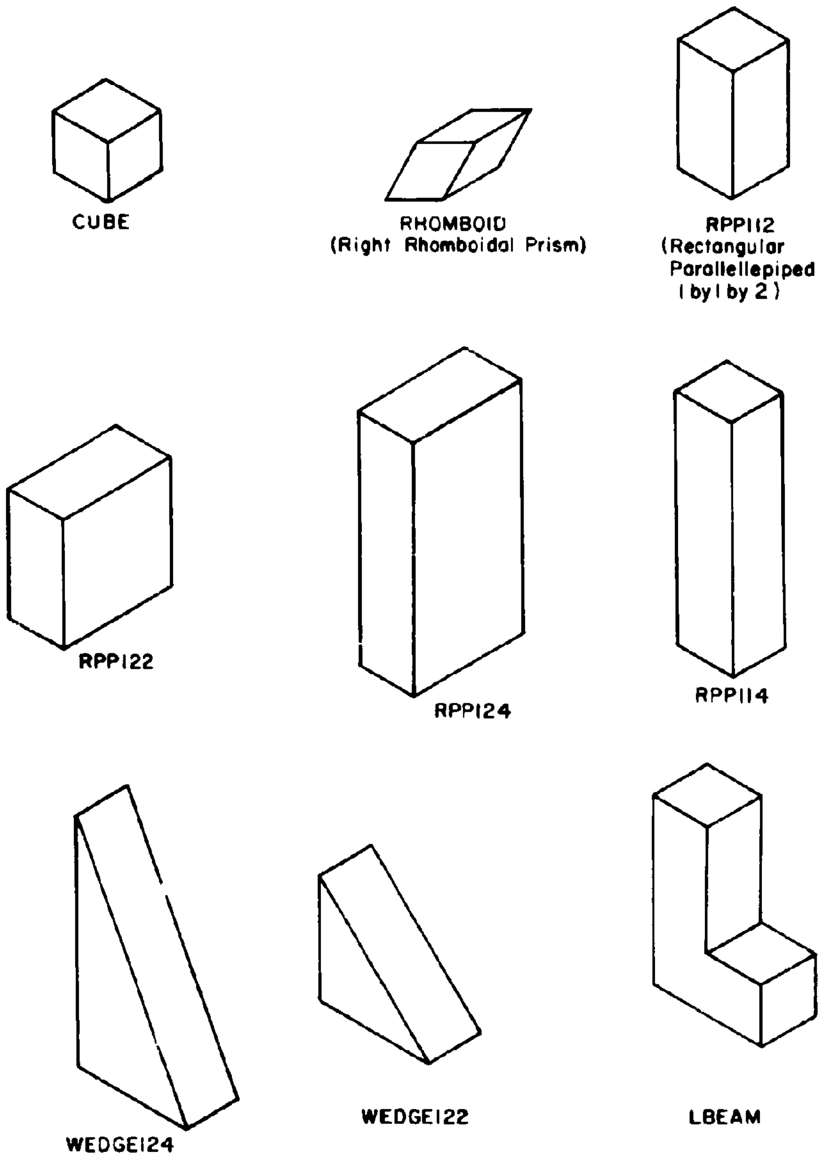
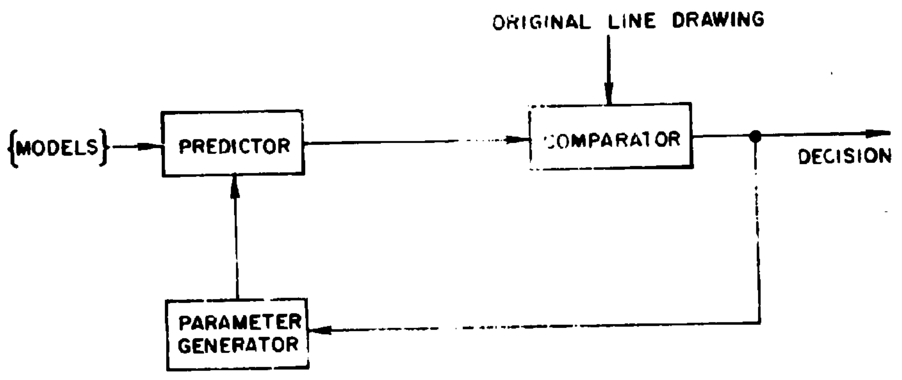

CUBE
RHOMBOID (Right Rhomboidal Prism)
RPP112 (Rectangular Parallelepiped 1 by 1 by 2)
RPP122
RPP124
RPP114
WEDGE124
WEDGE122
LBEAM

Figure 1-1 . Prototypes .

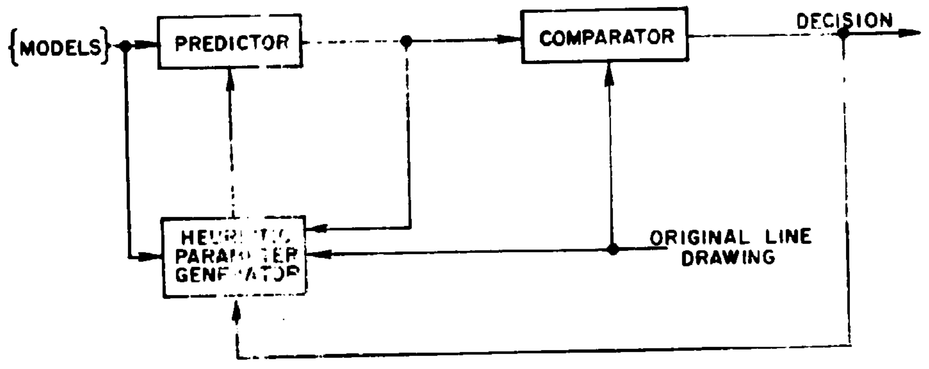
Assume for simplicity that each of these parameters (variables) is discrete.

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 predictor matches the input scene. The complexity of the predictor, of course, depends on the nature of the models and our interpretation of the word "matches" above. For the particular scene description system that we have implemented, the models are the structural descriptions of the solids shown in Figure 1-1. The variables to be specified consist of the number of objects present in the scene (M), the identity of the ith model ($1 \le i \le M$), and the 6 translational and rotational degrees of freedom for the ith model ($1 \le i \le M$). The problem is to assign values to these variables such that the predicted line drawing is identical with the original line drawing given as input.

The most direct way of interpreting the scene, given that bounds on the parameters are known, is by synthesis. We exhaustively assign each possible value to each parameter and check for a match between prediction and input for each such assignment (Figure 1-2a). Specifically, we begin by assuming that there is only a single object present in the scene. For each model we try all possible values of translation and rotation until either a match is found or all possibilities have been exhausted. If a match occurs,



EXHAUSTIVE INTERPRETATION
(a)



HEURISTIC INTERPRETATION
(b)

Figure 1-2 . Interpreting a Scene by Synthesis .

we assume that we have correctly interpreted the scene. If no match is found, we proceed by assuming that there are two objects present and we try all possible parameter values for each possible pair of models. Continuing in this fashion, that is, assuming 3, 4, ... objects present, the process will eventually terminate with a correct interpretation of the scene.

This approach to analysis by synthesis can be classified as a model-driven scheme. Although the procedure is grossly inefficient, it can be made relatively insensitive to errors in the input. One only has to design the matcher (comparator) to tolerate some discrepancies between the prediction and input. In contrast to this model-driven or top-down approach is the data-driven or bottom-up approach. In data-driven analysis local properties of the input are used to build a global interpretation of the scene. Although generally more efficient than top-down analysis, such procedures tend to be sensitive to noise. For example, in a scene which is known to consist of only a single object, one might infer from a triangular region that the object viewed is a wedge. The object may actually be a cube, however, with the triangular region a result of noise in the input. Decisions based only on local evidence tend to be risky in general.

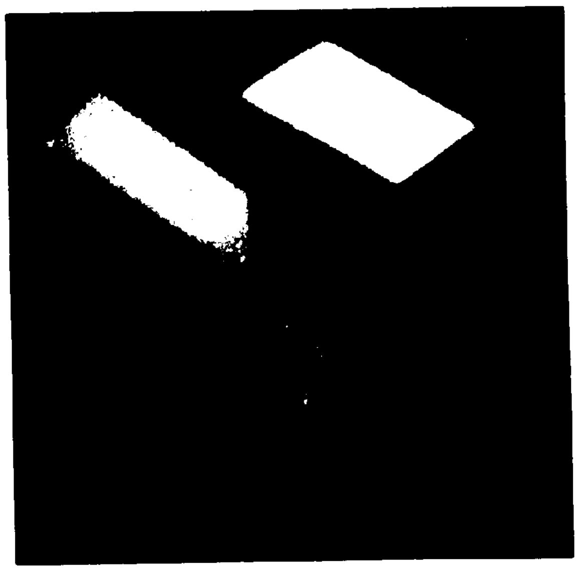
It seems clear that a combination of the model-driven and data-driven approaches is needed to efficiently process real world scenes. The approach which we propose consists of a heuristic hypothesis (parameter) generator and an algorithmic predictor-comparator. "A heuristic is a rule of thumb, strategy, trick, simplification, or any other device which drastically limits search for solutions in large problem spaces" ([5] page 6). If the heuristic generator produces an incorrect hypothesis due to an invalid assumption or over-simplification, the predictor-comparator can detect this error and request another hypothesis .

The "hypothesis and test" approach is a basic tool often used in solving complex search problems which arise in the field of artificial intelligence. The success of this approach depends on the cleverness of the hypothesis generator. As indicated in Figure 1-2b, the heuristic parameter generator will generally revise its parameter values so as to reduce any differences between its prediction and the original input.

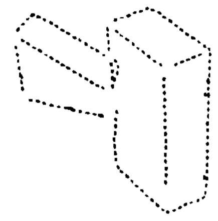
We view our model-based scene description scheme as an attempt to heuristicly search the tree of possible parameter values. The segmentation procedure described in Chapter 5 can be thought of as a group of heuristics designed to determine M. In the process we also generate M partial descriptions of the individual bodies. These plus some heuristic range estimates are used to determine the parameter specifying the identity of each object. The additional application of a few matching heuristics then permit the 6 translational and rotational degrees of freedom to be determined. If a failure occurs during the final comparison of the prediction with the input, or if a parameter cannot be assigned due to an earlier incorrect parameter assignment, earlier decisions can often be re-made. The heuristics are designed to limit the search space without excluding the correct answer.

Analysis of a Simple Scene:

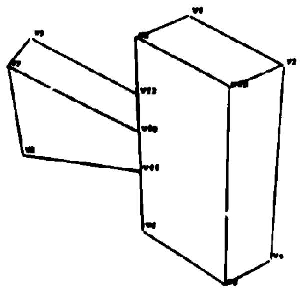
Analysis of a scene consists of transforming and abstracting the information in the input. The camera monitor in Figure 1-3a displays a scene to be analyzed. This image is read into memory and stored as a 333x256 matrix of intensity values. Each element in the array represents the brightness (0-15) at a point in the field of view. An edge-detector program transforms Figure 1-3a into the set of edge points shown in Figure 1-3b 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 1-3b into the line drawing shown in Figure 1-3c. This is accomplished by fitting straight lines to the edge points, extending these lines to form corners, and identifying closed regions. Our scene description programs



(a)



(b)



(c)

Figure 1-3 . A Simple Scene .

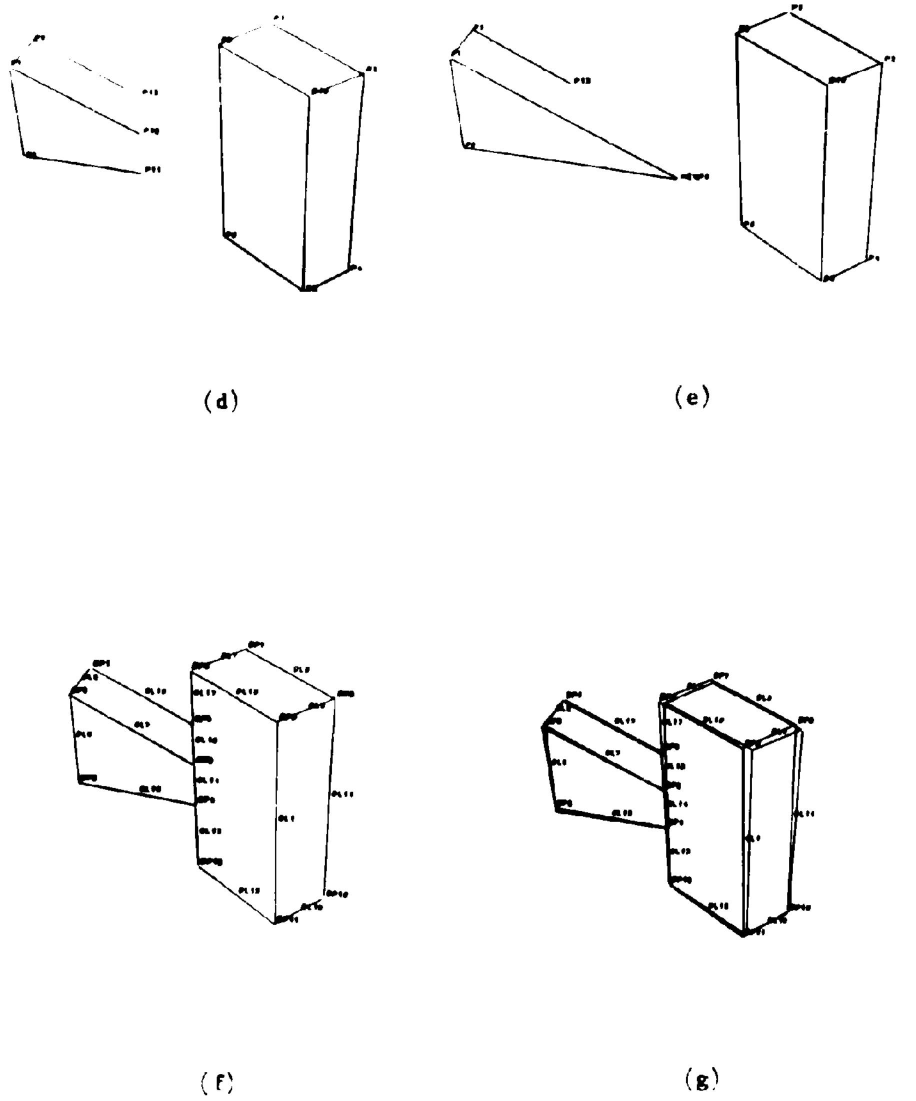


Figure 1-3 . A Simple Scene.

are designed to accept this line drawing as input. This thesis does not discuss preprocessing. Nevertheless, we are concerned with the quality of output that a preprocessor is likely to produce.

It is assumed that the set of objects is completely specified. The scenes are required to consist of only one or more of the objects shown in Figure 1-1. This particular set of objects 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 the objects are parallelepipeds (e.g. the wedges),

(4) Not all of the objects are convex (the LBEAM).

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

Our fixed size models are somewhat less general than the ones described by Roberts. Whereas his "cube" model represented all rectangular parallelepipeds, we need a separate prototype for each physically different solid. The discrete size restriction, however, provides an additional set of constraints. These constraints can be applied to resolve ambiguities that arise due to occlusion and noise.

Analysis of the line drawing proceeds in several stages. We assume that the camera has initially been calibrated (see Chapter 3 and the Appendix). The line drawing in Figure 1-3c is first segmented into pieces corresponding to individual bodies (Figure 1-3d). In cases where a body can be (partially) completed, the program does this (Figure 1-3e). 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 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 1-3f shows this prediction. Finally, the prediction and the original image (line drawing) are compared. If, as in Figure 1-3g, the two are approximately the same, the program assumes that it has correctly interpreted the scene.

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

The remainder of this thesis is divided into seven chapters. Chapter 2 describes past work related to ours. Chapter 3 considers the general problem of determining three-dimensional information from one or more two-dimensional images. The major portion of this thesis, Chapters 4, 5, and 6, describes a particular computer vision system that has been implemented. Chapter 4 presents an overview of this system with emphasis on the scene description programs. Chapter 5 describes in detail our programs which infer the structural relations in a scene. Chapter 6 presents the specific techniques that we use for object recognition and hidden-line elimination. In Chapter 7 the performance of the scene description programs is examined on a number of examples. Finally, Chapter 8 summarizes the results of our work and indicates areas we believe worthy of future investigation.

BLANK PAGE

CHAPTER 2

RELATED WORK

PICTURE PROCESSING AND PATTERN RECOGNITION

Rosenfeld [35] divides the problems of picture processing into three broad areas: encoding and approximation; filtering, restoration and enhancement; and pattern recognition and picture description. In this section we consider only the last area. Details and references into the literature concerning the other two areas can be found in Rosenfeld's book.

There has been an enormous amount of research in the area of two-dimensional pattern recognition during the last 15 years. The brief descriptions that we shall present below are, therefore, unquestionably superficial. We include them primarily to contrast these investigations with our own.

Classical Pattern Recognition:

Classical pattern recognition is concerned with classifying input patterns (pictures) into picture classes. For a given picture, P, a feature extractor, λ, operates on P to produce a feature vector, x, that is

$x = \lambda (P)$.

A decision function, D, is then applied to x such that

$$D(x) = \begin{cases} \geq 0 & \text{if } P \in C1 \\ < 0 & \text{if } P \in C2 \end{cases}$$

where C1 and C2 are the two picture classes. The two problems involved here are (1) How to choose λ? and (2) How to choose D? These questions are not, of course, independent.

To date most pattern recognition research has been concerned with the design of decision functions and has assumed that somehow the feature vectors have already been obtained. The type of data available determines how D can be computed. Four major forms which the data may take are:

(1) $p(x \mid Ci, \beta)$ is known, that is, we know the conditional density functions to within the specification of a set of parameters β.

(2) $p(x \mid Ci)$ is known, that is, the conditional density functions are specified completely.

(3) a set of training samples, T, such that for each $t \in T$ we know whether $t \in C1$ or $t \in C2$.

(4) a set of unclassified training samples.

These techniques have been applied to the recognition (classification) of such things as printed characters, blood cells, and speech. Nagy [26] and Ho and Agrawala [16] present a more adequate summary of the results and research in this area.

Picture Analysis Using Linguistic Techniques:

The techniques described in the previous section are ineffective in the analysis of highly structured pictures. For these pictures a description of the interrelations among the parts is required in addition to a categorization of the primitive components. The well developed techniques for the syntactic analysis of formal languages provide a powerful tool which can sometimes be effectively applied.

In formal language theory sentences are composed of strings of symbols. The two-dimensional relationships that exist between picture primatives, on the other hand, are considerably more complex than simple juxtaposition. Much work has been spent on generalizing the notion of concatenation so that interesting classes of pictures can still be described as strings of primatives. Recently, some two-dimensional "web grammars" have also been investigated.

Swain and Fu [7] classify linguistic pattern recognition techniques into those which are syntax-directed and those which are syntax-controlled. A syntax-directed procedure is one that simply has the goal of producing a syntactic description of the picture. A syntax-controlled procedure actually uses some form of grammar to direct the analysis.

The work of Shaw [36] is an example of the syntax-controlled approach. He has described a picture description language, PDL. The primatives of PDL can be any pattern having two distinguished points, a "head" and a "tail". PDL can describe any concatenation among the set of primatives. A class of pictures is defined by means of a restricted form of context-free grammar, G, generating sentences in PDL.

Given a picture, p, a picture parser (analyzer) uses

(1) a specification of the grammar, G and

(2) a recognizer for each primative in G

to determine if $p \in P(G)$. The important side-effect of successful recognition is the generation of D(p), a structural description of the picture p.

This technique is advantageous because it is goal-directed (top-down, model-driven). The grammar, G, directs the primative recognizers over p. Failures by the primitive recognizers can often be resolved by contextual information embedded in G.

Shaw applied his system to the analysis of spark chamber photographs. Other systems using the linguistic approach have been applied to character recognition, chromosome analysis, and analysis of mathematical notation. A recent survey article by Miller and Shaw [23] presents a more complete description of Shaw's approach and references other relevant literature.

THREE-DIMENSIONAL SCENE DESCRIPTION

In contrast to the vast amount of research that has gone on in the area of two-dimensional pattern recognition, relatively little has been done in the area of three-dimensional scene description. The emphasis of work in each of these two areas has been totally different except for a number of common preprocessing techniques. A forthcoming book by Duda and Hart [4] is the first attempt that we know of to treat both of these topics under one cover. Perhaps these separate efforts will merge when more is understood concerning the mechanization of perception. The scene description techniques of Roberts and Guzman described below are the main ones that have been reported on to date.

Perception of 3-D Solids: Roberts

The work of Roberts[33] is a classic in the area of machine perception. We shall only describe his method for identifying and locating objects in space. The system described in his thesis did considerably more than this, including preprocessing and display generation.

Assume that an instance of a particular prototype is resting on the table and being viewed with a camera. If the model vertices V1, V2, ... Vn are expressed using homogeneous coordinates as column vectors (see [33,1]), and if VJ' is the instance vertex corresponding to VJ of the model, then there exists a 4x4 positioning matrix T which translates and rotates the model such that

$$VJ' = T\ VJ \qquad J=1,\ 2,\ \ldots\ n.$$

Furthermore, if the camera has been properly calibrated, we can determine a projection matrix P such that

$$VJ'' = P\ VJ'$$

where two of the non-homogeneous coordinates of VJ'' are the image coordinates of the point corresponding to VJ' . The matrix H=PT, therefore, takes model points into image points. If for a given image model pair there exists a transformation H, then the image could be a projection of an instance of the model under the transformation $T=P^{-1}H$.

The Roberts system did a topology match between the projection points and model points in order to rule out obviously incorrect models and to set up the vertex correspondence necessary to compute H. By means of a "similarity test" it then derived for each potential model the best H taking the model points into the image points (in the mean square error sense). Finally, it chose the model which minimized this error. The position and orientation of an object could be determined up to a depth factor by this method. This final degree of freedom was specified by assuming the object to be supported by the "ground-plane" a known distance below the camera. Methods for handling compound objects composed of several model primatives and partially occluded (eclipsed) objects are found in his report.

The scenes which Roberts analyzed were all quite simple and the line drawings were ideal in the sense that they had no missing lines. We have attempted to design a system without these limitations.

Segmentation of 3-D Scenes: Guzman

In his Masters thesis Guzman [12] described several approaches (the programs POLYBRICK, TD, and DT) for identifying objects present in a scene. The line drawing input, however, was restricted to noise-free orthogonal projections. More recently [13] he has described his well known segmentation procedure (the program SEE) for isolating the individual bodies present in a visual scene. The idea is basically to take a line drawing such as the one in Figure 2-1a and separate it into two parts as in Figure 2-1b. This is accomplished by using local evidence accumulated at the individual vertices to determine which closed regions of the scene should be identified as belonging to the same body. Guzman's approach is interesting because it does well using only a few simple heuristics on extremely complex scenes. These scenes may contain arbitrary planar-faced solids.

The following is a somewhat simplified description of how SEE would work on the line drawing of Figure 2-1a. It begins by setting up a graph where the nodes of the graph correspond to regions in the projection. Links are set up between the nodes based on the types of vertices where the regions meet. For example, a FORK type vertex implies 3 links between the faces (nodes) meeting there as shown in Figure 2-1d. Similarly, an ARROW type vertex implies 1 link (Figure 2-1e), and a TJOINT vertex (Figure 2-1f) implies no

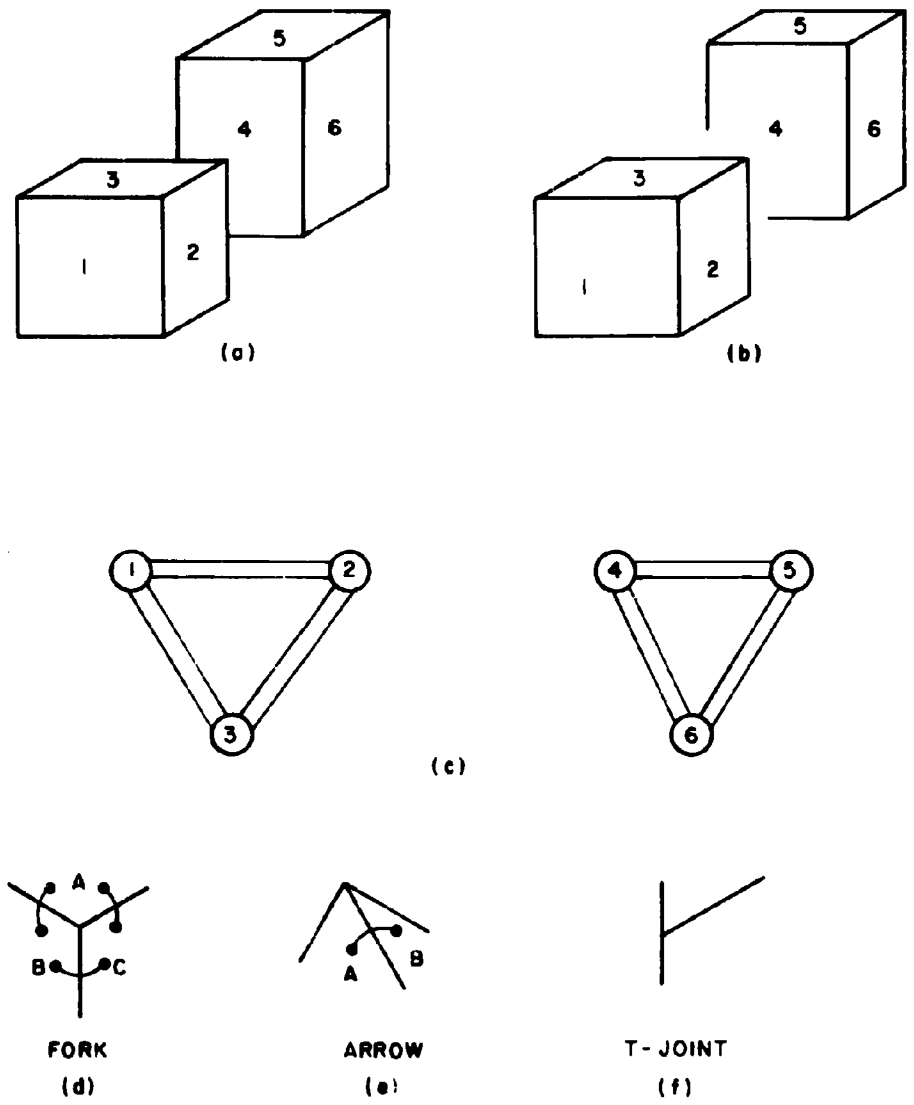


Figure 2-1 . Guzman's Segmentation Scheme .

links. T-joints generally occurs due to one body occluding another. The resulting graph is then merged according to the following rule: 2 nodes are merged if there exist 2 or more links between them. At the end of this merging the resulting nodes (called nuclei) consist of those faces that should belong to the same body.

There are several liberties that we have taken in the description of Guzman's algorithm as well as several details and refinements that we have simply omitted. Nevertheless, the above example does convey the idea of his approach. One problem with the approach is that it is sensitive to certain forms of error, i.e., certain lines if missing from the line drawing cause difficulty. We shall describe an alternate segmentation algorithm in Chapter 5 that does not have this problem.

Recently Huffman [18] has attempted to formalize and extend some of Guzman's ideas. The part of this work that has been reported concerns the use of constraints in the interpretation of ambiguous and contradictory line drawings. Huffman is interested in determining if a given line drawing could be the projection of any of a restricted class of planar-faced solids. We shall have more to say about his techniques in Chapter 3.

ROBOT PROJECTS

Over the past 5 years several robot projects have grown up at MIT's Project MAC, at the Stanford Research Institute, and here at the Stanford Artificial Intelligence Project. All of these have been directed toward investigating the coordination of perceptual and motor processes under computer control. Both at MIT and at Stanford the major effort has been to develop Hand-Eye systems [24,6]. At SRI the emphasis has been on the visual control of a motorized vehicle [28]. A more recent effort at Stanford has also been concerned with the visual control of a vehicle. Although many of the problems that these three groups face are similar, there has been surprisingly little duplication of effort. Much experience has been gained from taking alternate approaches to common problems.

Guzman's work described above developed out of MIT's Hand-Eye project. Our own work, of coruse, has been motivated by the Hand-Eye project at Stanford.

A recent thesis on the subject of learning by Patrick Winston of MIT has several aspects in common with our own research. His motivation, however, was somewhat different from ours. He has been investigating the learning of structural descriptions of scenes based on examples. The scenes consist of simple geometric solids. We shall discuss the generation and use of structural information about scenes in Chapters 4, 5, and 6.

RELATED RESEARCH IN COMPUTER GRAPHICS

Much of the work done recently in the area of 3-D computer graphics is particularly relevant. Machine perception might aptly be referred to as "graphics in reverse". Whereas graphics is concerned with the display of images of physical objects, machine perception is concerned with inferring 3-D structure from images. In each case, the problem of internally representing a solid becomes important. A review of some of the data structures used to describe real world objects is given by Gray [10]. The application of homogeneous coordinates to simplify geometric manipulation has recently been reviewed by Ahuja and Coons [1]. Finally, we should mention the work of Warnock and others [42,34] at the Unversity of Utah concerning the efficient generation of high quality half-tone renderings of three-dimensional solids.

RELEVANT PSYCHOLOGICAL STUDIES

People can easily interpret a line drawing as a three-dimensional scene. It seems reasonable, therefore, that psychological theory might suggest a means of mechanizing this process. Although we have not found

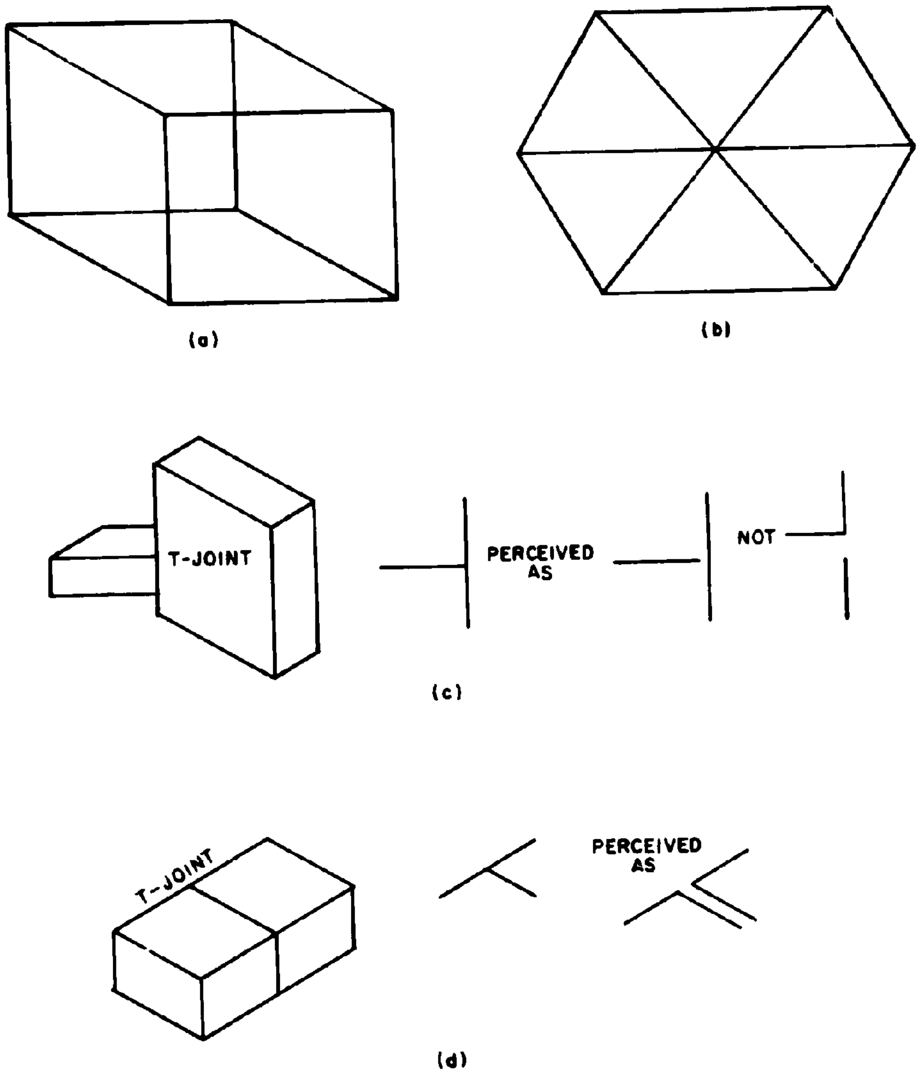


Figure 2-2 . Maximum Simplicity and Good Continuation this approach particularly fruitful, it is interesting to contrast some of our techniques with their human counterparts.

We believe that our general approach has some psychological basis. According to R.L. Gregory "... we do not perceive the world merely from the sensory information available at any given time, but rather we use this information to test hypotheses of what lies before us. Perception becomes a matter of suggesting and testing hypotheses...The continual searching for the best interpretation is good evidence for the general importance of augmenting the limitations of the senses by importing other knowledge" ([11] pp. 222-223). What "other knowledge" people have available or precisely how they generate and test hypotheses is not well understood.

Several vague Gestalt "laws of organization" indicate some of the ways that people organize or interpret visual data. For example, the "law of maximum simplicity" says that people tend to interpret an ambiguous situation in the simplest possible way. The drawing in Figure 2-2a is most often perceived as a three-dimensional wire cube. Although the drawing of Figure 2-2b is also a possible projection of a wire cube, it is most simply perceived as a plane figure. Closely related to this law is the "law of good continuation". Figure and background tend to be perceived in a way which minimizes the interruptions of straight or smoothly curving lines. The T-joint in Figure 2-2c is perceived as one edge hiding another although this interpretation may be modified as in Figure 2-2d due to other global information.

Many of the standard depth cues have parallels in our scene description system. These include eclipsing of one object by another, perspective, familiar size, relative upward location in the field of view, convergence, and accomodation. Examples and discussion of all the preceding issues can be found in an interesting book by Hochberg [17].

BLANK PAGE

CHAPTER 3

OBTAINING 3-D INFORMATION FROM 2-D IMAGES

Object recognition is accomplished by matching features in the line drawing projection against features of the prototype. Since the protypes are three-dimensional structures, comparison of geometric properties can proceed only after three-dimensional information has been inferred from the two-dimensional image. In this chapter we consider several ways in which this can be done.

First we describe the picture-taking process and a simplified camera model. Based on this model we briefly describe two methods (stereo ranging and focus ranging) for determining the 3-space location of individual points. These methods are quite costly to apply and motivate the heuristic methods described next. These techniques (including the well known "support hypothesis") are used to determine the 3-space location of individual points based on various additional assumptions. The major portion of this chapter, however, considers the constraints which exist among the points in a projection of a planar-faced solid.

THE PICTURE TAKING PROCESS

Figure 3-1 depicts a first-order approximation to the picture taking process. For any point Pj=(Xj,Yj,Zj) in the real world there is a unique corresponding point Pj'=(Xj',Yj') in the image. It is not possible in general to determine a unique point in 3-space corresponding to a specified point in the image although each picture point does have an associated ray. These rays can be determined as functions of the x,y,z coordinates if the camera is initially calibrated with respect to the real world (table). Our approach determines a 1-1 mapping (collineation) between the image plane and the plane of the table top and locates the camera (lens center) C. We can express the ray associated with any point PI parametricly as R(t)=tC+(1-t)PT where PT is the table point into which PI maps. A more detailed description of this calibration is given in the Appendix.

The above model is valid as long as the camera is not moved. If the camera is moved to a new position, perhaps to look somewhere previously out of the field of view, then the system would need to be recalibrated. In a recent dissertation, Sobel [37] describes a method that yields a camera calibration parameterized by pan, tilt, focus, and lens. These parameters are read by the computer from potentiometers attached to the camera. His calibration

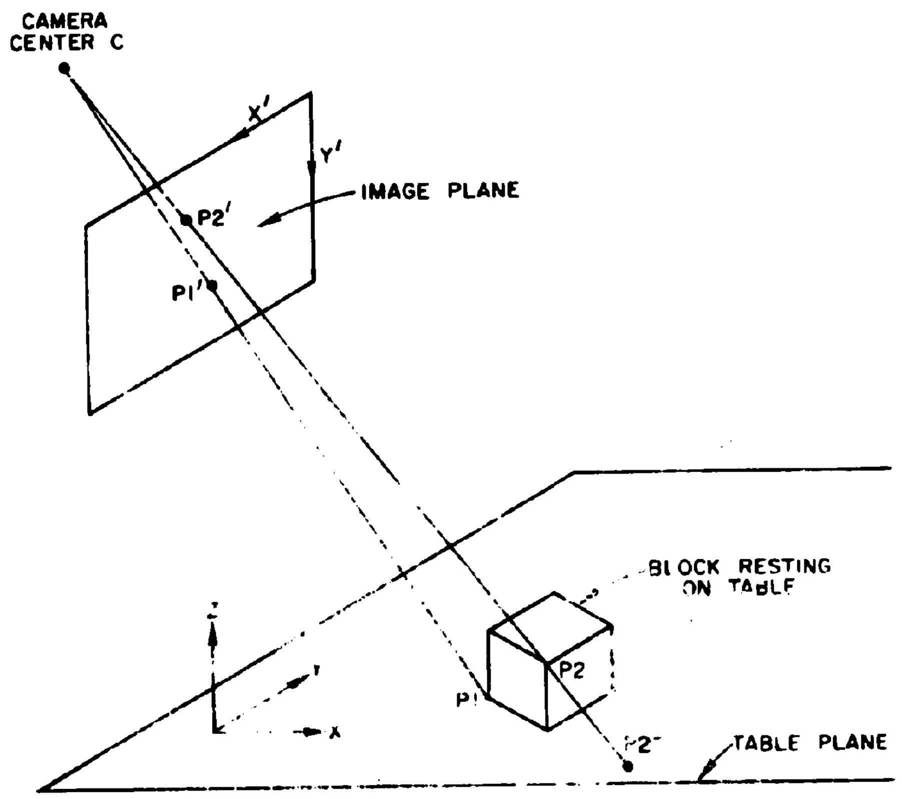


Figure [illegible]-1 . The Picture Taki[illegible] Process .

approach yields a more accurate and consistent camera model than the simple one described in the Appendix.

OBTAINING 3-SPACE POINT LOCATIONS

Stereo Ranging:

It is well known that a point viewed in two distinct projections can be located in space by triangulation if the cameras have initially been calibrated. In Figure 3-2 point P must lie along RAY1 based on IMAGE1 and must lie along RAY2 based on IMAGE2. The intersection of these two rays in space specifies P=(X,Y,Z).

There are essentially two parts to the stereo ranging problem, the correlation problem and the triangulation problem. The correlation problem consists of finding for a given point in IMAGE1 the corresponding point in IMAGE2. The triangulation problem is simply the geometric problem of intersecting two rays. Sobel [37] has recently investigated both of these problems. He has found a way to simplify the search for a correlate point considerably. He has also been quite concerned with the effects on triangulation of both noise in the data and imperfections in the camera model.

An alternate approach to using two cameras, is what we shall call "lazy susan stereo". To obtain two distinct

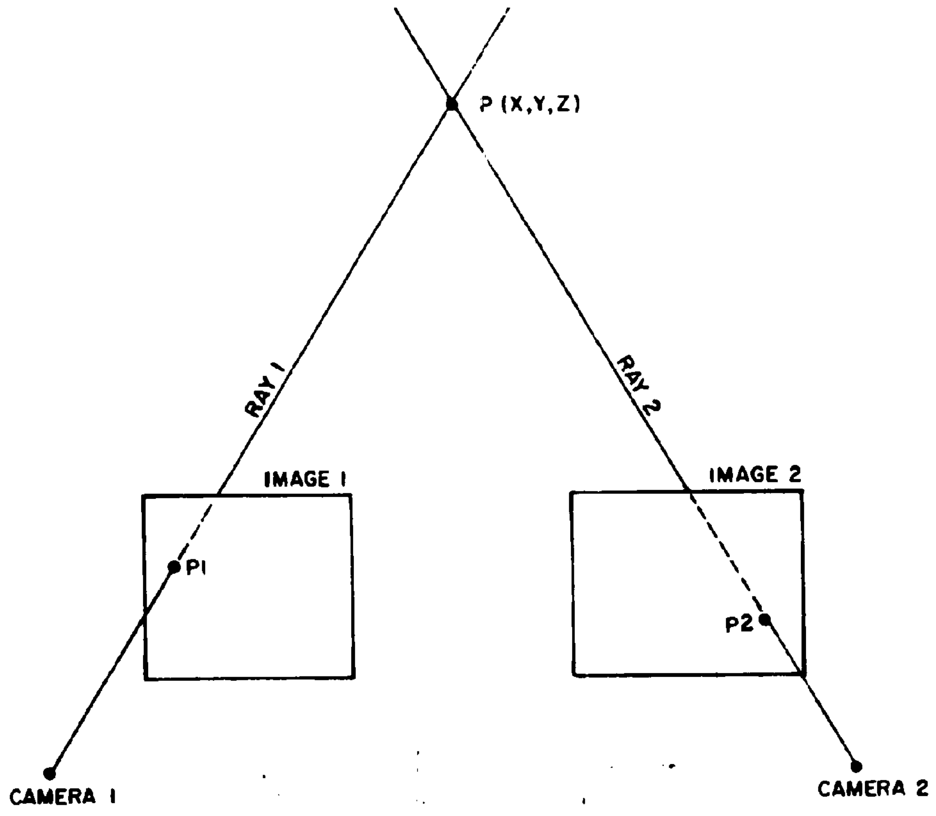


Figure 3-2 . Stereo Ranging .

projections we use a single camera and move the object. Our work table has a large circular section which can be rotated under computer control. This was primarily designed in order to add an additional degree of freedom to the arm and to make available different views of the scene. Since the disc can be controlled quite accurately, it is also possible to rotate it only a few degrees and duplicate narrow-angle stereo ranging. This method has the advantage of being quite easy to vary the angle between the stereo pairs. To date this approach has not received much serious attention.

Focus Ranging:

Focus ranging is another way of getting three-dimensional information from a line drawing projection. This approach has been investigated by Tenenbaum [40] and will be described only briefly below. The scheme is based on the simple lens equation:

$$1/D_o + 1/DI = 1/f .$$

In the equation, f is the focal length of the lens, Do is the object distance, and DI is the image distance. The focal length of the lens is assumed to be known. If one can determine the image distance when a feature point is in focus, then one can solve for Do, the depth of the point along its ray.

Focusing is done under computer control by actually moving the vidicon tube (image plane) closer and farther from the lens. A potentiometer allows the vidicon location to be read by the computer. Tenenbaum has developed schemes to determine whether a particular feature is in focus. For very accurate depth calculations by this method, it may be necessary to change to a longer lens.

Support Hypothesis:

Support hypothesis, initially described by Roberts [33], is the first heuristic technique we shall mention for inferring three- dimensional information from a single view. Support hypothesis assumes that an object is not suspended in space, that is, it is supported either by the table or by other objects. If we can determine, by some means, which object corners rest on the table, then the 3-space location of these corners are specified from the collineation. In Figure 3-1 identifying P1' as a "table point" says that its actual coordinates can be found as:

$$P1' = (x',y') \xrightarrow{A} P1T = (x,y) \rightarrow P1 = (x,y,0) ,$$

The homogeneous representation of P1', (X1',Y1',1), is multiplied by the collineation matrix A. The two non-homogeneous coordinates of the resulting vector with a z component of zero is the location of P1 in the X,Y,Z system. The implementation of depth finding using support hypothesis both for objects resting on the table and objects supported by other objects is considered further in the next section and in Chapter 5.

Points In Known Planes and Along Known Lines:

Locating points by support hypothesis can be thought of in the following way: each image point specifies a ray along which the corresponding object point must lie. The intersection of this ray with the known table plane (z=0) then determines the actual point in space. An identical argument allows any image point to be located if a plane in which it lies is known. The situation is similar if we know a line (other than its ray) along which the object point lies. This problem is just that of stereo triangulation.

We have applied these methods extensively in the system described in Chapters 4, 5, and 6. Of particular interest are the cases shown in Figure 3-3. In Figure 3-3a table point B=(XB,YB,0) is specified and line BP is known to be vertical (i.e. normal to the plane z=0). It follows that

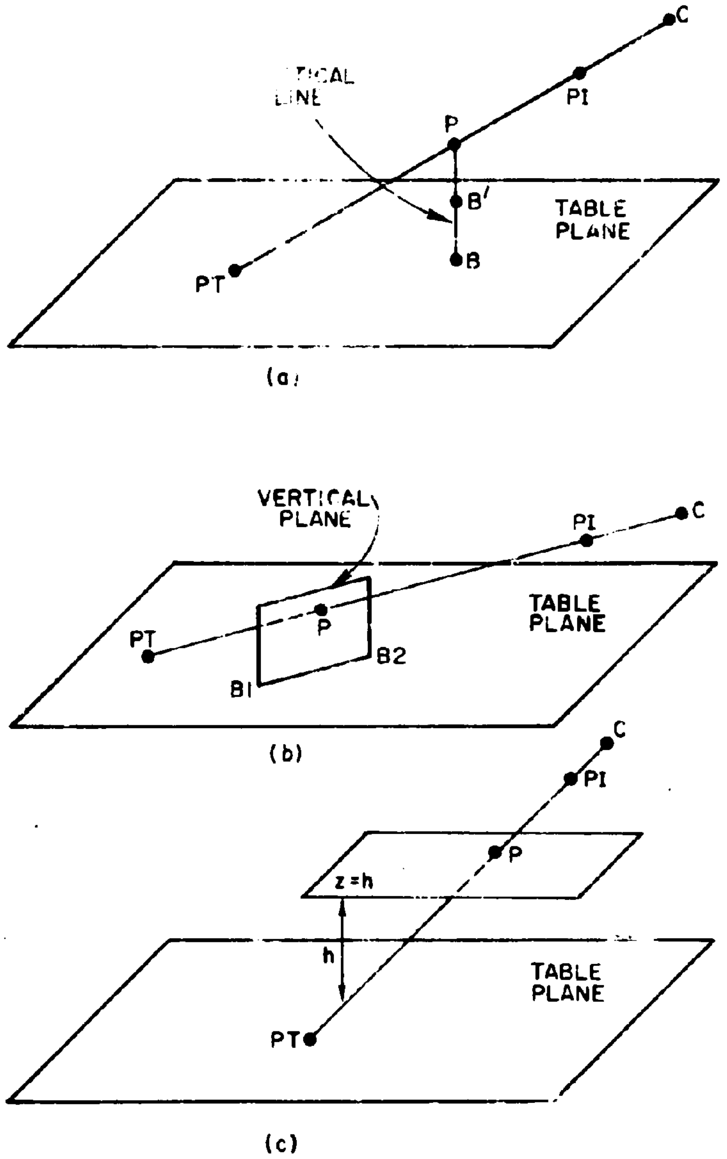


Figure 3-3 . Points in Known Planes and Along Known Lines .

P = INTERSECTION (BB' , CPT)

where PT = A PI

and B' = (XB, YB, 1.0).

In Figure 3-3b point P is known to lie in the vertical plane specified by table points B1=(XB1,YB1,0) and B2=(XB2,YB2,0). We can express points along CPT parametrically as:

$$[tX_c + (1-t)XT \ , \ tY_c + (1-t)YT \ , \ tZ_c \ ] \qquad (1)$$

where t=0 corresponds to PT and t=1 corresponds to C. Let PT' be the perpendicular projection of P onto then table plane (and onto the line B1B2). Points along B1B2 must satisfy

$$y = mx + b \qquad (2)$$

with m=m(XB1,XB2,YB1,YB2) and b=b(XB1,XB2,YB1,YB2). Substituting the first and second components of (1) into (2) and solving for t we get:

$$t=[mXT - YT + b]/[Y_c - YT - m(X_c - XT)].$$

This specifies P from (1).

The final case of importance is shown in Figure 3-3c where the unknown point P is assumed to lie in a known horizontal plane z=h. In this case we find:

$$X_p = (h/Z_c)(X_c - XT) + XT$$

$$Y_p = (h/Z_c)(Y_c - YT) + YT$$

$$Z_p = h$$

It is often possible to guess that a certain line or plane is vertical or horizontal and check that the assumption is consistent with the known set of prototypes. Consider, for example, the projection shown in Figure 3-4. Suppose that our program has identified the body as a RHOMBOID and has determined the 3-space locations of P1, P2, and P3 by assuming that thay lie in the table plane. For the class of objects shown in Figure 1-1, it follows that either plane P1P2P4 or plane P2P3P4 is a vertical plane. To identify the vertical plane the program proceeds as follows: (1) It assumes that P4 lies in the vertical plane specified by P1 and P2. (2) It determines the 3-space location of P4 as described above. (3) It compares the resulting length of edge P2P4 with the known edge lengths for the edges of the RHOMBOID prototype. (4) If a match occurs, the program concludes that its assumption about P1P2P4 was correct, otherwise (5) it determines the location of P4 assuming that

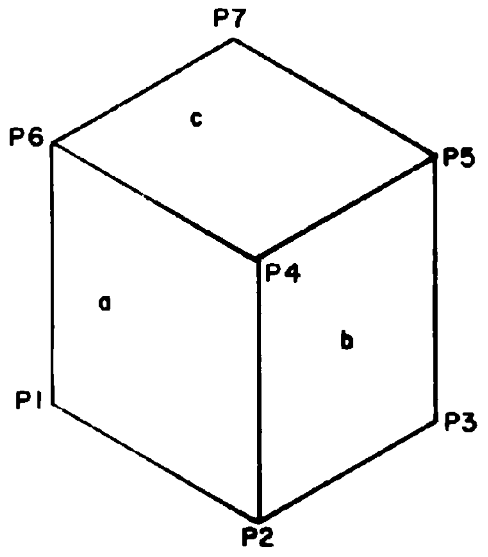


Figure 5-4 . A Block .

plane P2P3P4 is vertical. The difference between the prediction of P4 in (2) and the prediction of P4 in (5) is usually sufficient to locate P4 correctly.

CONSTRAINTS IN PROJECTIONS OF PLANAR-FACED SOLIDS

We have presented a model of the picture taking process and described several ways in which 3-D information can be inferred for individual points. We now consider the constraints which the location of some image points impose on the positions of others. From another point of view, we are interested in determining the amount of information that a single line drawing projection implies about the shape and position of the object viewed. The discussion in this section is not restricted to the set of objects in Figure 1-1.

Introductory Examples:

Consider once again the planar-faced solid shown in Figure 3-4 (we do not assume it is a RHOMBOID here). Assume the rays to all of the visible corners have been determined from monocular information as described previously. Also assume that points P1,P2,P3, and P4 are known in 3-space. . Since P1,P2, and P4 determine a plane and point P6 presumably lies in this plane, we can determine

P6 as the intersection of its ray and plane P1P2P4. A similar argument holds for determining point P5 from its ray and points P2,P3, and P4. Finally, P4, P5, and P6 specify the top face ,c, and point P7 can be located. For this simple example, the locations of a particular 4 points plus all of the rays specified the rest of the points uniquely.

As another variation on the same theme, consider the 4 points P1, P2, P3, and P7. Do the locations of these points in addition to all the rays specify the visible portion of the object uniquely? The argument that they do follows from the previous one. In terms of the ray coordinate of P4, call it t, we determine P5(t), P6(t), and P7(t). The variable t can then be determined since the value of P7 is known. This approach is analogous to the use of fictitious loop currents in electrical networks which if solved for determine all the actual currents .

As a final example of the constraints imposed by the assumption of planarity, we again consider the solid of Figure 3-4. As before, all the rays to the corners are assumed to be known. This time, however, the additional information is the 3-space location of only P7 and the facts that the plane of face a is vertical while the plane of face c is horizontal. Again we find that this information is sufficient to specify the visible portion of the object uniquely. There is clearly only one horizontal plane passing through P7. Knowing this plane, the rays to P4, P5, and P6 determine the 3-space locations of these points. There is only one vertical plane passing through P4 and P6, thus the points on face a can be determined. Finally from coplanar points P2,P4, and P5, point P3 can be determined. Had P7 not been located in 3-space, there would remain one unspecified degree of freedom. It is possible to convince oneself that this can be interpreted as either not knowing the size of the object, or not knowing its position. The shape of the object, however, would be specified. The subsequent discussion is an attempt to formalize and generalize on these examples.

Constraints:

Let us try to explicitly state the constraints implicit in Figure 3-4. These consist of the facts that points P1,P2,P4, and P6 lie in plane a, points P2,P3,P4, and P5 lie in plane b, and points P4, P5,P6, and P7 lie in plane c. Any plane, d, can be expressed as

$$d1x + d2y + d3z = \alpha$$

where point P=P(x,y,z) is any point in the plane d. If we divide through by α ($\alpha \neq 0$ for any face we can see as we are assuming here that the origin of the coordinate system is located at the lens center) and express the above relation as a dot product we get

$$D \cdot P = 1 \quad \text{where} \quad D = (D1, D2, D3) = (d1/e, d2/e, d3/e),$$

Knowing the ray along which point P lies is equivalent to knowing the unit vector in the direction of P. We represent Pi, therefore, as (βi)(Ui) where βi is a number to be determined and Ui is this unit vector. Now,

$$D \cdot Pi = (\beta i)(D \cdot Ui) = 1 \quad \text{or}$$

$$D \cdot Ui = 1/\beta i = \lambda i \quad \text{or}$$

$$D \cdot Ui - \lambda i = 0.$$

The "point-plane incidence constraints" for Figure 3-4 can, therefore, be written as:

$$A \cdot U1 - \lambda 1 = 0$$
$$A \cdot U2 - \lambda 2 = 0$$
$$A \cdot U4 - \lambda 4 = 0$$
$$A \cdot U6 - \lambda 6 = 0$$

$$B \cdot U2 - \lambda 2 = 0$$
$$B \cdot U3 - \lambda 3 = 0$$
$$B \cdot U4 - \lambda 4 = 0$$

$B \cdot U5 - \lambda 5 = 0$

$C \cdot U4 - \lambda 4 = 0$

$C \cdot U5 - \lambda 5 = 0$

$C \cdot U6 - \lambda 6 = 0$

$C \cdot U7 - \lambda 7 = 0$

These constraints form a system of 12 simultaneous linear equations in 16 unknowns ($\lambda 1 \ldots \lambda 7$, A1, A2, A3, B_1, B2, B3, C1, C2, and C3). If the rank of the system is r ($r \leq 12$), then all of the unknowns can be determined up to a scale factor in terms of $k=16-r$ parameters. This scale factor is determined by knowing the object size or the 3-space location of any corner. In particular, knowing k independent point locations specifies the visible portion of the solid uniquely. By independent points we mean simply that their ray coordinates can be assigned arbitrarily with respect to the above constraints . Four points on the same face are an example of a set of dependent points.

The technique described above is not restricted to the example of Figure 3-4. So long as the only constraint is of the form that a point is required to lie in a plane, this approach is applicable. The equations resulting from the constraints will not necessarily be independent, but well known techniques (i.e. Gauss-Jordan Reduction) can be used to determine the rank of the system. We do not claim that the point-plane incidence relations necessarily exhaust all the constraints inferable from a projection. In Figure 3-5 A, for example, since the planes corresponding to regions 1 and 2 can intersect along only one line, we may conclude that any 3 points in the set (P4,P5,P6,P7) form a dependent subset. That is, two points determine the line P4-P5-P6-P7 and this plus the ray to any point along the line specifies it uniquely. The facts that P6 and P7 are functions of P4 and P5 could be added explicitly at the expense of some complication. In what follows, however, we shall be concerned solely with the point-plane incidence constraints.

Specification of Trihedral Solids:

Consider the projections shown in Figure 3-5. For projection 3-5a one can determine all the visible corners in 3-space using only the locations of corners P1,P2,P3, and P4 to augment complete monocular information (the rays to all visible corners). The argument follows those given previously. For Figure 3-5b, however, 4 points are not sufficient as counting equation will show. Since there are only 28 equations and 33 unknowns, at least 5 points must be determined to specify the body uniquely. In this section we consider a particular class of solids and give conditions sufficient to guarantee that a non-degenerate projection of

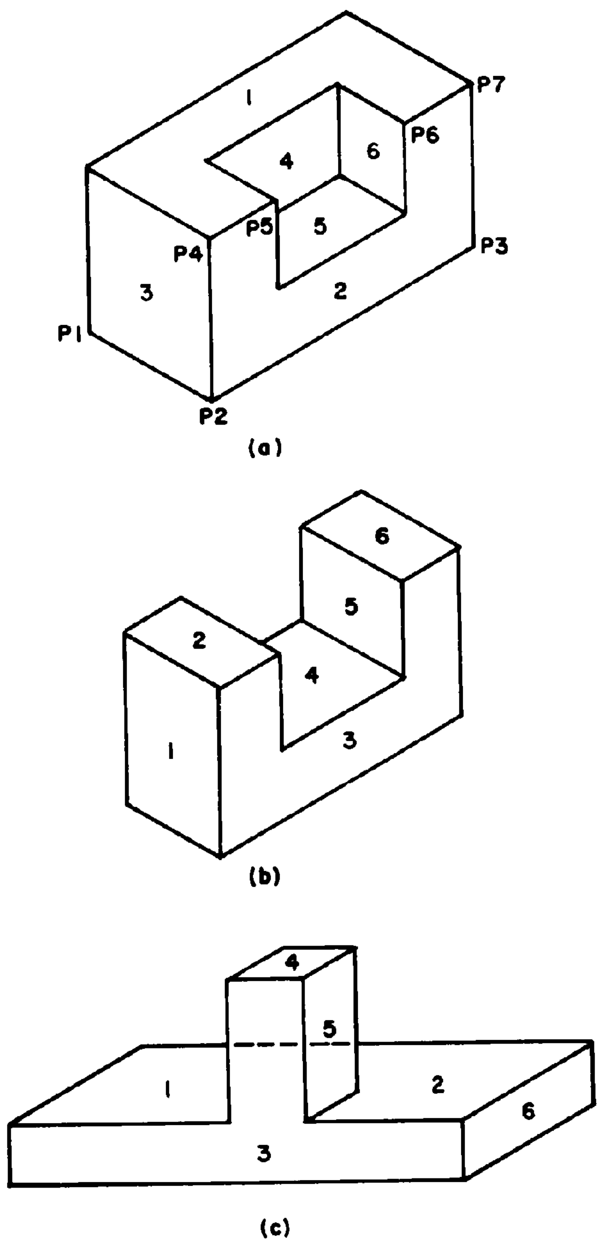


Figure 3-5 . Three Trihedral Solids .

one of these solids is specified by the locations of 4 independent points. The class that we shall consider are those solids for which exactly 3 planes meet at each corner. We shall refer to these objects as "trihedral (trilinear) bodies". By non-degenerate projection we mean only that the topology seen does not change if we move the object (or equivalently our eye) a small distance.

Let R=(R1,R2,...,Rm) be the set of simple closed regions in a given projection P. Let M be the mapping from R to the set of faces of the corresponding solid. In general, M is into and many to one (e.g. Figures 3-4, 3-5a, 3-5b, and 3-5c). We define the "face adjacency graph" for projection P, G(P) (or simply G), to be the undirected graph with each Ri∈R as a node of G and an edge between node Ri and node Rj iff (1) region Ri and region Rj have a common bounding line, L, in projection P, and (2) L corresponds to an edge between M(Ri) and M(Rj). The face adjacency graph for projection 3-5a is given as Figure 3-6a. We temporarily postpone the question of how G is determined.

For G and any graph derived from it we define two nodes to be "mergeable" if there are two or more edges between them which correspond to non-collinear lines in the projection . In Figure 3-6b node 1 and node (2,3) can be merged to form the new node (1,2,3) in 3-6c. We say that "graph G is mergeable" if by adding a single edge between some pair of adjacent nodes it can be reduced by a sequence

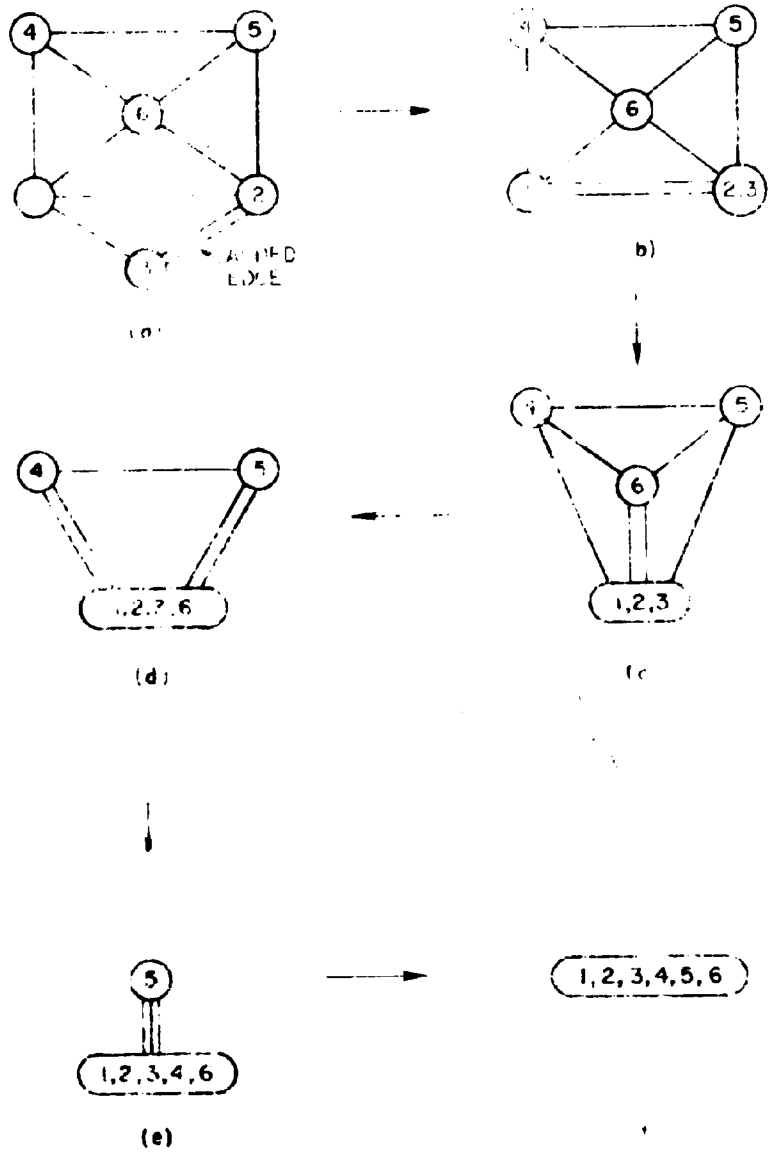


Figure 3-6 . Merging the Face Adjacency Graph of Figure 3-5a .

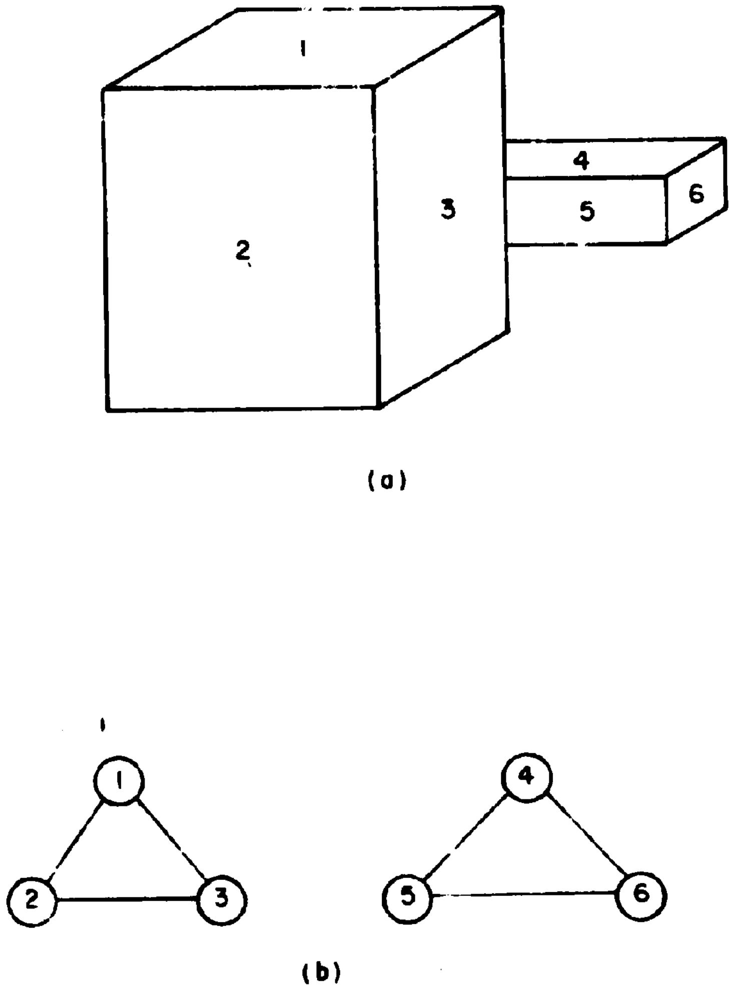


Figure 5-1 . An Unconnected Face Adjacency Graph .

of merges to a graph consisting of only a single node (the added edge is considered non-collinear with every line in P). We assume throughout the rest of the discussion that G(P) is connected, that is, it has no disjoint subgraphs. If this were not the case (see projection 3-7a and graph 3-7b) the projection might be better interpreted as several objects rather than one.

We can now state the connection between mergeability of G and the specification of a trihedral body from its projection as a

THEOREM: Given a non-degenerate projection, P, of a trihedral solid, if G(P) is mergeable then all visible vertices of the object can be located in 3-space by knowing the rays in space along which all visible points lie and the 3-space location of exactly 4 independent points.

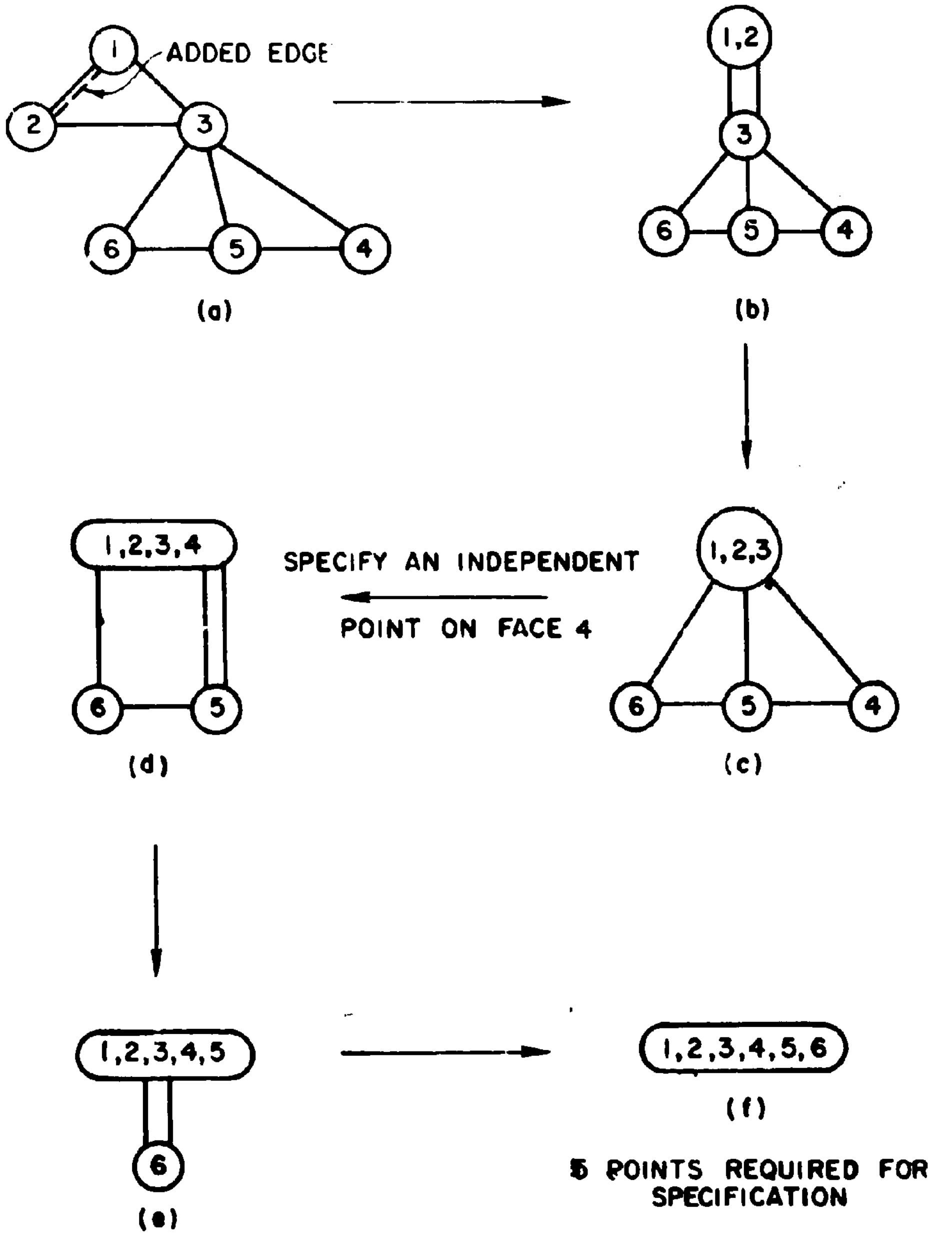
PROOF: First we prove that there are 4 particular points sufficient to specify the object completely. The results of the previous section then imply that any 4 points will suffice.

Assume that G(P) is mergeable and consider the two adjacent nodes between which the single link is added. Clearly, the face corresponding to either of these nodes can be specified (all its visible vertices determined) by locating exactly 3 independent points on it. Now the face corresponding to the adjacent node n be specified by locating only one additional independent point on it as two of its vertices are common to the one previously specified. Call the supernode into which these two nodes are merged K, the set of known faces. Since G is mergeable, either K includes every node and all the visible vertices have been specified, or there exist two or more links between K and some remaining unmerged node. If more vertices remain to be specified, the face corresponding to any one of those linked to K by at least 2 edges can be specified without locating any additional points as it must have at least 3 independent points (2 non-collinear edges) in common with those faces in K. This process of specification is guaranteed to continue until K=R if G(P) is mergeable. Since 3 points are clearly not sufficient (all points could actually be in the plane determined by these 3 points and the object only a picture itself), exactly 4 points are required.

To see that any 4 independent points will specify the object, we note that the only set of constraints applied above was that 3 points on a face specify the rest. These, however, are just the constraints imposed by the point-plane incidence equations, thus the defect of this linear system (number of unknowns - rank of the coefficient matrix) must be 4. Any 4 independent points ,therefore, are sufficient for specification and the theorem is proved.

Figure 3-6a gives the face adjacency graph G for Figure 3-5a and Figures 3-6b, c, d, e, and f show one possible sequence of merges. From this sequence we conclude that graph G is mergeable. This verifies our previous claim that 4 points are sufficient to specify this object from its projection. Figure 3-8a shows the face adjacency graph for the projection of Figure 3-5b. It is not difficult to convince oneself that in this case G is not mergeable. We shall show later how a non-trivial set of points sufficient for specification can be selected in general.

Roughly, one might say that G will be mergeable if those faces which are visible are not "too occluded" by other visible faces. With a little thought it is clear that if hidden lines are not removed from the projection of a trihedral object (i.e. if the object were actually a wire basket), then 4 points would always be sufficient to specify the object (G would be mergeable). It is the missing links (edges and vertices) between visible faces that cause difficulties (Figure 3-5b). Nevertheless, some edges can be occluded if enough others are present (Figure 3-5a). Based upon additional assumptions of regularity it may be possible to add totally occluded lines to the projection and extend partially hidden ones so that enough of the wire basket is present for 4 points to suffice. An example of the case in point is Figure 3-5c where 5 point locations are required for specification. If the dotted



ADDED EDGE
1
2
3
4
5
6
(a)
1,2
(b)
1,2,3
(c)
SPECIFY AN INDEPENDENT POINT ON FACE 4
1,2,3,4
(d)
1,2,3,4,5
(e)
1,2,3,4,5,6
(f)
5 POINTS REQUIRED FOR SPECIFICATION

Figure 3-8 . Merging the Face Adjacency Graph for Figure 3-5b .

line is added, however, (i.e. regions 1 and 2 are assumed to be part of the same face) then only 4 points are required.

We turn now to the question of determining G(P) from a projection. There is little problem in determining the nodes of G for an arbitrary projection. Similarly, there is no trouble in finding out whether or not two regions (Ri and Rj) in P share a common line L. Since we do not want to assume that we have already determined any mapping between the projection and a stored prototype, however, there is a problem in determining whether L corresponds to part of an actual edge between M(Ri) and M(Rj). While a few simple heuristic tests applied to the projection would undoubtably give reasonably reliable results, we shall consider a more analytic approach below. This approach utilizes some techniques developed by D.A. Huffman at U C Santa Cruz.

Huffman has catalogued the 13 distinct ways in which a corner can appear in a non-degenerate projection of a trihedral body [18]. Figure 3-9 gives this catalogue along with an example of each of the entries. The notation (following that of Huffman) is that a + signifies a "convex edge" and a - signifies a "concave edge". An edge labeled with an arrow implies that the face to the right (when looking toward the arrow) of the edge is visible while that to the left of the arrow is not visible. He uses this catalogue in a labeling procedure. A projection must be able to be properly labeled for it to correspond to a real

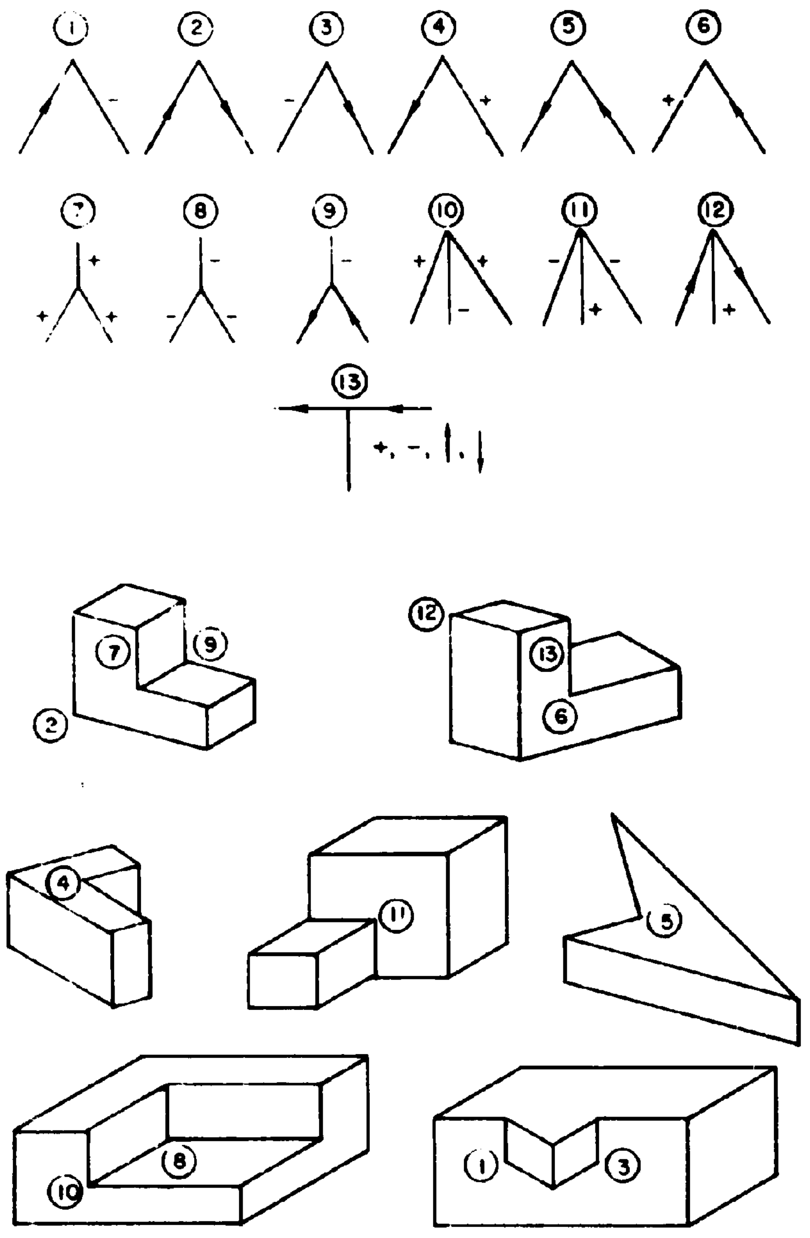


Figure 3-9 . Huffman's Catalogue and Examples .

projection of a trihedral body. We shall also proceed by labeling the projection. The process is initiated by marking all exterior edges of the projection with "arrows" (in a clockwise direction). Un-labeled edges are subsequently marked based on those edges already labeled and the constraints imposed by the catalogue. Figure 3-10 shows the complete labeling for a simple projection. Any line in the projection labeled with an "arrow" would not imply an edge between the nodes corresponding to its left and right regions in G(P) whereas a "+" or a "-" labeling would imply such links.

The notion of mergeability can be extended in a straight-forward manner so as to imply an upper bound on the number of points needed to specify any trihedral object from its projection. Suppose that G is not mergeable, that is, after adding an edge between two adjacent nodes and merging G as far as possible $K \neq R$. We must, therefore, specify at least one more point in 3-space for the object to be determined. Let us pick this point on a face corresponding to a node adjacent to K and represent the specification of this point by the addition of an edge between K and this node. Now the merging can proceed as before. If the merging process again terminates prematurely (with $K \neq R$), we repeat the above procedure. Figure 3-8 illustrates this idea for the projection of Figure 3-5b.

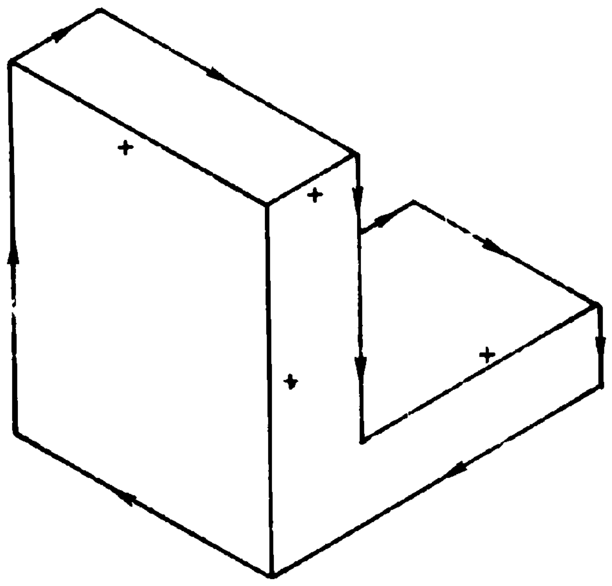


Figure 3-10 . A Labeled Projection .

We can define G(P) to be N-mergeable if N is the minimum number of edges that must be added to G for it to be reducible by a sequence of merges to a single node (nucleus). From the above argument it follows that if G is N-mergeable, then the object can be specified uniquely from (3+N) independent points in addition to complete monocular information. This is consistent with our previous result if we interpret "mergeable" to mean "1-mergeable".

Applications:

The practical application of these ideas for the interpretation of 3-D scenes is fairly obvious. We have shown that for the right type of projection if the actual 3-space location of an appropriate set of object points can be found (in many cases only 4), then the rest of the object points can be located from only monocular information. The initial points may be located by any of the methods described earlier in this chapter. From another point of view, if all or some of the object points have been located using heuristic procedures, then we have a check on the consistency of our assumptions.

BLANK PAGE

CHAPTER 4
VISION SYSTEM ORGANIZATON

In the introduction we indicated that our programs are part of a larger computer vision system. This chapter considers the organization of that system. In addition, the basic structure and operation of the scene description programs is presented. Chapters 5 and 6 describe our scene description techniques in more detail.

SINGLE vs. MULTI OBJECT SCENES

There are two distinct scene description programs, SIMPLE and COMPLEX. SIMPLE has the job of describing portions of the scene in which there is only a single body. COMPLEX has the more difficult task of describing portions of the scene in which there are several mutually-occluding bodies. This division is motivated by our desire to treat simple scenes simply. For the set of solids shown in Figure 1-1, a single object can almost always be recognized from its "outline" (see Figure 4-1).

The edge-follower program begins by determining the outline enclosing a blob in the field of view. A local gradient operator scans upward (see Figure 4-2) over the digitized intensity matrix looking for a significant intensity discontinuity. When such a discontinuity is

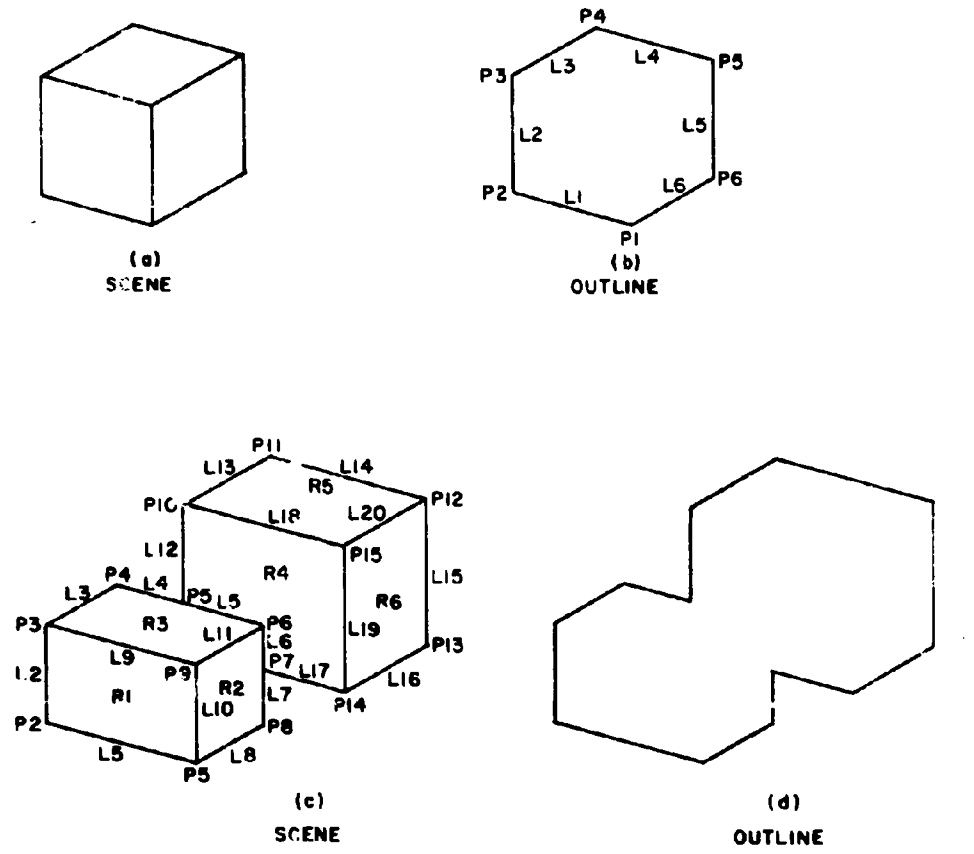


Figure 4-1 . Scenes and Their Outlines .

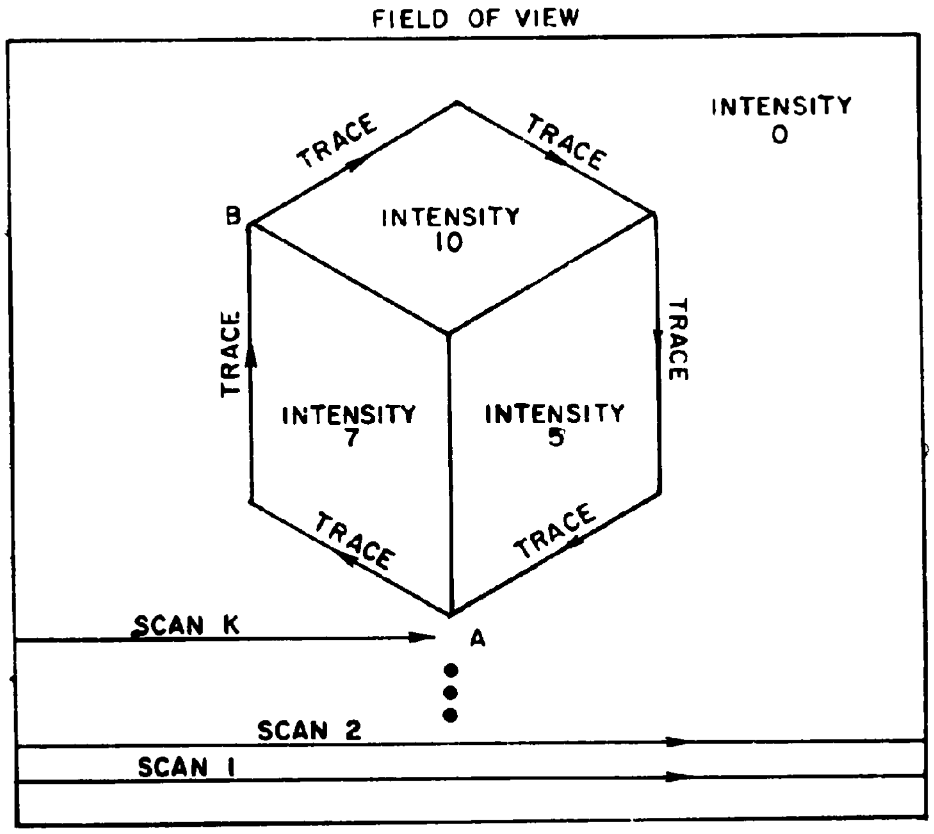
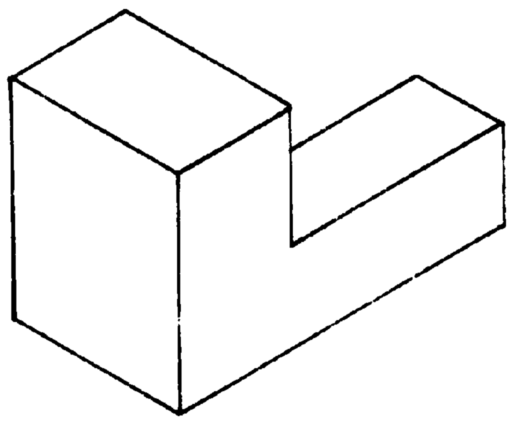


Figure 4-2 . The Edge-follower Finding a Block .

found (point A), the scanning is interrupted and the program enters trace mode. It tries to follow around the exterior edges of the blob keeping the background intensity on one side of the edge being traced. This last requirement keeps the edge-follower from taking the wrong path at a corner like B. The complete outline has presumably been traced when the initial point traced is detected. Lines are subsequently fitted to the edge points to produce the outline.

The edge-follower infers the nature of the blob from the complexity of its outline. Currently, the complexity of an outline is determined by the number of lines it contains. If the outline is simply a triangle, the blob is assumed to be noise and forgotten. If the number of lines, N, exceeds NMAX, the edge-follower assumes that the blob corresponds to a group of objects, otherwise it assumes that the blob is a single object. NMAX is chosen so that instances of our most complex prototype, the LBEAM, will be classified properly (NMAX=9). If N≤NMAX, the outline is passed to SIMPLE for analysis. If N>NMAX, additional preprocessing is required. Specifically, interior edge points must be detected, lines must be fit to these points, these lines must be linked to the outline, regions must be determined, and the background must be identified. The resulting line drawing is passed to COMPLEX for analysis. Grape [8] has been investigating the problem of producing accurate line drawings based on


(a)

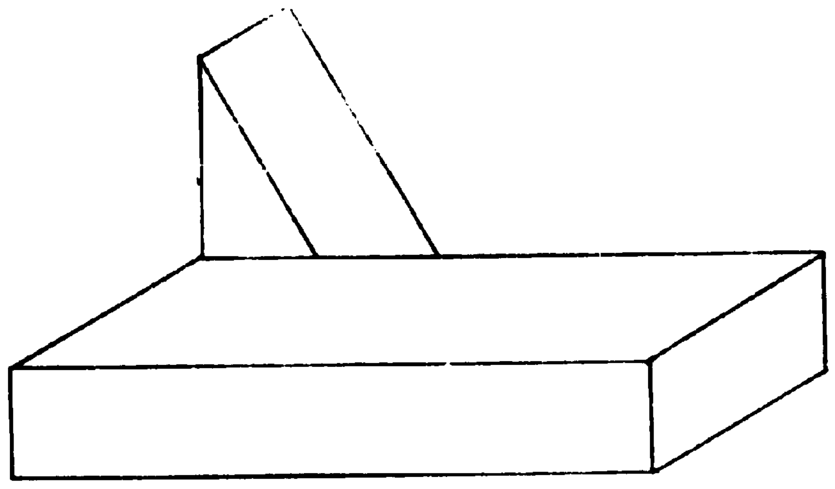

(b)

Figure 4-3 . Two Scenes with Nine Line Outlines .

noisy edge data for some time.

The blob classification scheme that we have described can, of course, fail. Both scenes in Figure 4-3 would result in blob outlines with 9 lines. This does not cause any problem, however, since the outline of Figure 4-3b will not be recognized by SIMPLE. Such a failure signals the preprocessors to examine the interior of the outline and subsequently pass the resulting line drawing to COMPLEX for analysis.

INPUT FORMATS

Outline Representation:

Outlines are represented internally using the associative structure of the SAIL language [39]. Although a complete description of this language can not be presented here, a basic familiarity with SAIL's associative mechanism is necessary for an understanding of the representation used for outlines, line drawings, and prototypes.

The basic associative element in SAIL is the "item". Associations or triples of the form A ⊗ O ≡ V (read "A of O is V") exist where A, O, and V are items. Items may be typed, that is, they may have a DATUM which is a real number, an integer, an array, a string, or a set. Sets and the usual set operations of union, intersection, and subtraction also exist in SAIL. The ability to selectively search the store of associations exists, however, the details of this ability are not crucial to the representation issue at hand.

The internal description of the outline of Figure 4-1b is given as:

```
LINE • SCENE ≡ L1
LINE • SCENE ≡ L2
•
•
LINE • SCENE ≡ L6
POINT • SCENE ≡ P1
•
•
POINT • SCENE ≡ P6
ENDPOINT • L1 ≡ P1
ENDPOINT • L1 ≡ P2
•
•
ENDPOINT • L6 ≡ P6
ENDPOINT • L6 ≡ P1
```

The items SCENE, LINE, POINT, ENDPOINT, and LJ (VJ) are untyped items, that is, they do not have any associated datums. The PJ's are typed as real array items where the array for each point contains (XJ,YJ), the image coordinates of the point.

Complete Line Drawings:

The representation of complete line drawings is slightly more involved than the one for outlines. It is best presented in terms of an example. We shall use the shorthand notation A • O ≡ (V1, V2,... Vn) for the actual set of triples A • O ≡V1, A•O ≡V2, ... A •O ≡Vn. The associations representing projection of Figure 4-1d are:

```
BACKGROUND • SCENE ≡ BACK
POINT • SCENE ≡ (P1, P2, ...P15)
LINE • SCENE ≡ (L1, L2, ...L20)
REGION • SCENE ≡ (R1,R2, ... R6, BACK)
ENDPOINT • L1 ≡ (P1, P2)
.
.
ENDPOINT • L20 ≡ (P19, P15)
BOUNDARY • R1 ≡ (L1, L2, L9, L10)
.
.
BOUNDARY • BACK ≡ (L1,L2,L3,L4,L12,L13,L14,L15,
                   L16,L17,L7,L8)
CORNER • R1 ≡ (P1,P2,P3,P9)
```

.

.

CORNER • BACK ≡ (P1,P2,P3,P4,P5,P10,P11,P12,P13,
P14,P7,P8)

Each of the points, lines, and regions of the scene are represented by items. All items except the Pj's are untyped. The real array attached to each Pj again locates the corresponding point in image coordinates. For each region of the line drawing there are a set of associations which specify the boundaries of the region. A set of associations also specify the corners of each region. For each line two triples describe its end points. No dangling lines, i.e. those which border only a single region, are present in the line drawing. The region-finder eliminates them during preprocessing. The redundancy in the BOUNDARY, CORNER, and ENDPOINT associations (clearly the corners of a region = union of the end points of all the boundaries) results in programs that are easier to read and more efficient at run time.

Prototype Representation:

As previously mentioned, a separate prototype is kept for each of the objects in Figure 1-1. The prototype contains complete structural information (topology and

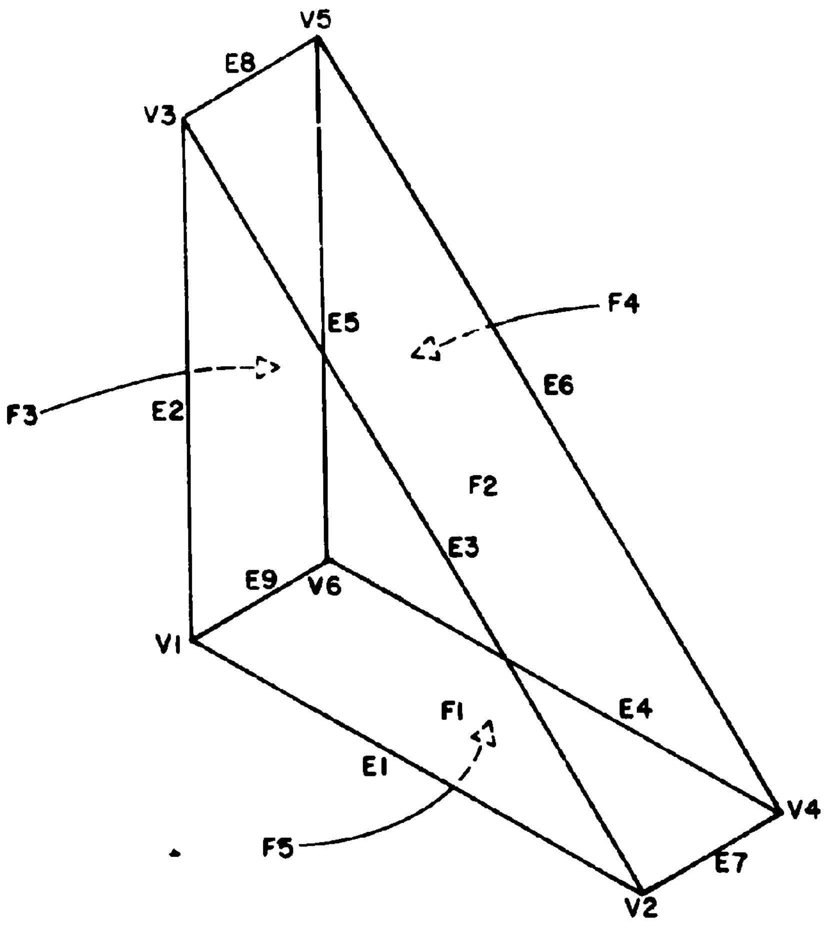


Figure 4-4 . Prototype WEDGE122 .

geometry) about the corresponding solid. The representation of prototypes differs only slightly from the representation described for line drawings. For each model the item SCENE above is replaced by the item representing the prototype (e.g. CUBE, WEDGE122, etc.). Similarly, the items POINT, LINE, and REGION are replaced by VERTEX, EDGE, and FACE respectively. Obviously, none of the associations mentioning the background, BACK, exist for the 3-D models. For the WEDGE122 shown in Figure 4-4, the set of associations are:

VERTEX • WEDGE122 ≡ (V1, V2, ...V6)

EDGE • WEDGE122 ≡ (E1, E2, ...E9)

FACE • WEDGE ≡ (F1, F2, ...F5)

ENDPOINT • E1 ≡ (V1, V2)

.

.

ENDPOINT • E9 ≡ (V1,V6)

BOUNDARY • F1 ≡ (E1, E2, E3)

.

.

BOUNDARY • F5 ≡ (E1, E7, E4, E9)

CORNER • F1 ≡ (V1, V2, V3)

.

.

CORNER • F5 ≡ (V1, V2, V4, V6)

The array attached to each vertex now contains the homogeneous coordinates of the vertex relative to the center of mass of the object. Edges and faces also have associated datums in the 3-D case. For an edge the datum is a real number, the length of the edge. In the case of a face, the datum is an array which specifies the unit normal to the face. A more thorough description of this 3-D world model is given in [30].

SINGLE BODY RECOGNIZER-"SIMPLE"

Historically, SIMPLE was the first scene description program that we wrote. Our goal at that time was to see how little preprocessing we had to do in order to analyze single object scenes. As indicated earlier, our current motivation for having a separate single body recognizer is efficiency. Except for the fact that SIMPLE must infer its features from an outline rather than a complete line drawing, the techniques that it uses are conceptually the same as those used by COMPLEX. For this reason, we do not describe it in any detail in the remainder of this thesis.

COMPLEX SCENE ANALYZER-"COMPLEX"

We have summarized in the flowchart of Figure 4-5 the overall operation of COMPLEX. COMPLEX consists of 3 basic parts, SEGMENT, RECOGNIZE, and HIDDEN. If a line drawing consists of more than a single object, COMPLEX segments the line drawing into parts corresponding to individual bodies before attempting recognition. The procedure, SEGMENT, assumes only that the line drawing is a projection of a planar-faced solid, that is, it is independent of the set of prototypes in Figure 1-1. The partial body projections are then matched against the stored prototypes by the recognizer, RECOGNIZE. This scheme, first advocated by Guzman, has the desirable property that recognition time increases only linearly with the complexity of the scene (number of bodies present). If recognition is attempted without previously segmenting the scene into bodies [33], one has no idea which lines or regions to compare with a prototype. Chapter 5 describes SEGMENT in detail. The general idea of segmentation is a basic one and is also used in the computer analysis of connected speech [31,44].

RECOGNIZE is designed to make decisions based on either complete or incomplete line drawings of individual objects. Nevertheless, there are situations when an incomplete line drawing does not provide sufficient data for

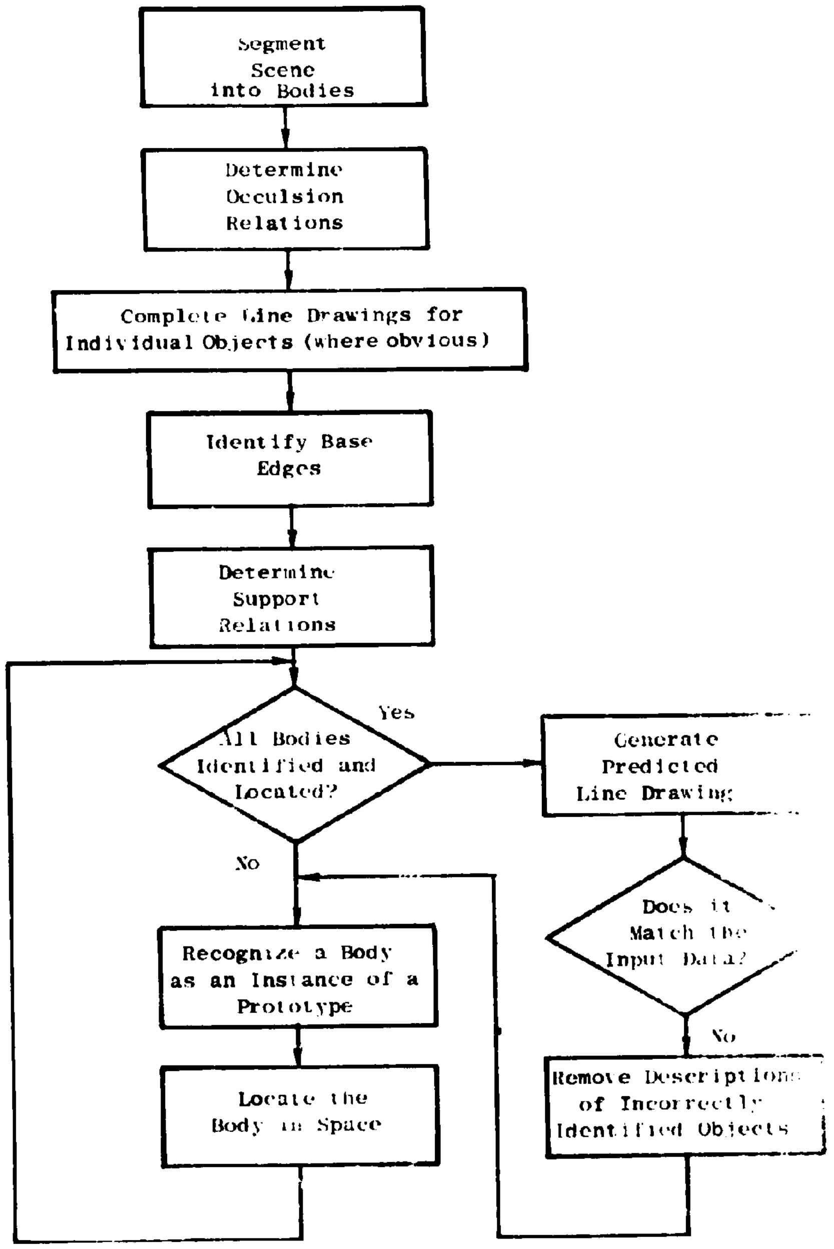

Segment Scene into Bodies
Determine Occulsion Relations
Complete Line Drawings for Individual Objects (where obvious)
Identify Base Edges
Determine Support Relations
All Bodies Identified and Located?
Yes
No
Generate Predicted Line Drawing
Does it Match the Input Data?
No
Recognize a Body as an Instance of a Prototype
Locate the Body in Space
Remove Descriptions of Incorrectly Identified Objects

Figure 4-5 . The Overall Operation of COMPLEX .

recognition. RECOGNIZE attempts to complete occluded line drawings in those cases where the missing lines are obvious. The example in Chapter 1 illustrates one such case. The remaining situations where completion is "obvious" are described in Chapter 6. Completion, as segmentation, is accomplished without reference to the stored prototypes.

The assumption that a lone object is supported by the table is certainly a reasonable one. In scenes consisting of many bodies, however, blocks may be supported by other blocks in a variety of ways which are not easily distinguished. For example, in Figure 4-6 it is not easy to tell from 2-D information whether Body1 rests flat on Body2 or whether it leans on Body2. If we have no prior knowledge about how the blocks are arranged, it would appear that the best strategy for locating image points in 3-space is to apply stereo or focus ranging for all points not clearly on the table.

In the context of the Hand-Eye system, however, we do have some "weak" information about the scenes. The type of structure which we plan to build or manipulate will typically consist of blocks either resting on the table or resting on the top faces of other blocks. If leaning blocks occur, this will usually indicate an error (e.g. a block may have fallen out of the hand or a construction may have tumbled). COMPLEX tries to recognize the scene assuming that each object is either supported by the table or on the

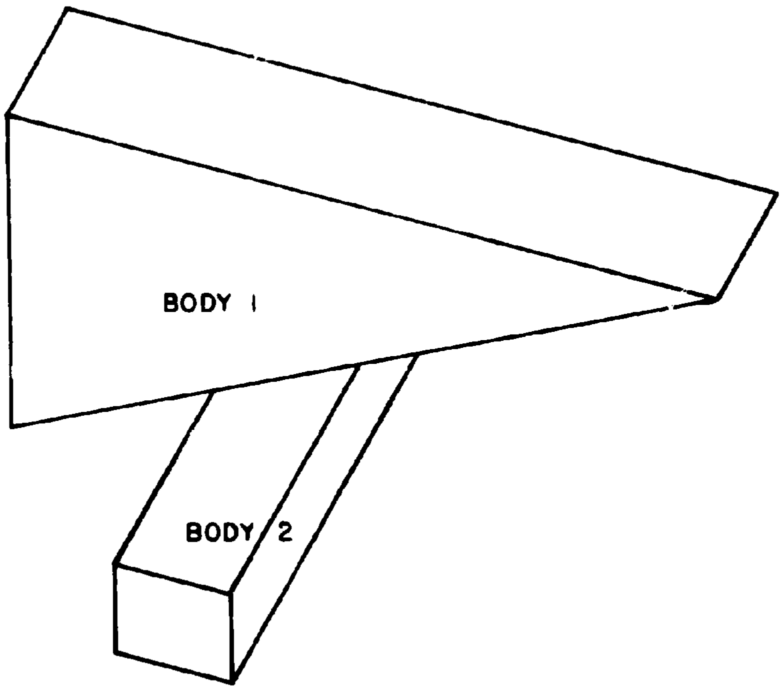


Figure 4-6 . Is BODY1 Leaning on BODY2 or Resting Flat on Top of It?

horizontal top of another object. If this analysis is successful, it concludes that its assumptions were valid. If somewhere in the analysis an object cannot be properly recognized, COMPLEX considers the possibility that the support for the body was improperly determined (i.e. it probably was leaning). COMPLEX then calls on stereo or focus ranging to locate the corners of the object for subsequent re-recognition. The support relations can usually be determined quite easily if the base edges of each object have already been identified. Techniques for doing this are described in the next chapter.

After the support relations have been determined, the partial projections can be analyzed in a straightforward manner. First, those blocks supported by the table are recognized and located in space. Then any object, Bi, supported by an object on the table, Bj, is recognized by assuming that Bi is supported in plane z=h where h is the height of the top (maximum z) of Bj. COMPLEX proceeds in this manner, applying RECOGNIZE to all the potential supporters of a given body and then to the body itself.

Since the top objects are generally the least occluded, it would be better to recognize the top-most object first and proceed downward. The problem with this approach is that COMPLEX has no geometric information about the top object unless either stereo or focus ranging (both costly) is applied. Consequently, COMPLEX orders the individual recognitions as described above.

After all of the bodies have been tentatively identified, COMPLEX activates the predictor-hidden line eliminator, HIDDEN. The result is a line drawing that indicates how the hypothesized scene description would appear from the point of view of the camera. If this predicted line drawing matches the original input (line drawing and TV intensity image) to within some prespecified tolerance, then COMPLEX accepts its previously tentative analysis. If the two do not match, it tries to detect and correct the identity and/or location of those bodies that have been mismatched.

From the scene description which COMPLEX generates, the "hand" and associated programs can determine the appropriate motions necessary to grasp and move any body within reach. The scene description is also of interest to the strategy program which determines how to carry out a given task.

THE HAND-EYE SYSTEM

The block diagram of Figure 4-7 shows all of the Hand-Eye system modules. For clarity, many of the lines indicating module interactions have been omitted. Each of the boxes represents a separate "job" as understood by our modified PDP-10 time-sharing system [25]. "Message

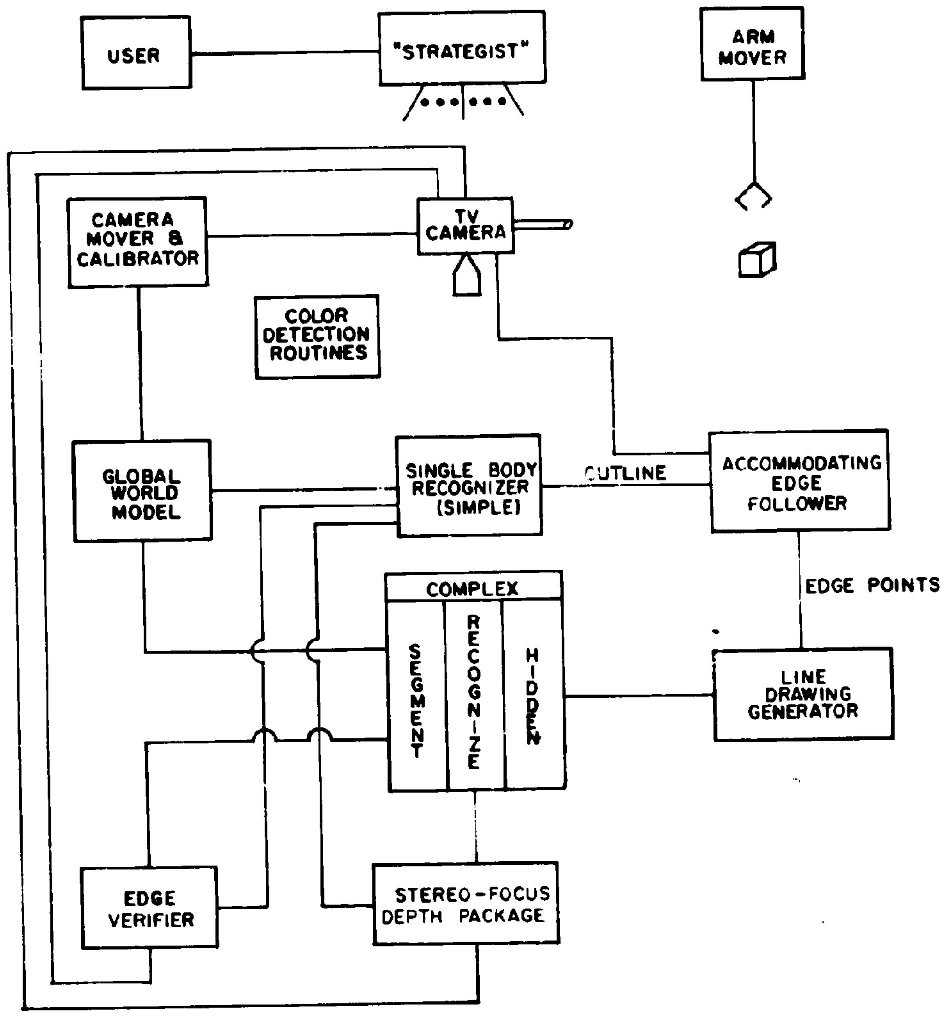


Figure 4-7 . The Hand-Eye System .

procedures" provide a means of inter-module communication while the "global world model" serves to record data of interest to several system components. A description of these features can be found in [38]. The size of the entire system is on the order of 300K words. At the time of this writing all of the modules are complete or nearing completion and an initial system configuration exists.

The user specifies a task to the STRATEGIST module. This program determines how to activate the modules in order to carry out the task. The STRATEGIST, therefore, has the ability to talk to and receive requests from all of the other blocks in Figure 4-7. Suppose, for example, the task were to simply analyze a particular area of the field of view. If the camera were not already calibrated, the STRATEGIST would activate the CAMERA CALIBRATOR to read the potentiometers indicating the camera position and store the appropriate camera model in the GLOBAL WORLD MODEL. The ACCOMMODATING EDGE-FOLLOWER would then scan the specified area and determine what sort of blob it thinks is present. If only a single object were present, the outline of the object would be passed to SIMPLE for analysis. If the scene were determined to consist of more than one object, the interior of the outline would be processed by the ACCOMMODATING EDGE-FOLLOWER and all the edge-point data passed on to the LINE DRAWING GENERATOR. This program would eventually produce a line drawing and pass it on to COMPLEX for analysis.

The result of a successful scene description by either SIMPLE or COMPLEX is that the identity and location of each object present in the scene is stored in the associative structure of the GLOBAL WORLD MODEL. If, for example, the scene is determined to contain two objects, say a WEDGE122 and an LBEAM, the following associations would be added:

INSTANCE • WEDGE122 ≡ OBJECT1

INSTANCE • LBEAM ≡ OBJECT2 .

The datums of the items OBJECT1 and OBJECT2 are 4x4 arrays that specify the position of the corresponding object in space. The EDGE VERIFIER can be activated by either of the scene description programs to check that a predicted edge actually exists in the TV image. Similarly, the stereo-focus DEPTH FINDER can be called to verify a corner location or at any time if monocular cues become insufficient.

The precise interaction of the modules depends on the state of the environment and the task which has been specified. For example, it is possible that the LINE DRAWING GENERATOR will need to call upon the EDGE VERIFIER. The EDGE VERIFIER, in turn, might require the CAMERA MOVER to change to a longer lens in order to see a particular area more clearly. Considerably more experimentation needs to be done before we have a clear understanding of how well the modules perform as a system.

CHAPTER 5

INTERPRETATION OF SCENE STRUCTURE

In this chapter we describe programs that interpret the structure in scenes. By this we mean that the programs determine certain binary relations that exist between pairs of bodies. For example, the scene structure may simply be "Body1 supports Body2". Structural relations between bodies can be determined independent of the prototypes to which the bodies correspond. As indicated in the previous chapter, RECOGNIZE uses the support relations to identify and locate the objects in the scene. We begin by considering the segmentation of a line drawing into bodies.

A SEGMENTATION PROCEDURE

Problems with a Region Based Approach:

We have sketched in Chapter 2 the technique developed by Guzman for segmenting a scene into bodies. If our preprocessors were able to produce ideal line drawings, we could simply incorporate SEE (Guzman's program) as part of COMPLEX. 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 shadows.

The problems of missing and extra lines can to some extent be traded for one another. Parameters in the preprocessor can usually be set so that all the actual edges appear in the line drawing. Such settings, however, often cause extraneous lines to appear as well. On the other hand, if one set the preprocessor parameters so that all noise lines are rejected, some of the true edge lines are also rejected. In practice, one must choose between analyzing an incomplete line drawing or a line drawing containing spurious lines. The former appears to be the more tractable.

Given an incomplete line drawing, a reasonable approach is first to 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 a program has been considered both by the MIT vision group [2] and by Grape [9] here at Stanford. If the resulting line drawing were guaranteed to be complete, a segmentation procedure like Guzman's could then be applied to it. We have found, however, that no such guarantee can be made. The segmentation algorithm described below will work both on complete and incomplete line drawings.

To motivate our approach consider applying SEE to the line drawing of Figure 5-1. Such a line drawing is common when the preprocessor parameters are set to eliminate noise lines. Because R3 corresponds to faces of 2 bodies,

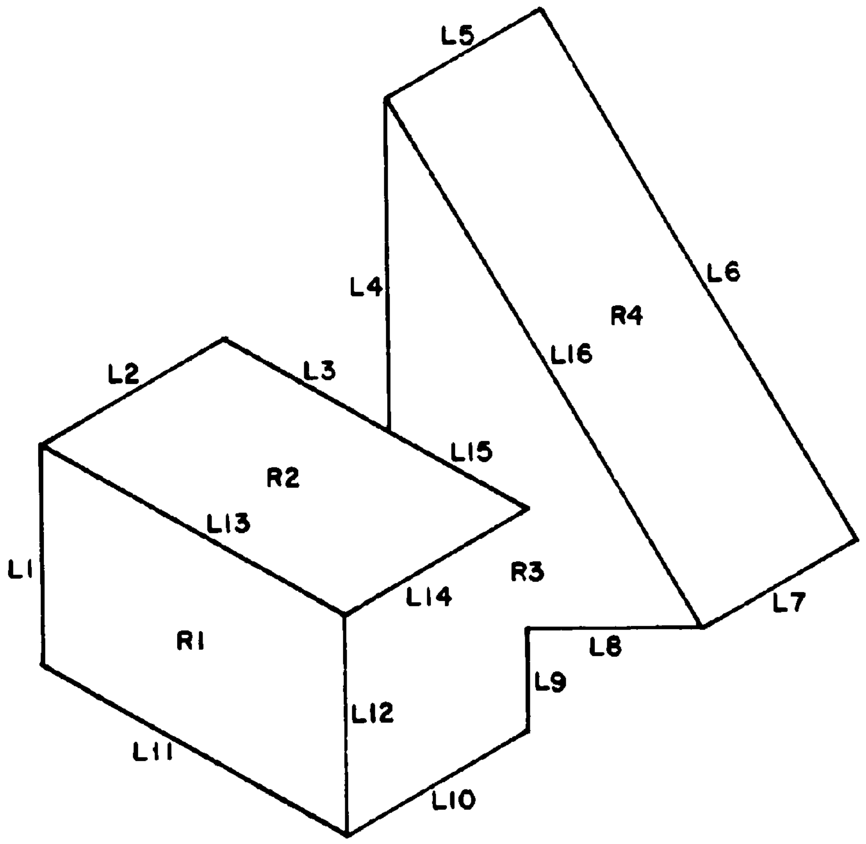


Figure 5-1 . A Scene with a Missing Line .

there is absolutely no correct way for SEE to partition the set of regions. In this case, it incorrectly reports the scene to be a single body since all regions get merged into a single node. It is clear that any scheme which attempts to segment the scene by regions will have this difficulty.

The segmentation procedure, SEGMENT, employed by COMPLEX first partitions the scene into bodies by line. This is accomplished, as in SEE, by combining bits of local evidence accumulated at the vertices. The line partition is subsequently used by a region assignment procedure to detect and split regions such as R3. The resulting set of regions is then partitioned by body. In comparison with Guzman's scheme, this line based approach appears considerably less sensitive to missing lines.

The lines most often missed are the interior lines, that is, those lines which do not border the background. This is because the contrast between an object and the background is usually greater than the contrast between adjacent object faces. For our scenes consisting of white blocks on a black cloth this is particularly true. Although the procedure which we shall describe shortly is intended primarily to handle such cases, scenes with missing exterior edges will often be segmented correctly as well. SEGMENT is not currently equipped to handle scenes containing shadow lines. In the examples of Chapter 7 that were processed without touch up, shadow edges were avoided by the use of diffuse lighting. A later section of this chapter describes several more realistic proposals for dealing with spurious lines in the input.

Line Segmentation:

We shall now describe our approach in detail. The procedure SEGMENT consists of a basic mechanism that partitions the majority of the lines by body. Various "special case techniques" are then applied to attempt to assign the remaining lines. Although a few of the original lines may not get classified by SEGMENT, the recognizer for individual bodies, RECOGNIZE, is prepared to work from partial data.

SEGMENT begins by classifying the vertices in the scene into a number of categories similar to those described by Guzman. The vertex types are shown in Figure 5-2 . The motivation for this classification will become clear as the rest of the segmentation algorithm is described. Very roughly, those vertices labeled GOOD<something> or just <something> indicate that some of the edges incident at that type of vertex should be assumed to belong to the same body. Those vertices labeled BAD<something> indicate that the edges incident at that type of vertex should not be assumed to belong to a single body. We describe the vertex types as follows:

Arrow Vertices: Three lines meeting at a point with one of the angles greater than 180 degrees is classified a GOODARROW (see Figure 5-2a) unless either (1) one of the regions of less than 180 degrees is the background (Figure 5-2b) or (2) the middle line or "shaft" of the arrow is the top of a KJOINT (Figure 5-2c) (see K-vertices below). In these two cases the vertex is labeled BADARROW.

"Y" (Fork) Vertices: Three lines meeting at a point with all of the angles less than 180 degrees is classified a GOODY if at least one of the lines is also the shaft of a GOODARROW vertex (Figure 5-2d). Otherwise, the vertex is classified BADY (Figure 5-2e).

"L" Vertices: A vertex where two lines meet is labeled GOODL if the angle greater than 180 degrees is the background (Figure 5-2f) or both of the incident lines are the "tops" of GOODTs (see T Vertices below) (Figure 5-2g). Otherwise, (Figure 5-2h) the vertex is classified as a BADL.

"T" Vertices: A vertex where three lines meet, two of which are collinear is a T-joint. The collinear lines will be referred to as T-tops and the non-collinear line is

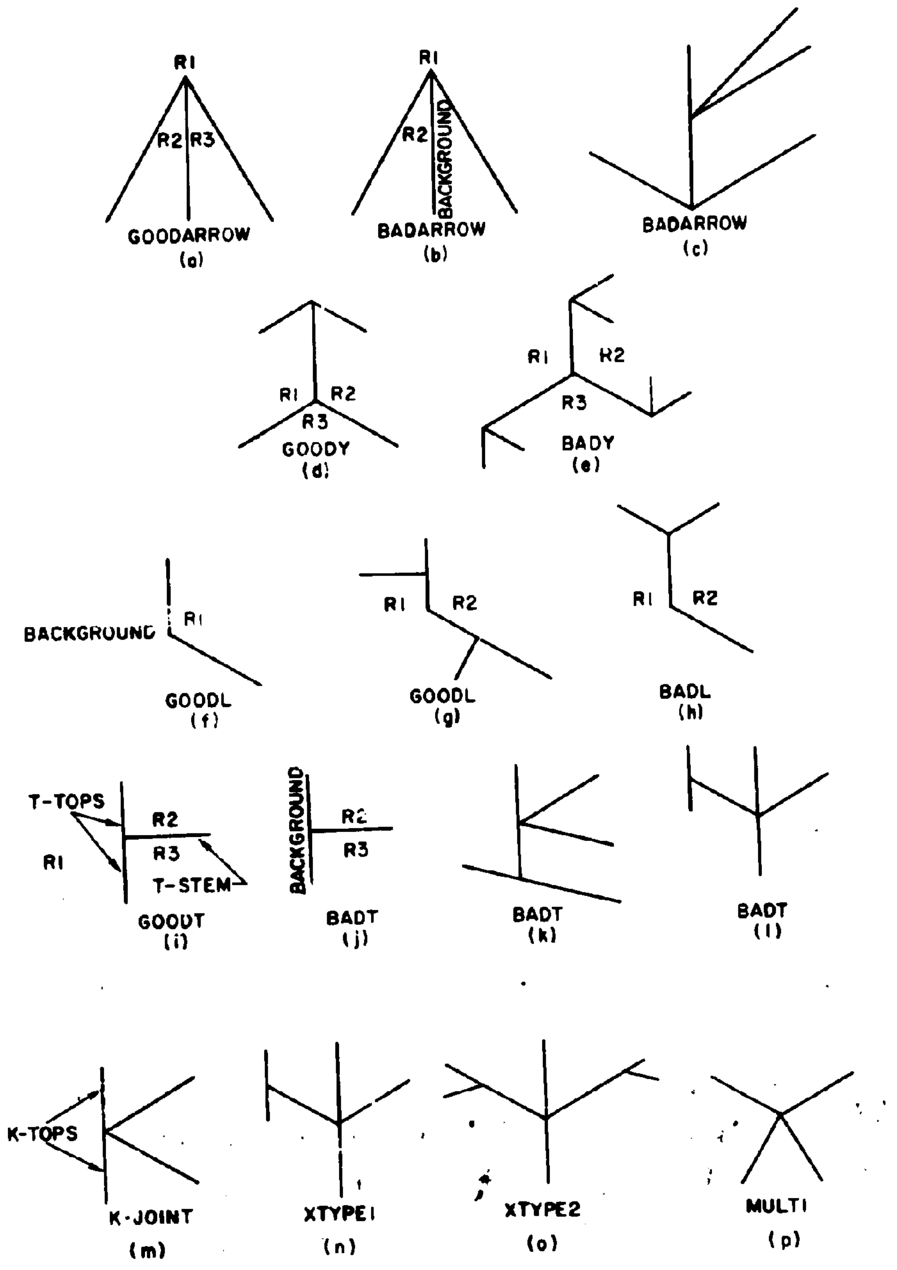


Figure 5-2 . The Vertex Types .

called the T-stem. The vertex is labeled GOODT (Figure 5-2l) unless either (1) the region labeled R1 is the background (Figure 5-2j) or (2) the T-stem is the top of a KJOINT (Figure 5-2k) or (3) the T-stem is one of the non-collinear lines of an Xjoint (see X-joints below) (Figure 5-2l). In these three cases vertex is labeled BADT.

"K" Vertices: A vertex where 4 lines meet is labeled a KJOINT if two of the lines are collinear and the other two are on the same side of the collinear pair (Figure 5-2m). The collinear pair are referred to as the K-tops.

"X" Vertices: A vertex where 4 lines meet is called an X-joint if two of the lines are collinear and the other two are on opposite sides of the collinear pair (Figures 5-2n and o). If either of the non-collinear lines is the stem of a T-joint, then the vertex is labeled XTYPE1, otherwise it is labeled XTYPE2.

MULTI Vertices: A vertex where 4 or more lines meet that is labeled "X" or "K" is labeled MULTI (Figure 5-2p). Some line drawing projections with labeled vertices are shown in Figure 5-3.

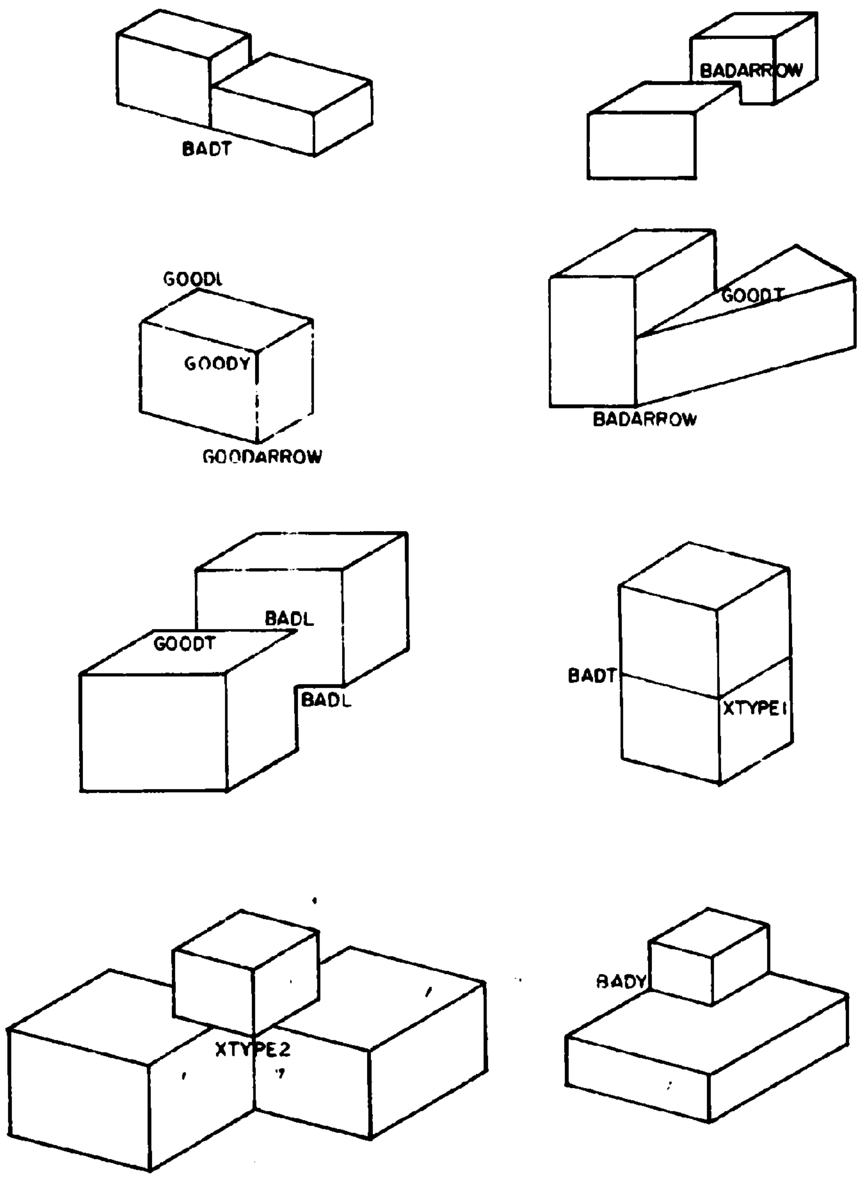


Figure 5-3 . Examples of the Vertex Types .

After classifying the vertices of the line drawing, SEGMENT proceeds to set up an undirected graph based upon this classification. Some vertices, BADARROWs, BADYs, BADLs, BADTs, and MULTIs, generate no nodes in the graph. Most vertices, GOODARROWs,GOODYs, GOODTs, GOODLs, and XTYPE2's, generate a single node in the graph. Two of the vertex types, KJOINT's and XTYPE1s, generate two nodes in the graph.

SEGMENT associates a set of lines with each of these nodes. The table below indicates tho contents of each node set as a function of the corresponding vertex type.

VERTEX TYPE	WHICH OF THE INCIDENT LINES?
....	....
GOODARROW	all 3 lines
GOODY	all 3 lines
GOODL	both lines
GOODT	both collinear lines
XTYPE2	the 3 lines which form an "arrow"
KJOINT	Node 1: the 2 collinear lines
	Node 2: the other 2 lines
XTYPE1	Node1: the 3 lines which form an "arrow"
	Node 2: the collinear line not in the set of Node 1 plus 2 "new lines" (see below)

At an XTYPE1 vertex SEGMENT interprets each of the non-collinear edges as a double edge resulting from alignment of two objects. The two "new lines" added to Node 2 in this case represent the 2 missing edges.

Edges or links between the nodes are based upon the sets associated with the nodes. If Ni and Nj are two nodes and their associated sets are Si and Sj respectively, then an edge exists between Ni and Nj if and only if Si ∩ Sj≠PHI. The set associated with a node implies that the lines contained in the set should belong to a single body. If a line is contained in two node sets, the lines in both of the sets should belong to a single body. This fact is recorded by the link between the nodes. The graph set up for the projection of Figure 5-1 is shown in Figure 5-4.

Several special situations that demand attention are shown in Figure 5-5 . Figure 5-5a illustrates a pair of "matched BADLs". Two BADLs are called matched if they are corners of the same region and a line of one of them is collinear with one of the lines of the other. SEGMENT generates an additional node of the graph for this case and associate with it the set of collinear lines. This node is linked to the rest of the graph as were the nodes resulting from the actual vertices in the scene. Although the scene of Figure 5-5 would be correctly partitioned without this additional node, the matched BADL heuristic can be quite useful (see Chapter 7).

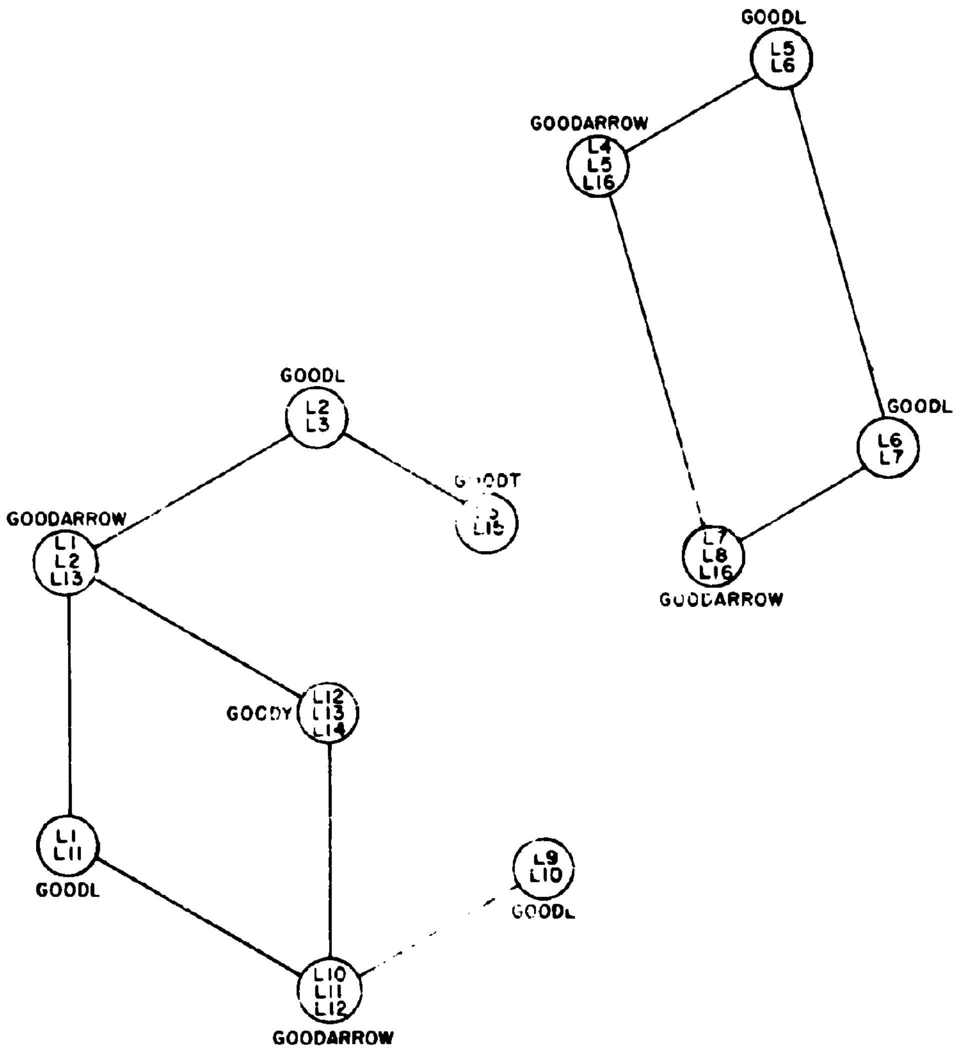
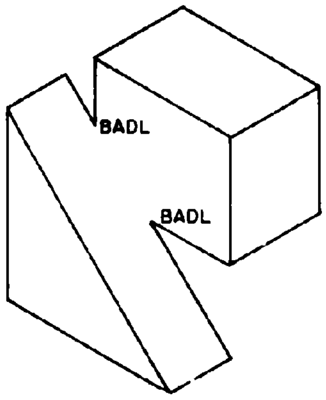
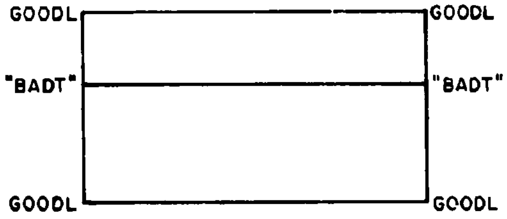


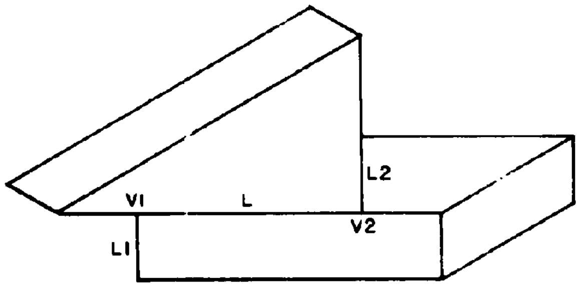
Figure 5-4 . The Graph Set Up for the Projection of Figure 5-1 .



(a)



(b)



(c)

Figure 5-5 . Some Special Situations Handled by SEGMENT .

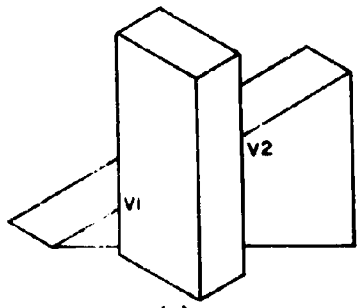
BADT vertices usually result from one body abutting against another. As indicated in Figure 5-5b, however, BADTs can also result from a degenerate view of a single object. Since distinguishing between these two cases appears to be important, SEGMENT examines each BADT after the initial labeling is completed. If, as in Figure 5-5b, both T-tops are also incident at L-vertices, the BADT is changed to a GOODARROW.

The last special situation handled at this point is illustrated in Figure 5-5c. SEGMENT attempts to remove spurious links resulting from adjacent T-joints. If a pair of adjacent T-joints sharing a common T-top are detected (V1 and V2), and the corresponding T-stems (L1 and L2) do not bound a common region, SEGMENT deletes the common line (L) from all the node sets and updates the links between the nodes. If this were not done, only a single body would be reported for Figure 5-5c.

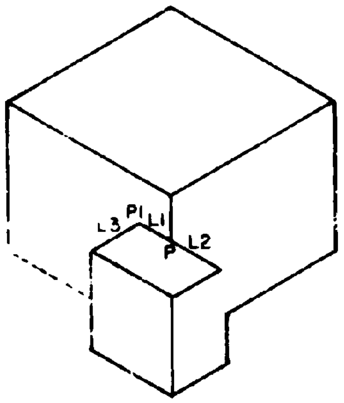
SEGMENT applies a merging procedure to the graph set up above to produce an initial description of which lines belong to which bodies. This process is similar to the merging process described in Chapter 3. In this case two nodes of the graph will be merged if there are one or more links connecting them. When two nodes are merged into a new node, there is associated with the resulting node a set of lines. This set is the union of the sets associated with the 2 merged nodes.

Ideally, when the merging terminates, each remaining node corresponds to a separate body in the scene, and every line in the scene is associated with a node set. In practice, however, this is not the case. Consequently, SEGMENT calls upon various special case heuristics at this point to merge unmerged nodes and assign unassigned lines. Figure 5-6a illustrates a situation where the procedure as described above is clearly inadequate. The partially occluded wedge will be reported as two bodies. Guzman handled this problem by initially adding extra links between non-adjacent regions based on "matching T-joints". SEGMENT does a similar thing but after the merging has been completed. It looks for evidence that two final nodes should be coalesced. In Figure 5-6a the matching T-joints at V1 and V2 imply that the two nodes corresponding to pieces of the wedge should be combined.

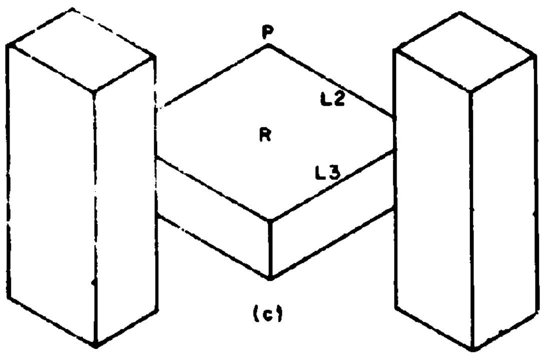
Final nodes having associated sets of length 2 are examined at this point. SEGMENT assumes that such nodes do not correspond to separate bodies. If it cannot find evidence for merging them with other nodes, they are simply deleted. The two cases where a merge occurs are illustrated in Figures 5-6b and c. In Figure 5-6b, the BADL at P1 is taken as evidence that the node at P (whose associated set contains L1 and L2) should be merged with the node whose associated set contains L3. In Figure 5-6 c, the T-stems at L2 and L3 sharing a common T-top and bordering a common



(a)

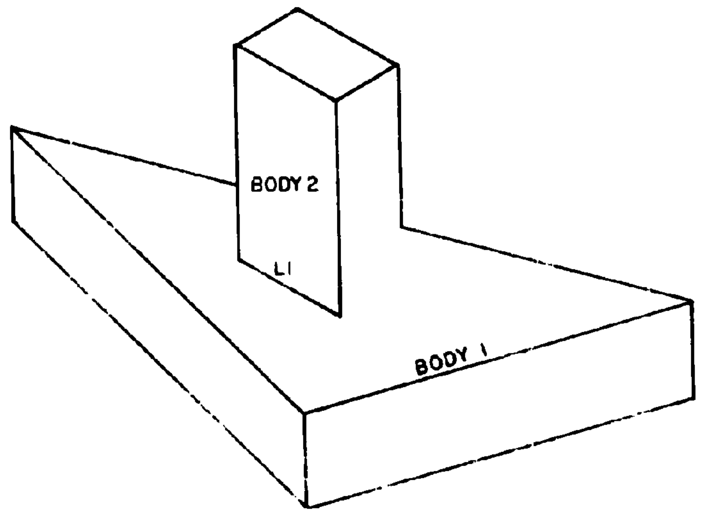


(b)

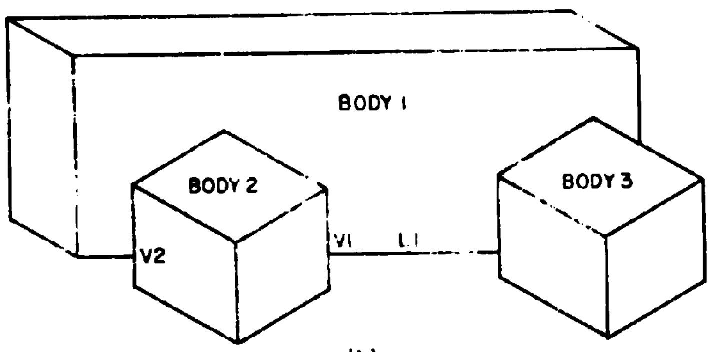


(c)

Figure 5-6 . Cases in which Additional Merging Occurs .



(a)



(b)

Figure 5-7 . Cases in which Additional Lines get Assigned to Bodies.

region, R, are taken as evidence that the node associated with the GOODL vertex, P, should be merged with the node whose associated set contains L3.

It is not necessarily the case that every line in the projection belongs to a set associated with some node of the graph when the merging has been completed. In Figure 5-7a , for example, L1 is not a member of any node set because it is incident at 2 BADL vertices. SEGMENT proceeds to assign an unassigned line to a particular body, B, if one of the regions which the line bounds has all of its classified lines belonging to B. In this case L1 gets assigned to Body 2. Figure 5-7b illustrates another omission which SEGMENT can correct. The omitted line, L1, is put in the node set corresponding to Body1 based or the matching T-joints at V1 and V2.

Region Assignment:

Simply stated the region assignment problem is: given a line drawing projection and a classification of each line as to which body it belongs, classify the regions as to which body they belong. (For the case of Guzman's region-based segmentation the line assignment problem is: given the regions classified by body, classify the lines by body.) This information is needed to set up the complete description of each individual body.

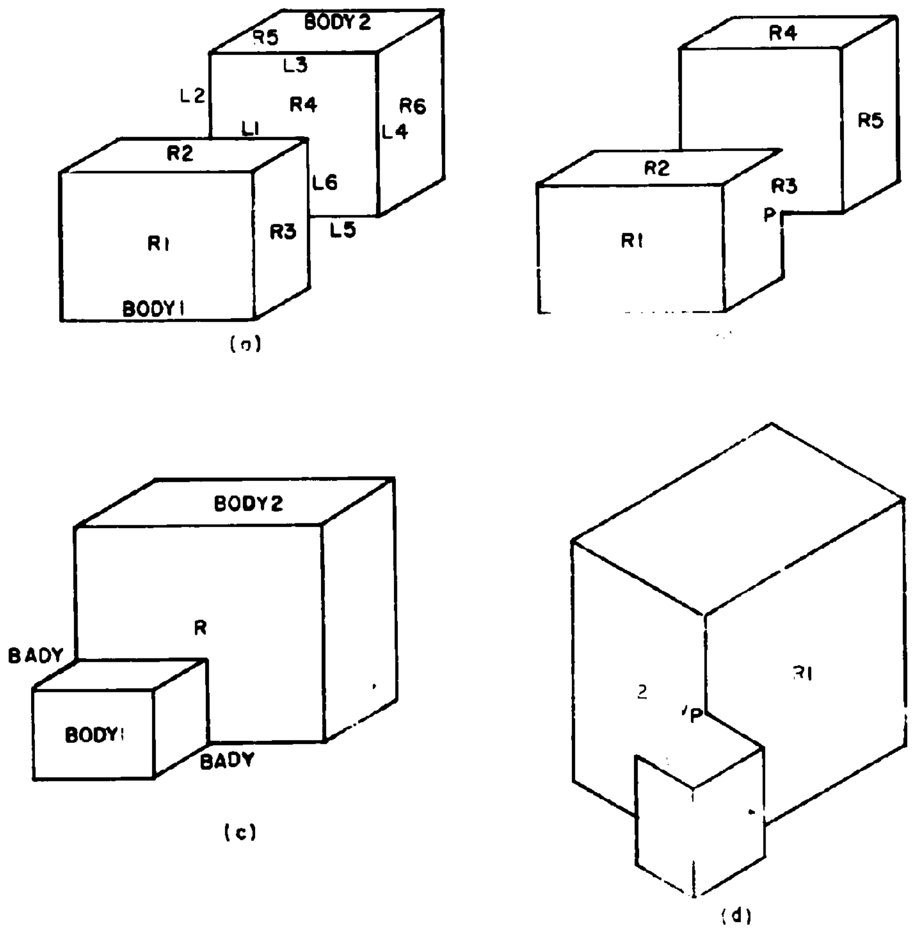


Figure 5-8 . Examples of the Region Assignment Problem .

Consider the line drawing of Figure 5-8a. Assuming that the lines have been assigned to the appropriate bodies, it is straightforward to identify the bodies to which some regions belong. Regions R1,R2, and R3 belong to one body and regions R5 and R6 belong to the other body since all the bounding lines for each of these regions belong to a single body. Region R4, however, is more difficult to classify since its boundaries belong to different bodies. L1 and L6 belong to one body and L2, L3, L4, and L5 belong to the other body. The problem is exacerbated by missing lines as in Figure 5-8b. Although regions R1,R2,R4, and R5 can be classified in a straightforward manner, R3 cannot. It should be split into two parts and one part associated with each body. Similar problems arises in Figures 5-8c and d.

SEGMENT proceeds to assign the regions to bodies based on a number of heuristics. If all of the bounding lines of a region belong to a single body, SEGMENT concludes that the region belongs to that body. For regions having lines belonging to several bodies SEGMENT needs to determine if the region should be assigned to a single body or should be split into faces belonging to two bodies. For simplicity it is assumed that no region need be split into parts corresponding to more than 2 bodies. A check is first made to see if any corners labeled BADL have lines from two separate bodies incident there (e.g. the points labeled "P" in Figures 5-8b and d). Such a corner is taken as evidence that one of the regions at the BADL should be split. The region that gets split is never the background. If neither of the regions at the "bad" BADL is background, the region which is split is the one corresponding to the angle greater than 180 degrees (R2 in Figure 5-8d). If the two lines at the BADL belong to bodies B1 and B2, one face is assigned to B1 and the other is assigned to B2.

If no BADL vertices of this variety are detected for a region, SEGMENT assumes that the region should be assigned to a single body and looks for evidence as to which one. If one of the bounding lines of this region is the stem of a T-joint (Figure 5-8a), then SEGMENT assigns the region to the body to which the T-stem is assigned. Otherwise, if a bounding line of the region is incident at a BADY where two of the lines have been assigned to one body and the third to another body (Figure 5-8c), SEGMENT assigns the region to the body to which this third line belongs. In Figure 5-8a R4 get assigned to Body 2 based on either L2 or L5. In Figure 5-8c R gets assigned to Body 2 based on either of the labeled BADYs.

At this point the line segmentation procedure has presumably "partitioned" the set of lines in the scene according to body. The region assignment procedure has generated a "partition" among the regions in the scene according to body. Based on this information it is not difficult to come up with a complete description of each

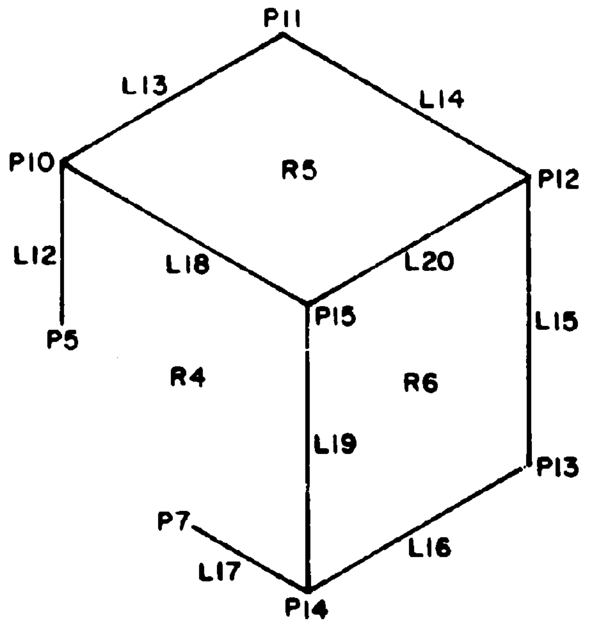


Figure 5-9 . A Partial Line Drawing of a Single Body .

(partially) visible body. Consider the scene of Figure 4-1c. After segmentation the description for the occluded body (see Figure 5-9) is

```
BODY • SCENE ≡ BODY2
FACE • BODY2 ≡ (R4, R5, R6);
LINE • BODY2 ≡ (L12, L13, ...,L20)
POINT • BODY2 ≡ (P5, P10, P11, P12, P13,
                          P14, P7, P15)
ENDPOINT • L12 ≡ (P5, P10)
.
.
ENDPOINT • L20 ≡ (P12, P15)
BOUNDARY • R4 ≡ (L12, L19, L18, L17)
BOUNDARY • R5 ≡ (L13, L14, L18, L20)
BOUNDARY • R6 ≡ (L15, L16, L19, L20)
CORNER • R4 ≡ (P5, P10, P15, P14, P7)
CORNER • R5 ≡ (P10, P11, P12, P15)
CORNER • R6 ≡ (P12, P15, P14, P13)
```

This description is the input that the single body recognizer, RECOGNIZE, accepts.

Some Thoughts on Handling Extraneous Lines:

Extraneous lines may appear in the line drawing due to reflections, noise in the system, or as the result of one object casting a shadow on another. Figure 5-10 presents several examples. For the single bodies in Figure 5-10a and Figure 5-10b, it is easy to convince oneself that SEGMENT assigns the lines to the bodies appropriately, that is, a single body is reported in each case. The difficulty, however, is that the descriptions passed on to RECOGNIZE will be bad because of additional lines and regions. In Figure 5-10c the bodies will again be segmented properly except for the extra line and region R. In Figure 5-10d things are somewhat worse and the segmentation procedure reports only a single body. Figure 5-10e illustrates a line drawing with both additional and missing lines. It is clear that arbitrarily meaningless line drawings can be imagined.

Although we have not considered the problem of extraneous lines in the depth that we have considered the problem of missing lines, a few ideas about how such cases might be handled seem appropriate. A likely result of the presence of an extra line is the generation of a triangular region (R in Figures 5-10a, c, and d). While certainly not all triangular regions are shadow or noise regions, triangular regions where 2 of the corners are T-joints as in

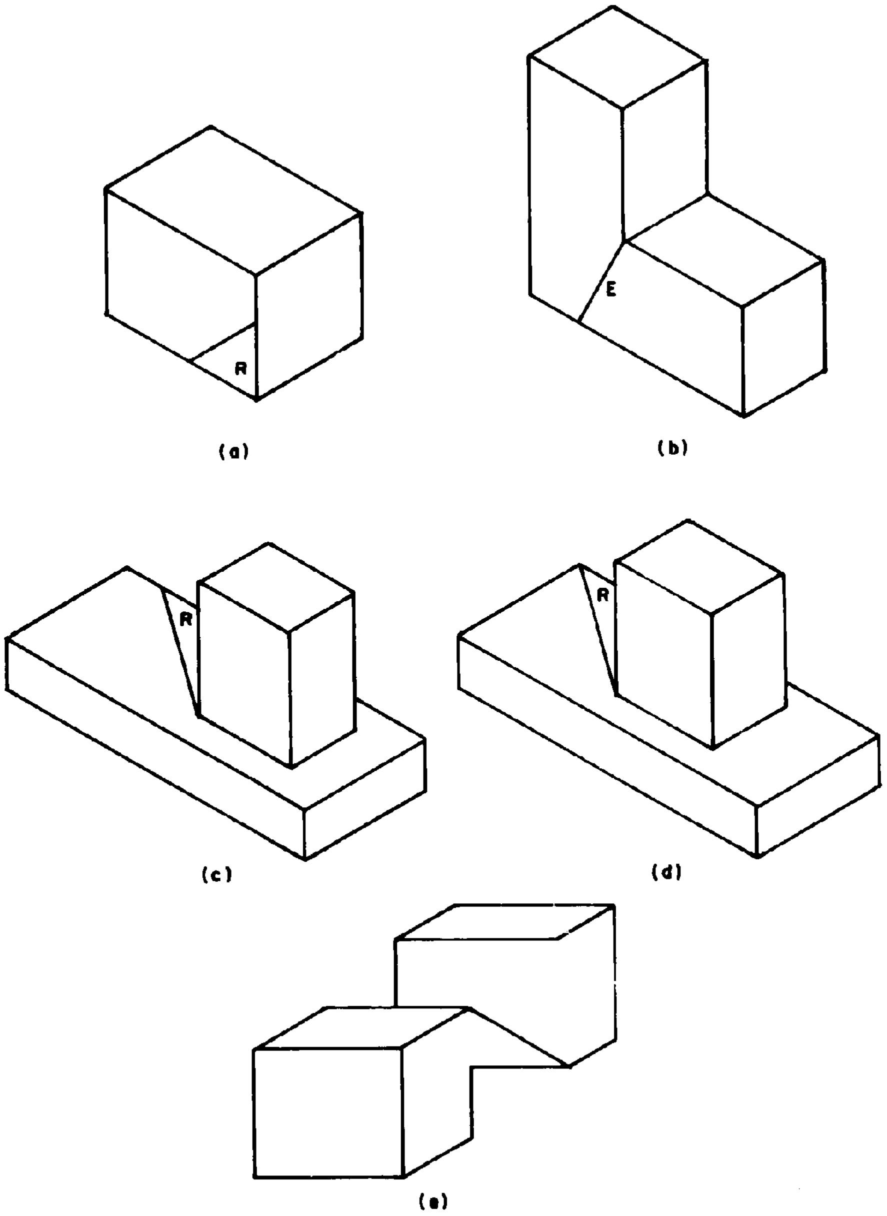


Figure 5-10 . Some Scenes Containing Extraneous Lines .

Figures 5-10a and c are highly suspect. The edge which is not the top of either T-joint is in each case the shadow edge. In Figure 5-10b the MULTI vertex is a strong indication that something unusual is happening. Often MULTI vertices arise where an edge of one body happens to intersect an arrow or Y vertex of an occluding body. The additional T-joint, however, gives a strong indication that the connecting edge, E, is probably an extraneous one.

The above heuristics would undoubtedly be valuable in enhancing the usefulness of our segmentation procedure. Line eliminating heuristics could easily be incorporated into SEGMENT. Handling situations such as Figure 5-10e, however, may still be beyond the system's grasp. A more model-directed approach will probably be necessary to handle scenes such as this. A number of additional ideas can be found in a recent report by Orban [29] describing a shadow eliminating preprocessor for Guzman's program SEE.

SUPPORT RELATIONS

The problem is to determine the support planes for each of the bodies in the scene. As indicated in Chapter 4, knowledge of the support planes for an object allows RECOGNIZE to identify and locate the object from only a single view. The relatively costly operations of focus or stereo ranging appear to justify a fair amount of effort being spent to infer depth from monocular cues.

Unless evidence to the contrary appears, COMPLEX assumes that the objects are resting flat on top of one another in planes parallel to z=0. First, COMPLEX determines the base edges of each body in the scene. Next a set of potential supporters for each object is determined from a knowledge of the base edges and the geometry of the original image. Finally, during the recognition process for an object (after all the potential supporters of the object have been recognized and positioned in space) COMPLEX examines each potential supporter so as to rule out those whose potential support planes are not parallel to z=0 and those which are not the tallest.

Base Edge Determination:

COMPLEX employs a number of heuristics to determine which lines of a body correspond to base edges. Initially, all the lines of a body are considered as potential base edges. A sequence of tests (filters) is then applied to eliminate those lines which should not be included. In the majority of cases, the base edges are determined after checking only a few of these conditions. The relatively large number of rules are required to handle the "exceptional" cases which occur infrequently.

The tests which COMPLEX applies to the individual bodies are the following:

(1) Eliminate all body lines that are not exterior, that is, they do not border on the background. (For the projection of Figure 5-11a we know the base edges must be a subset of (a,b,c,d,e,f,g,h,i)).

(2) Eliminate all lines appearing vertical in the image unless the bottom point of the vertical has 3 lines incident at it. (This rules out lines a and g in Figure 5-11a but not line a in Figure 5-11b).

(3) Eliminate lines connected to the top of lines eliminated in (2); (Eliminates b and f in Figure 5-11a).

(4) Eliminate lines at downward open L-type vertices (Eliminates b,c,e, and f in Figure 5-11a)

(5) Eliminate lines meeting at arrow or T-type vertices where the middle line points downward. (Eliminates c and d in Figure 5-11a and a and b

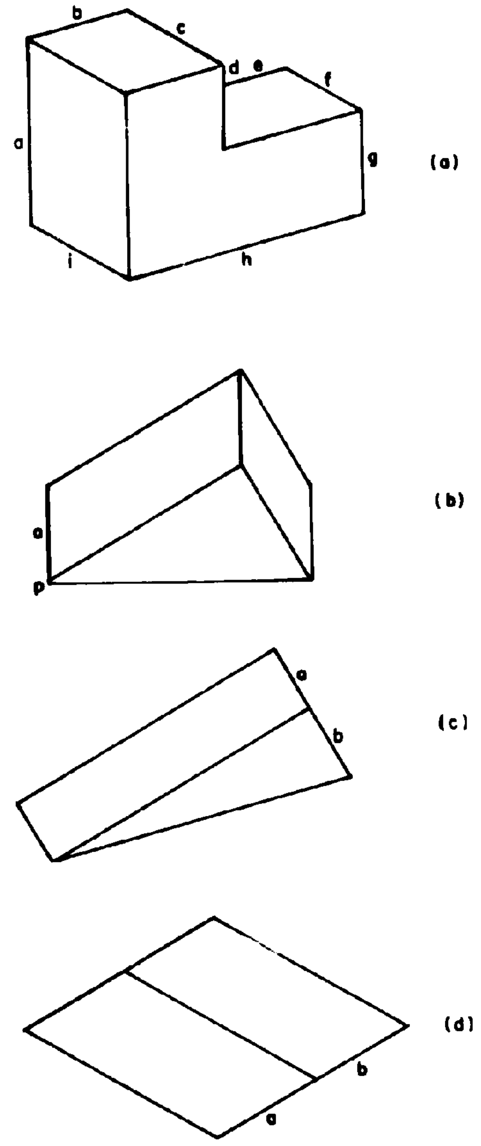


Figure 5-11 . Examples of the ways in which Base Edges Appear.

in Figure 5-11c)

(6) If the lowest point on the body is an L-type vertex, eliminate the line incident there with the largest absolute slope, i.e. do not keep 2 boundaries of any visible face as base edges. (Eliminates line a in Figure 5-11d)

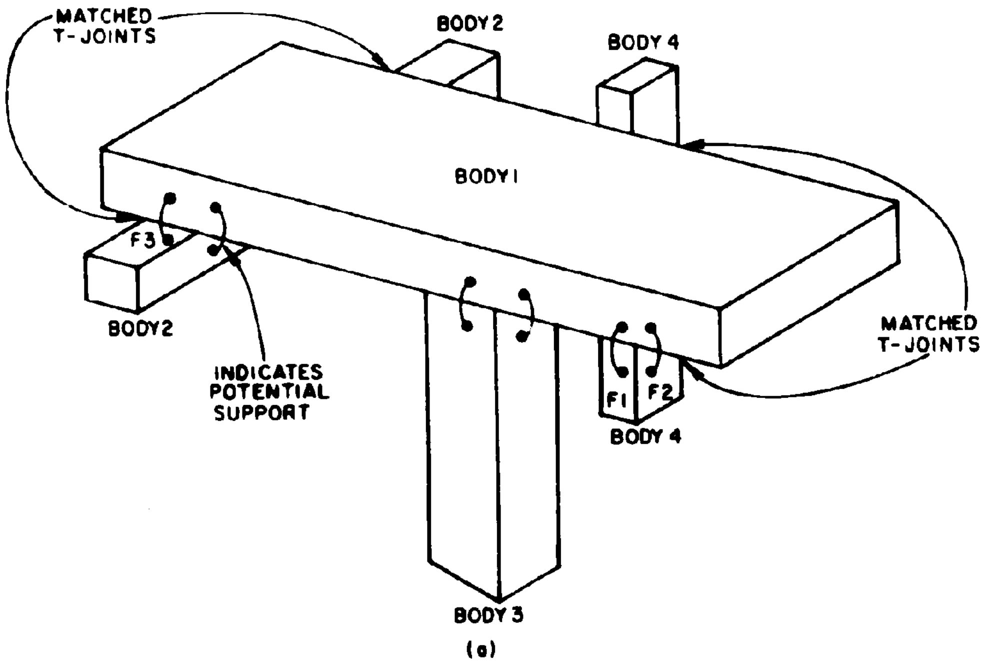
(7) If either of the outside lines at a T or an arrow vertex has been eliminated, eliminate the other outside line. (Eliminates line b in Figure 5-11d)

After all of these test have been applied, COMPLEX assumes that the remaining lines are the actual base lines. While there are certainly other ways to detect the base edges of an object (see for example [43]), the above approach appears adequate for all the scenes we have attempted to analyze.

Potential Supporter Determination:

For each pair of bodies, Body1 and Body2, COMPLEX must determine if Body2 supports Body1. It is relatively easy to answer this question either "NO" or "MAYBE" based on a single view once the base edges of all objects have been identified. For one body to support another it must first be the case that the bodies are adjacent. SEGMENT checks for this by searching for an edge in the line drawing that bounds both a region of Body1 (R1) and a region of Body2 (R2). If such a line exists and is (part of) a base edge of Body1, COMPLEX records Body2 as a potential supporter of Body1. The face of Body2 providing the support is the face corresponding to region R2 in the original line drawing. If no such line exists, COMPLEX concludes that Body2 does not support Body1. Since the existence of such a line only provides an indication that Body2 might support Body1, all such lines must be recorded for later confirmation or rejection. In Figure 5-12a the potential support indicators are represented by the little links between the appropriate regions.

Although the set of potential supporters often agrees with what people come up with when asked to describe the same scene, discrepancies do arise. Figure 5-12a illustrates one of these situations. When asked to describe the support relations in this scene, most people say that Body3 supports Body1. COMPLEX, however, says that besides Body3, Body2 and Body4 are also potential supporters of Body1. One of these discrepancies is quite understandable. If Body3 were not quite so tall, it could also be ambiguous to people whether Body2 or Body3 or both support Body1. COMPLEX postpones such decisions until after both Body2 and Body3 have been recognized. At that time it knows the



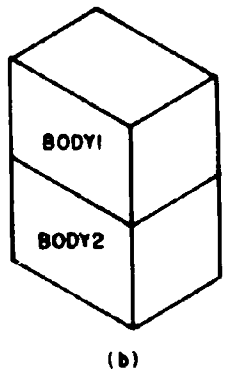


Figure 5-12 . Support Relations .

height of the top of each block from the known size of prototypes and can make the proper choice (the highest).

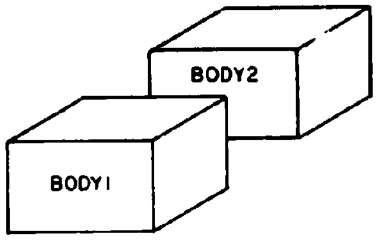
The link indicating that Body4 is a potential supporter of Body1 is handled somewhat differently. One would assume that people tend not to make this mistake because F1 and F2 are perceived as being vertical (compare with F3 of Body2) and the two parts of Body4 are seen as hidden behind Body1. Although COMPLEX could attempt to rule out Body4 as a supporter of Body1 based only on 2-D information, it seems more appropriate to also postpone this decision until after Body4 has been recognized. At that time transformed normals to F1 and F2 can be computed. Since neither face is horizontal (i.e. parallel to z=0), COMPLEX concludes that Body4 does not support Body1. This rule is only applied where there is at least one pair of matched T-stems that intersect a base edge of the supported body. Since all of the T-stems of Body3 intersecting the base of Body1 are unmatched, the top of Body3 is assumed to be invisible.

The only special support situation about which COMPLEX knows is shown in Figure 5-12b. The "X" vertex with its vertical collinear segments indicates that Body2 supports Body1.

OTHER STRUCTURAL RELATIONS

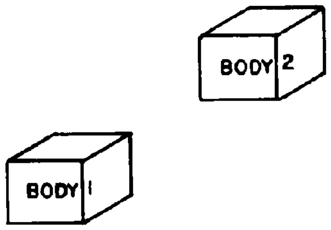
The previous sections have described the derivation and application of a particular structural relation, SUPPORT. Some of the other structural relations which might also be considered are shown in Figure 5-13. The simplest of these is the OCCLUSION relationship. The chief indicator of OCCLUSION is the T-vertex. In general, the body owning the two collinear T-tops is the occluder of the body which owns the T-stem. Occlusion can also be indicated by K, X, MULTI, and BADY vertices. These may be thought of as degenerate T-joints.

OCCLUSION is the only relation other than support which is currently being utilized by COMPLEX. If no base edge of an object is visible (and the object will consequently not be recognized), COMPLEX suggests to the STRATEGIST program that it would be appropriate to either rotate the scene through some large angle or to move those objects occluding the object in question. The scene analysis can subsequently be tried again.



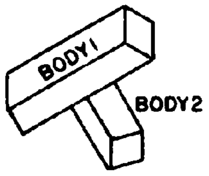
BODY1 Occludes BODY2

BODY2 Behind BODY1

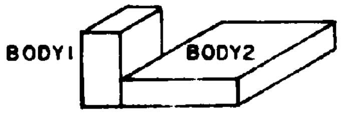


BODY2 Behind BODY1

BODY2 Right of BODY1



BODY1 Leans on BODY2



BODY1 Abuts BODY2

Figure 5-13 . Other Structural Relations .

BLANK PAGE

CHAPTER 6

RECOGNITION OF OBJECTS IN THE SCENE

Having determined the partial projections corresponding to individual objects and developed some understanding of how the individual bodies in the scene are arranged, COMPLEX proceeds with the recognition phase. Recognition consists of both identifying an object with a prototype and locating the object in space. We refer to the routine which does this as RECOGNIZE. After all of the bodies have been processed, a predicted line drawing is generated and compared with the input. A match indicates that the analysis has been successful. The remainder of this chapter describes each of these processes in detail.

SIMPLE COMPLETION

Before recognition is attempted, the simple completion routines try to complete partial line drawings of a single object (see Figure 6-1). 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 RECOGNIZE does not demand that the projection it is presented be complete, its chances of success are better when there is no data missing.

There are three completion procedures, JOIN,

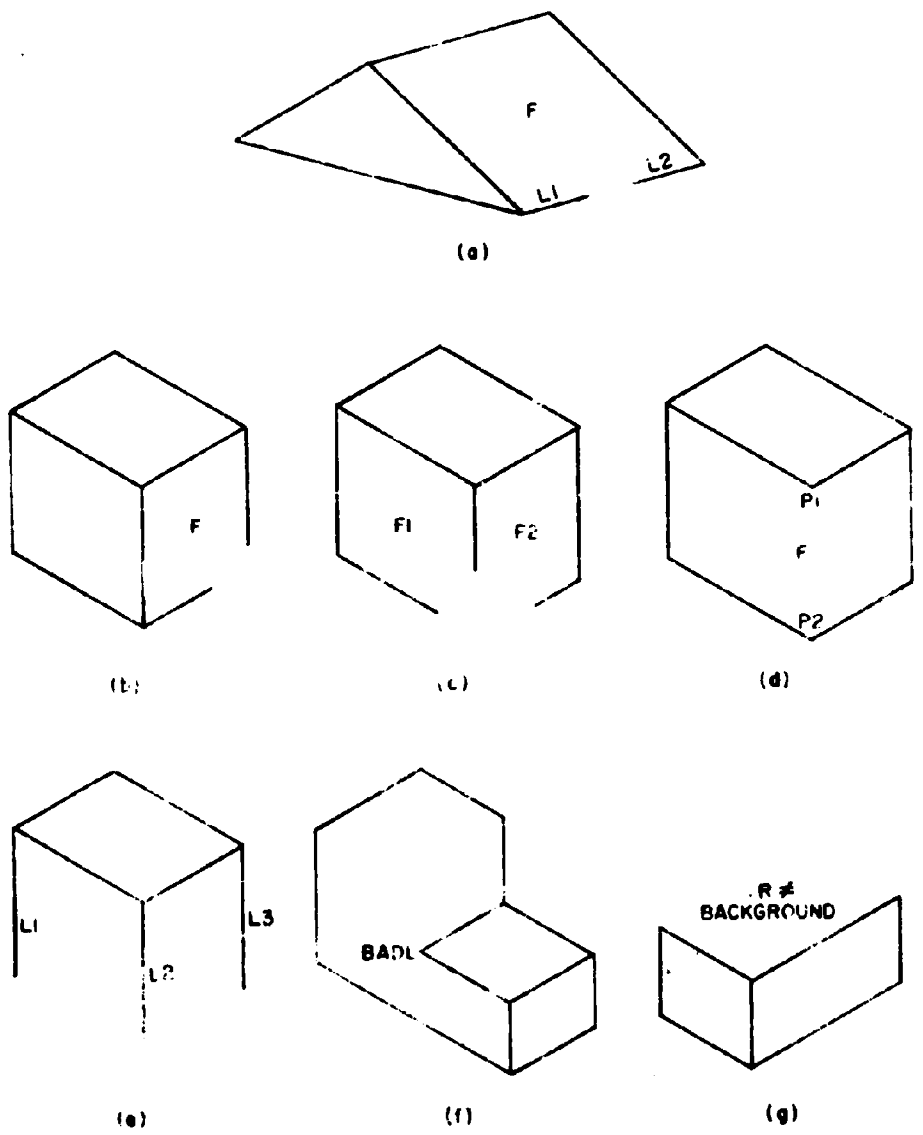


Figure 6-1 . Cases in which the Completion Routines can be Applied .

ADDCORNER, and ADDLINE. JOIN handles cases similar to that of Figure 6-1a. It looks for a face, F, which two hanging collinear lines indicate is incomplete. It replaces these two lines (L1 and L2) by a single line and updates the rest of the associations to reflect this fact.

ADDCORNER handles cases such as Figures 6-1b and c. It looks for a face that is incomplete due to two hanging lines (F of Figure 6-1b) which can be extended to form a corner. It completes the face by adding this new corner and again updates the rest of the associations. Figure 6-1c is handled by first completing F1 and then completing F2. One must, of course, have limits on how far an edge may be extended so that neither L1 and L2 nor L2 and L3 in Figure 6-1e are extended to infinity.

ADDLINE tries to find evidence that an entire line has been missed. Although there are no dangling lines in Figure 6-1d, something is definitely missing. ADDLINE adds a line between P1 and P2 based on the BADL at P1 and the pair of parallel lines incident at these corners. Face F is split into two faces and the rest of the data structure is updated accordingly.

Since all of our prototypes are trihedral, one can legitimately apply a "Huffman Labeling" to the (partial) projection. It appears that in some cases this labeling can not only detect that the projection is invalid (incomplete), but also indicate where a new line might be added. For example, In Figure 6-1d both of the lines meeting at the vertex P1 must be labeled "+". This indicates that something is wrong near this vertex. To date we have not explored this approach in any detail.

IDENTIFYING THE OBJECT AS AN INSTANCE OF A PROTOTYPE

RECOGNIZE identifies or "names" an object by extracting features from its line drawing projection, P, and matching these against the properties of the 3-D stored prototypes. If we define PROTOTYPES to be the set containing all the prototype items, we can represent this process schematically as:

$$\text{PROTOTYPES} \rightarrow T_1(P) \rightarrow S1 \rightarrow T2(P) \rightarrow \ldots \rightarrow T_n(P) \rightarrow Sn \quad .$$

RECOGNIZE applies a test, T1, to the projection P. Based on the outcome of this test it can say that P could only correspond to members of the set S1 where S1 is a subset of PROTOTYPES. Similarly, it applies subsequent tests to further restrict the possible interpretations of projection P. Hopefully, for some i, $1 \leq i \leq n$, Si is a singleton and the object is identified. If this is not the case, then the RECOGNIZE reports only that the object viewed is one of the members of set Sn.

The set of tests are chosen such that a projection can usually be identified even if several lines are missed because of occlusion or noise. This type of procedure is applied both by RECOGNIZE in the case of (partially) complete line drawings and by SIMPLE for outlines. The specific tests, of course, differ in these two situations. The sequence of tests is fixed and chosen roughly so as to apply the "cheapest" tests first. RECOGNIZE does not try to find a true "least cost" sequence of tests.

The tests can be divided into two basic classes, topological tests and geometric tests. The topological tests compare features which are projective invariants and consequently can be examined directly from the projection. The geometric tests match such things as lengths and angles which are not projectively invariant. Three-dimensional information must be inferred from the projection before matching of geometrical properties can be accomplished. The tests themselves are extremely simple because there are not very many properties which distinguish the solids of Figure 1-1. The order in which we describe the tests is the order in which RECOGNIZE applies them.

Topological Tests:

After the 3 completion routines have been applied, RECOGNIZE checks to see how successful they were. It assumes that a partial projection corresponding to a single object is complete if it has no remaining dangling lines as in Figure 6-1e, no internal BADLs as in Figure 6-1f, and no face with less than 3 bounding lines as in Figure 6-1g.

(1) For a complete line drawing corresponding to a single object one might think first of matching the line drawing (treated as an undirected planar graph) against the edge structure of each model. While such an approach is possible for convex solids, general subgraph matching is not easily implemented. RECOGNIZE, therefore, takes a slightly different approach. For each model there are a finite (usually small) number of topologically distinct views. For each view we predetermine the number of faces and the number of vertices that are visible and associate this "projection pair" with the corresponding model. Figure 6-2 shows the set of distinct projections of a wedge and their corresponding projection pairs. When a complete line drawing is detected, the number of faces and number of vertices visible are determined and matched against the pairs stored with each model. With the exception of degenerate views, there turns out to be few pairs common to more than one topologically distinct model.

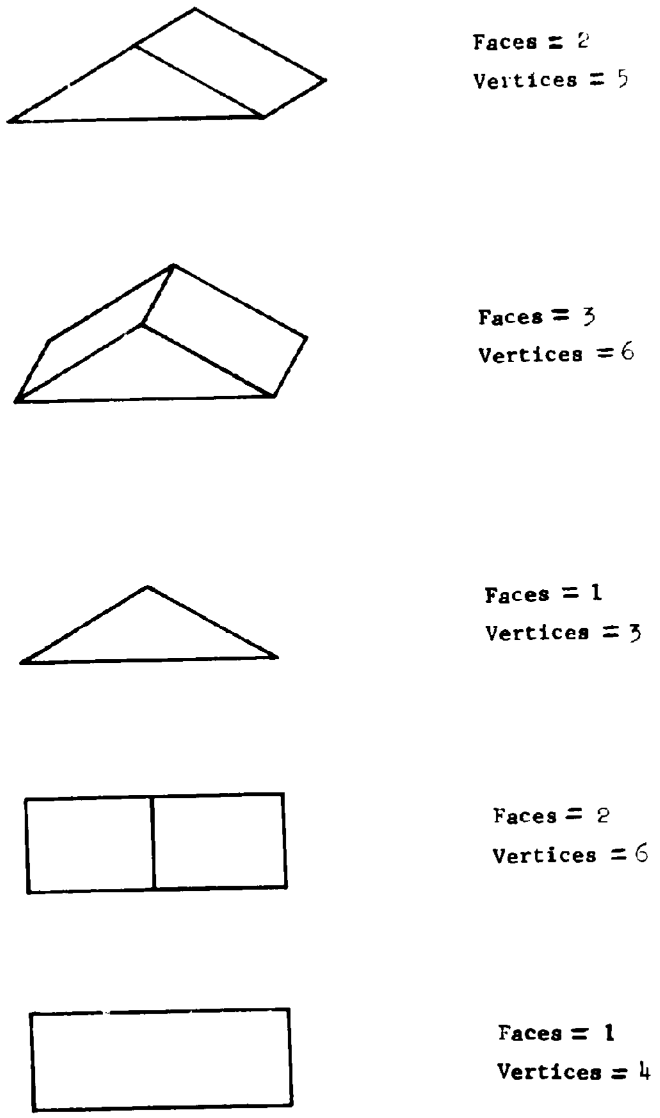


Figure 6-2 . The Projection Pairs for a Wedge .

If the line drawing is not complete then the face types are used to partition the set of possible matches. For example, if a triangular face is present, then the projection could not possibly be a rectangular parallelipiped. Similarly, the existence of a hexagonal face indicates an LBEAM.

The only topological property of an outline is its number of edges.

Geometric Tests:

The tests in this class depend on obtaining depth information, i.e. 3-space location of points, by one of the methods described in Chapter 3. Since corners can be located substantially faster by support hypothesis than by any other method, RECOGNIZE tries to match base edge lengths and angles first.

(2) RECOGNIZE finds the lengths of any base edges which are either totally or partially visible. For a complete edge of length L, it rules out any model not having at least one edge, E, such that L-DELTAL<length(E)<L+DELTAL where DELTAL is an appropriately chose tolerance. For an incomplete (dangling) base edge of length L , a potential model must have at least one edge of length geater than L-DELTAL.

(3) If there is a visible corner where two base

edges meet, RECOGNIZE computes the angle, THETA, between these edges. A model is only accepted as a potential match if one of its faces has an angle within DELTATHETA of this angle.

(4) For the scenes that RECOGNIZE considers, it is usually possible to neglect perspective effects during recognition. In particular, edges normal to the table too appear approximately vertical in the image. RECOGNIZE also assumes that a vertical line in the image arises from an edge normal to the table (this may occasionally not be true). It uses the technique described in Chapter 3 to determine the actual length of such an edge. The lengths of vertical edges are used to rule out possible matches as in (2).

(5) If RECOGNIZE still has not succeeded in identifying the projection , it will compare the known edge lengths incident at visible corners of the projection against those incident at corners in the object models.

LOCATING THE OBJECT IN SPACE

After deciding to which model a particular object in the scene corresponds, RECOGNIZE proceeds to determine its location in space. The most obvious way to do 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 system. The techniques described below provide a convenient way to record and compute these 6 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 coordintes as PJ=(XJ,YJ,ZJ,1) and those of the instance as PJ'=(XJ',YJ',ZJ',1) then we can write

$$PJ' = T\ PJ$$

where

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

The upper richt-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 hereafter mean finding the matrix T which moves the matching prototype to the position of the instance.

There are 6 parameters needed to specify T, Therefore, knowing the locations of 3 corresponding points on the object and model will allow T to be computed. The 6 parameters are DX, DY, DZ, and the 3 angles implicit in R.

We now describe how RECOGNIZE determines 3 corresponding points o: the object and model.

Matching:

In Figure 6-3 it is clear that a rigid body motion can cause either A',B', and C' or A'', B'', and C'' of the prototype in Figure 6-3b to be aligned with A, B, and C of the object in Figure 6-3a. On the other hand, A, B, and C will not match A''', B''', and C''' of the model in Figure 6-3c. We refer to this problem of finding corresponding points as the "matching problem".

RECOGNIZE matches an object to a prototype in several steps. First it locates 4 independent (non-coplanar) connected corners of the object. Next, it computes the lengths of the edges connecting these 4 points. It then uses these lengths to partition the set of model edges into subsets which could correspond to each of the 3 object edges. Finally,RECOGNIZE selects triples of model edges from these subsets. The first triple having certain features in common with the 3 object edges is taken as the correct match. We shall elaborate on each of these points.

Although 3 points are sufficient to specify the

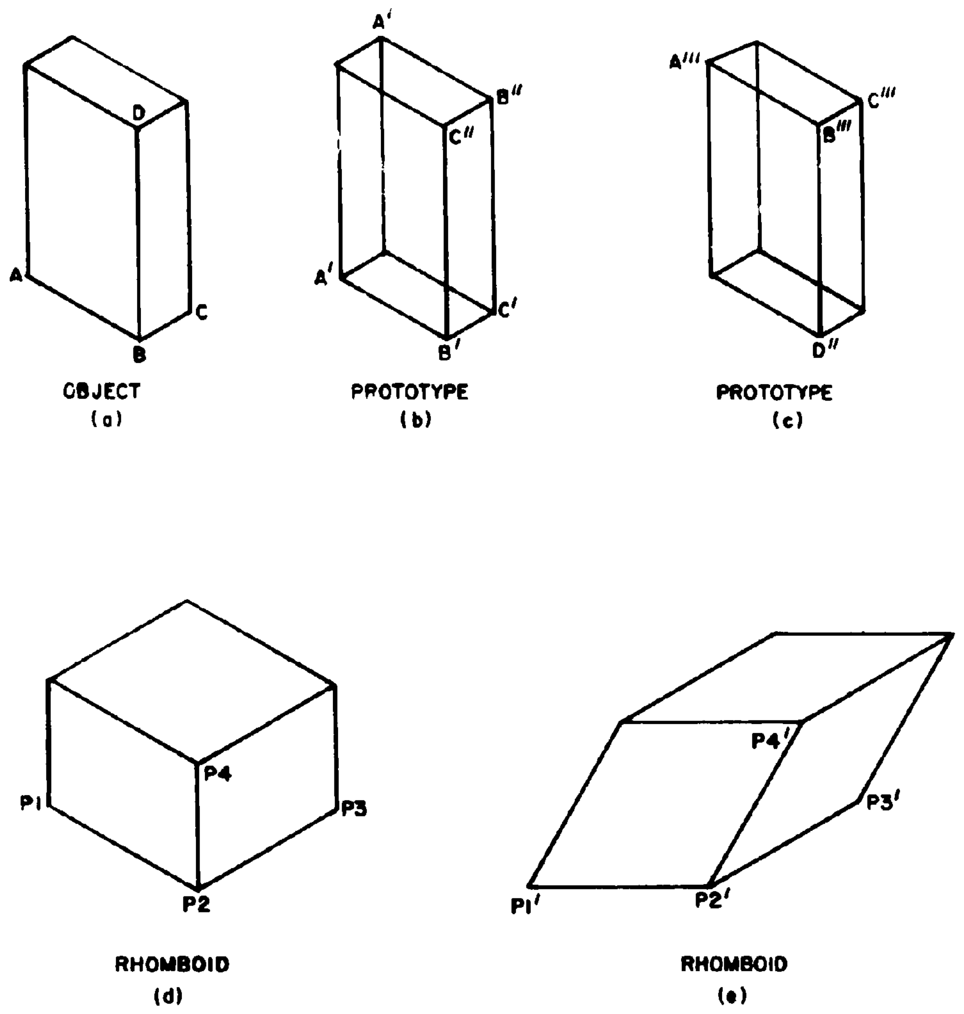
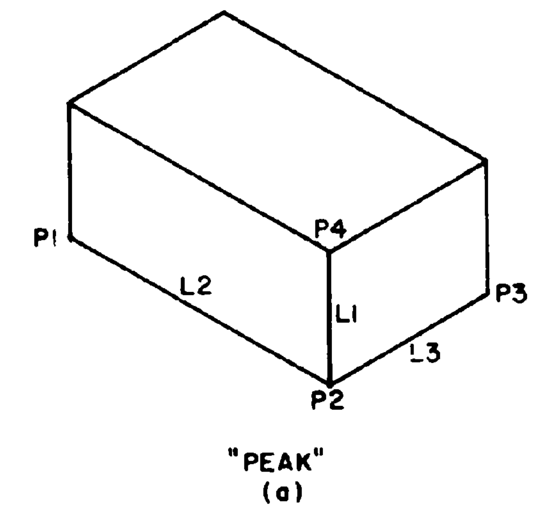


Figure 6-3 . Matching an Object with its Prototype .



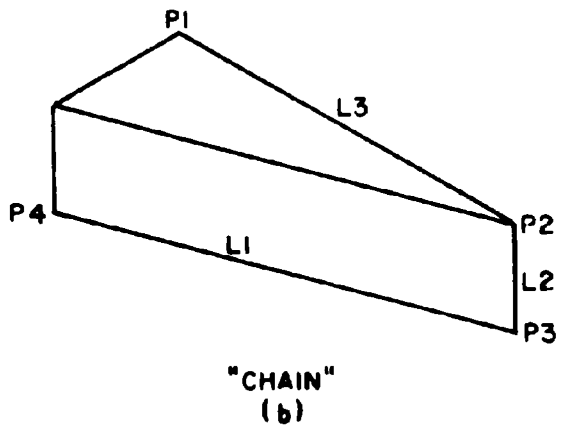


Figure 6-4 . Example of a PEAK and a CHAIN .

transform after a point correspondence has been set up, the fourth point greatly facilitates the matching process. The points, P1,P2,P3, and P4, are selected such that the lines having them as end points form either a "peak" as in Figure 6-4a or a "chain" as in Figure 6-4b. Usually these four points have been located during the identification phase of the recognition process. If not, RECOGNIZE proceeds to locate the remaining corners. The techniques described in Chapter 3 for locating a point based on assumptions about a face are applicable now since the prototype corresponding to the object is known.

Having determined P1, P2, P3, and P4, RECOGNIZE computes the actual lengths of the edges L1, L2, and L3 (see Figure 6-4). Associated with each Lj is a set SLj where SLj contains those edges of the model that are within a prespecified tolerance of the length of Lj. As possible matches to (L1,L2,L3) RECOGNIZE chooses (t1,t2,t3) such that tj∈SLj. If the 3 object lines form a peak, then the 3 model edges must also form a peak. Similarly, if the object edges form a chain then so must the model edges.

A simple procedure eliminates such errors as Figure 6-3a matching Figure 6-3c. RECOGNIZE proceeds as fillows:

(1) It determines V2 = P1 - P2,

V3 = P3 - P2,

and V1 = P4 - P2 for a peak or

V1 = P4 - P3 for a chain.

(2) It computes the dot product $V1 \bullet (V2 \times V3)$.

(3) If PJ' is the model point potentially matching object point PJ then RECOGNIZE computes

$$V2' = P1' - P2'$$

$$V3' = P3' - P2'$$

and $V1' = P4' - P2'$ for a peak or

$V1' = P4' - P3'$ for a chain,

(4) Finally, it computes the dot product $V1' \bullet (V2' \times V3')$.

(5) For a match it requires that the two dot products have the same sign.

If we identify P1, P2, P3, and P4 with A, B, C, and D in Figure 6-3a, the object gives $V1 \bullet (V2 \times V3) < 0$ whereas the match implied by the model in Figure 6-3c gives $V1' \bullet (V2' \times V3') > 0$. RECOGNIZE does not require that these two dot products are equal because some of the object lines may be partially occluded.

These conditions are still inadequate since points P1, P2, P3, and P4 of Figure 6-3c could match P1', P2', P3' and P4' of Figure 6-3d. To rule out such mismatches RECOGNIZE requires that the angles between correspondng lines be the same.

So far we have neglected the fact that one or more of the object lines may be partially occluded. RECOGNIZE assumes that it can find a peak or a chain where not more than one line of the three edges is occluded. If LJ is partially occluded, then it puts into SLj only those model edges having lengths greater than that of LJ. If line LJ in the projection is partially occluded and matches line LJ' in the model, RECOGNIZE sets up a "pseudo model point" to match the occluded end of Lj. The point is located |LJ| down the model edge LJ'.

Once 4 corresponding points are identified, RECOGNIZE may use any 3 of them to compute the transform T. It seems reasonable to choose the 3 points which have the longest two edges connecting them. This reflects the fact that we presumably have greater confidence in long lines than short ones.

Computing the Transform:

Given 3 corresponding points P1-P1', P2-P2', and P3-P3' and the two associated vectors V2-V2' and V3-V3' on the model and object RECOGNIZE proceeds to compute T as

$$T = D R .$$

Here R is a 4x4 matrix specifying a rotation and D specifies a translation. RECOGNIZE begins by setting up two orthogonal coordinate systems centered at P2 and P2'. The unit vectors U1=V2xV3/|V2xV3|, U2=V2/|V2|, and U3=U2xU1 specify the axes of one system. The second system is specified by U1'= V2'xV3'/|V2'xV3'|, U2'=V2'/|V2'|, and U3'= U2'xU1'. RECOGNIZE then computes

$$R1 = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ u1 & u2 & u3 \\ \downarrow & \downarrow & \downarrow \end{bmatrix} , \qquad R2 = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ u1' & u2' & u3' \\ \downarrow & \downarrow & \downarrow \end{bmatrix} ,$$

$$R* = R1\ R2^T ,$$

and $$d = [DX, DY, DZ] = V2 - R*\ V2'.$$

It follows that

$$R = \begin{bmatrix} & & & 0 \\ & R* & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } D = \begin{bmatrix} 1 & 0 & 0 & DX \\ 0 & 1 & 0 & DY \\ 0 & 0 & 1 & DZ \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Although only 3 matched points are required to specify the transform, T, one might expect that it would be better to estimate T based on more than 3 points. We have not found it necessary to do this. If this is done, however, one must not neglect the fact that the upper left 3x3 submatrix of T is required to be a pure rotation matrix. A parameter adjustment algorithm could easily be applied to DX, DY, DZ, and the three angles implicit in R* so as to minimize the cumulative error for all the points being matched.

SEQUENCING

The flowchart of Figure 6-5 shows the algorithm we are currently using to control the recognition of individual bodies in the scene. This is a more honest version of the bottom half of Figure 4-5. As was mentioned earlier, RECOGNIZE is generally applied in a bottom-to-top manner. The bodies resting on the table are recognized before the bodies which they support. The complexity of the flowchart stems from the facts that (1) all of the potential supporters of a given body need to be recognized before the true supporter can be determined (see Chapter 5) and (2) when recognition fails, COMPLEX needs to decide what to do next. If an object cannot be matched with any model and it is resting on the table, COMPLEX reactivates RECOGNIZE after relaxing the parameters which control the amount of deviation from a model which is tolerated. If the object still cannot be recognized, COMPLEX gives up on it.

If an object cannot be identified as any prototype

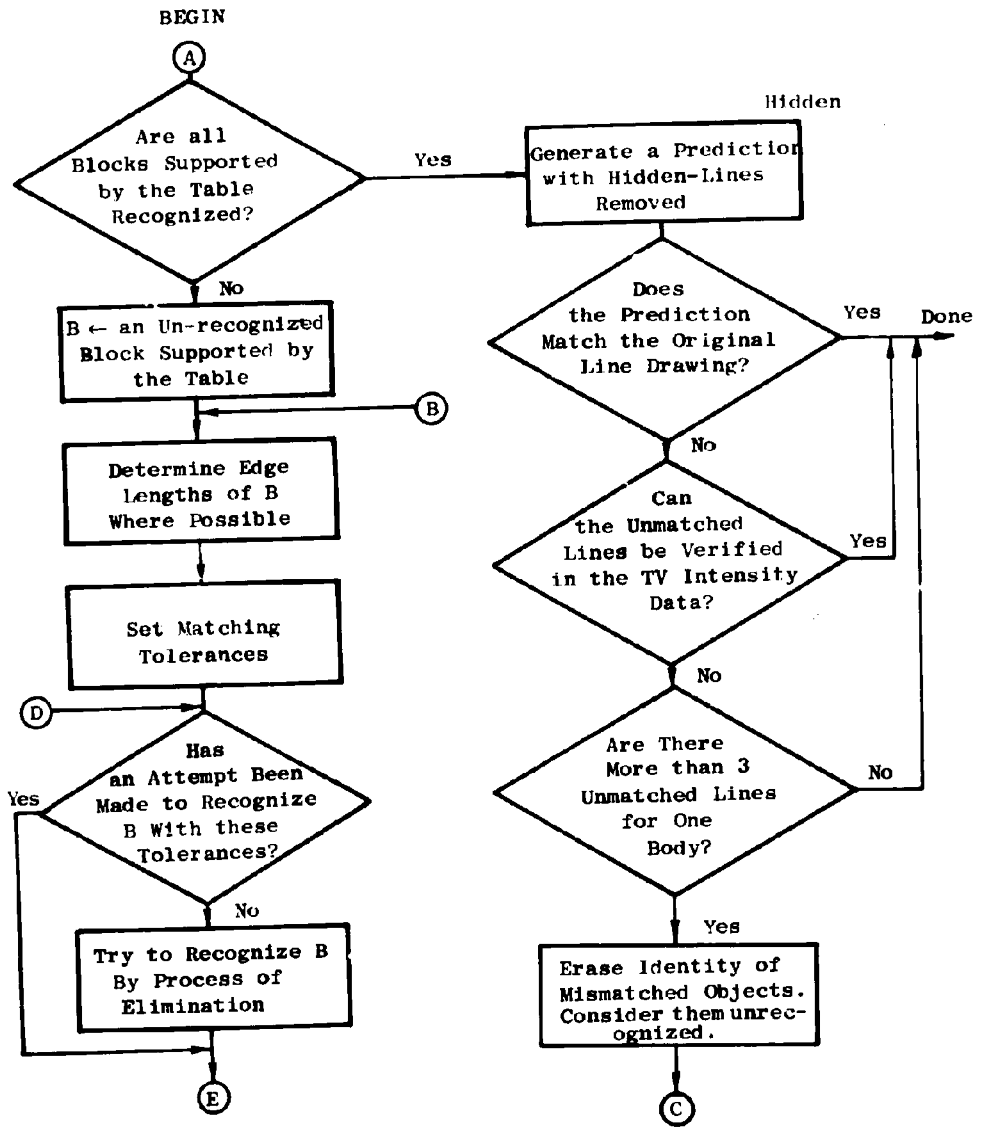

BEGIN
A
Are all Blocks Supported by the Table Recognized?
Yes
No
Hidden
Generate a Prediction with Hidden-Lines Removed
Does the Prediction Match the Original Line Drawing?
Yes
Done
No
B ← an Un-recognized Block Supported by the Table
B
Determine Edge Lengths of B Where Possible
Set Matching Tolerances
D
Has an Attempt Been Made to Recognize B With these Tolerances?
Yes
No
Try to Recognize B By Process of Elimination
E
Can the Unmatched Lines be Verified in the TV Intensity Data?
Yes
No
Are There More than 3 Unmatched Lines for One Body?
No
Yes
Erase Identity of Mismatched Objects. Consider them unrecognized.
C

Figure 6-5 . Operation of COMPLEX .

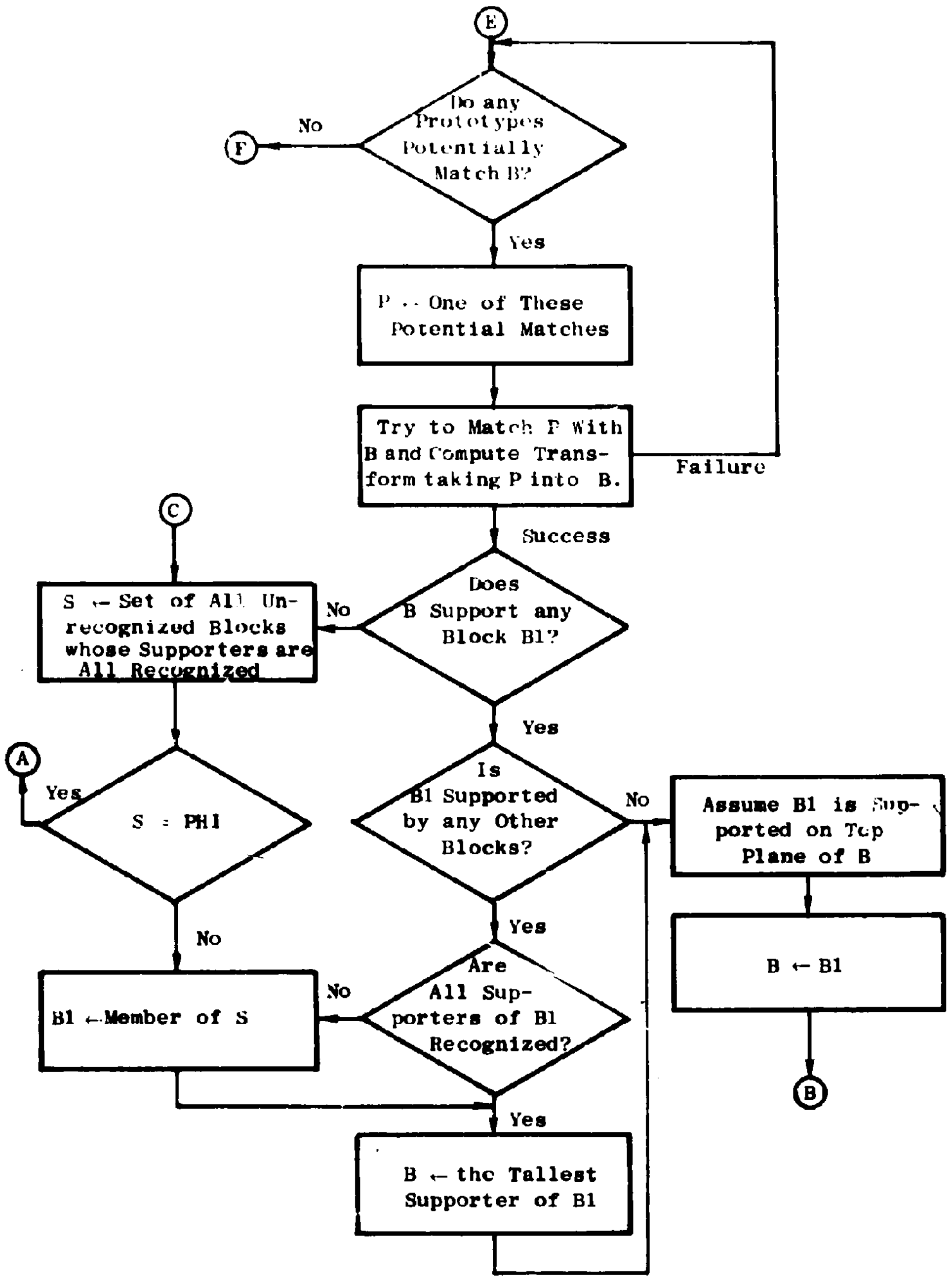

E
Do any Prototypes Potentially Match B?
No
F
Yes
P ← One of These Potential Matches
Try to Match P With B and Compute Transform taking P into B.
Failure
Success
C
Does B Support any Block B1?
No
S ← Set of All Unrecognized Blocks whose Supporters are All Recognized
Yes
A
Yes
S = PHI
No
Is B1 Supported by any Other Blocks?
No
Assume B1 is Supported on Top Plane of B
B ← B1
B
Yes
Are All Supporters of B1 Recognized?
No
B1 ← Member of S
Yes
B ← the Tallest Supporter of B1

Figure 6-5 . Operation of COMPLEX

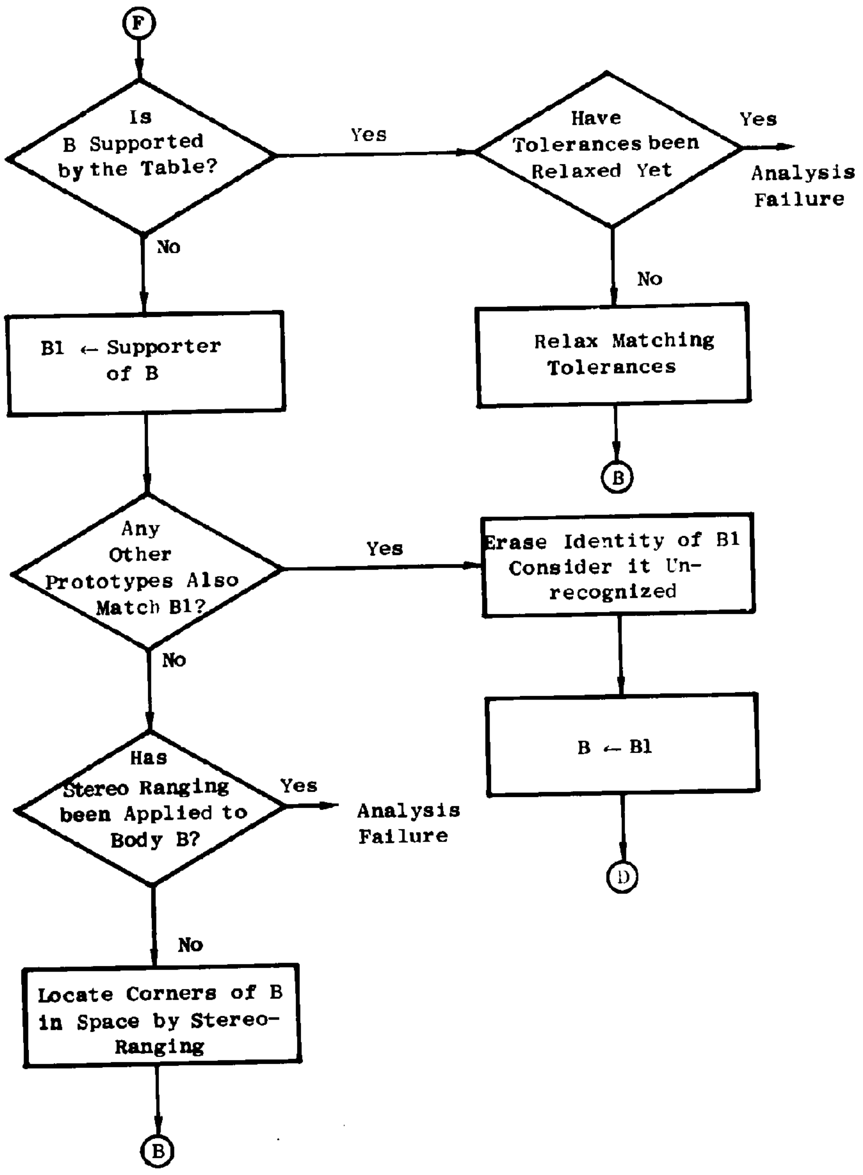

F
Is B Supported by the Table?
Yes
Have Tolerances been Relaxed Yet
Yes
Analysis Failure
No
No
B1 ← Supporter of B
Relax Matching Tolerances
B
Any Other Prototypes Also Match B1?
Yes
Erase Identity of B1 Consider it Un-recognized
No
B ← B1
Has Stereo Ranging been Applied to Body B?
Yes
Analysis Failure
D
No
Locate Corners of B in Space by Stereo-Ranging
B

Figure 6-5 . Operation of COMPLEX

and is supported by other objects, it may be either leaning on its supporter or the supporter itself may have been incorrectly identified. If the identification of the supporter led to only one possible prototype, COMPLEX assumes that it was correctly identified and calls on the stereo package to re-locate the corners of the supported object. If there were other possible prototypes which might have matched the supporter, COMPLEX tries one of them and continues with the recognition. The examples presented in the next chapter will bring these issues into sharper focus.

PREDICTION - THE HIDDEN LINE PROBLEM

After having recognized and located each of the objects in the scene, COMPLEX is in a position to predict how the scene should appear in the image. As mentioned in Chapter 2, this problem has been investigated rather extensively in recent years by workers in the area of computer graphics. A major goal of their reasearch has been to find efficient algorithms for solving the so-called "hidden line problem". For the extremely complex objects and scenes with which they deal, an efficient algorithm is a necessity rather than a luxury. COMPLEX, on the other hand, is currently not able to analyze scenes of such complexity and consequently is not as strictly constrained by the need for efficiency. The algorithm HIDDEN described below is quite simple. Although HIDDEN is probably not practical for scenes of more than about 20 simple objects, it does satisfy our needs at present. After presenting a simplified description of the operation of HIDDEN, we shall describe some speed-up techniques which have been employed.

Figure 6-6 illustrates the essence of the hidden line problem. Having determined to which model a particular object in the scene corresponds and the transform which moves the model out into its proper position in the real world, HIDDEN can predict where each corner of the object would appear in the image. There exists a 4x4 projection matrix P (see the description of the Roberts system in Chapter 2) such that multiplying the homogeneous representation of a corner by P yields the coordinates of the point in the image. The problem, of course, is that this process does not take into account the fact that the object is opaque. In Figure 6-6 lines a, b, and c should not actually appear in the image. The problem is even worse for concave bodies or groups of bodies. In these cases only parts of a given line may be visible. The algorithm below provides a method for determining which lines or parts of lines should be present in the image from a given point of view. The method proceeds in two phases.

During the first phase HIDDEN removes lines on "back faces" of individual objects. This technique is common to many existing hidden-line algorithms. The idea is that if a

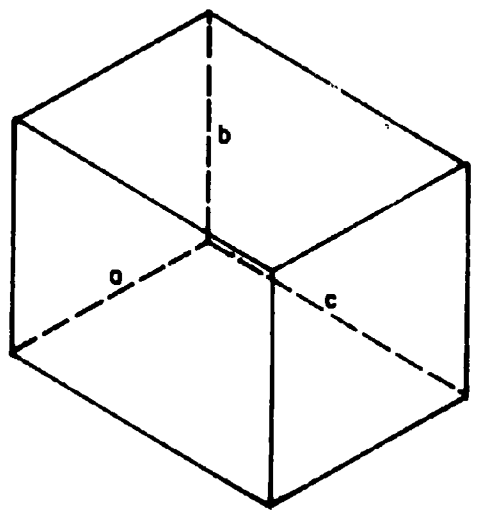


Figure 6-6 . The Hidden-Line Problem .

face of an object is totally hidden, then none of its bounding edges should be displayed (predicted). The invisibility of a face due to the object hiding itself is quite easy to determine from a comparison of the face normal, Nf, with the "line of sight". The line of sight or principal ray, PR, is a vector through the lens center which pierces the image plane at its center. If PIC is the center of the image, A is the collineation matrix, and C is the lens center, then PR = (PIC)(A) - C. The condition for total invisibility of a face due to self-occlusion is therefore:

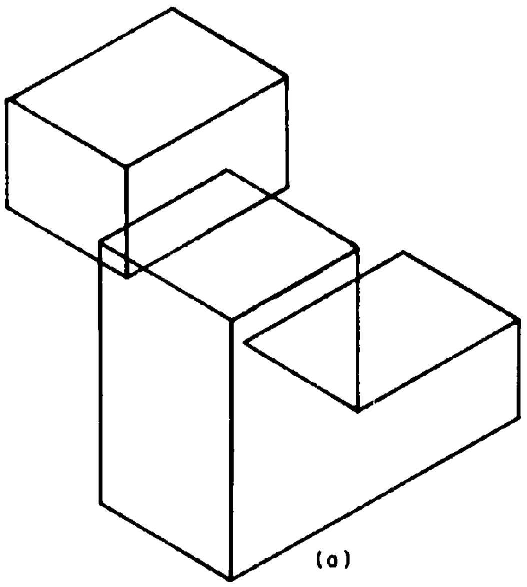
$$Nf \bullet PR \geq 0 \rightarrow F \text{ is invisible}$$

$$Nf \bullet PR < 0 \rightarrow F \text{ is visible} .$$

The normal to face F can be found as:

$$Nf = nf \; T^{-1}$$

where nf is the normal stored with the protctype which matches the object under consideration and T is the inverse of the 4x4 homogeneous transform which correctly positions the model in space. If all of our scenes consisted of only a single convex polyhedron, then all hidden lines would be removed by this point. Figure 6-7a, however, shows a



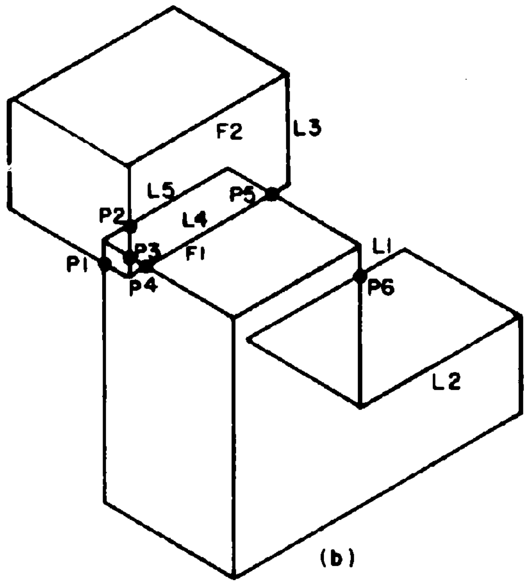
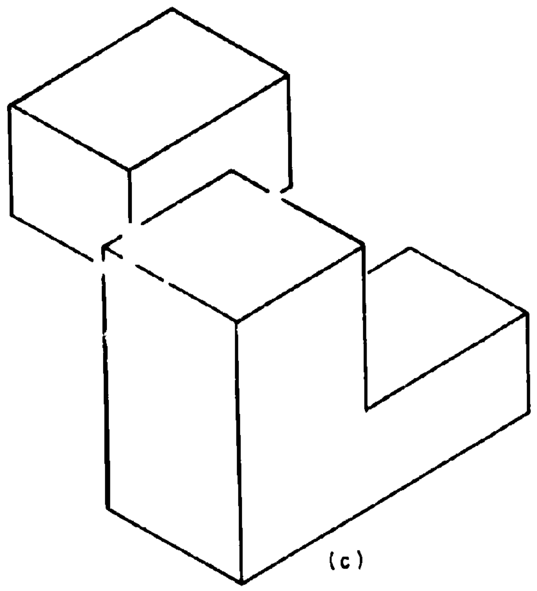


Figure 6-7 . Eliminating Invisible Segments .


(c)

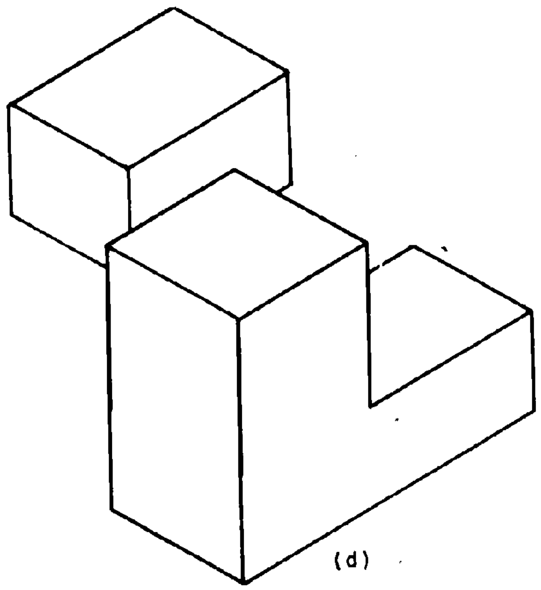

(d)

Figure 6-7 . Eliminating Invisible Segments typical image after the back face lines have been removed.

We now describe phase 2 of the hidden line removal procedure. HIDDEN begins by identifying all points in the image where lines cross, that is, P1, P2, P3, P4, P5, and P6 in Figure 6-7b. Note that these points do not exist in any internal data structure but only in the mind's eye of the viewer. They can be determined, however, by considering all pairs of lines for intersections.

After they have been located, any little segment (line between two points in the image) is either totally visible or totally invisible. To determine if a line is visible or not HIDDEN notes first that a line which is not enclosed in any of the other faces (e.g. L1, L2, and L3) must be visible. A line enclosed in another region may be visible (e.g. L5 enclosed in F2) or may be invisible (e.g. L4 enclosed in F1). To determine if a line enclosed in one or more faces is visible, HIDDEN must consider the 3-space location of this line relative to each of the face planes. If P is a point on segment L, to check for visibility relative to face F we compute:

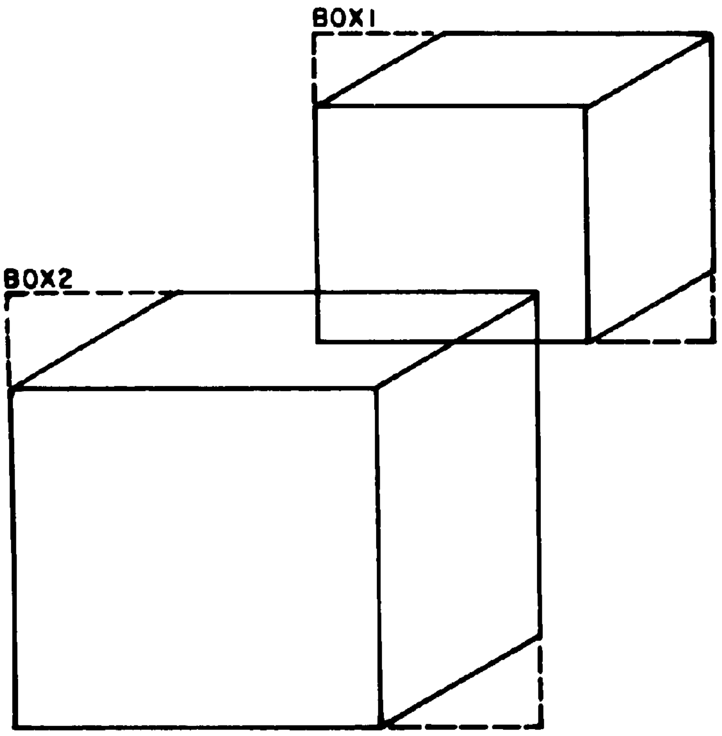
$$Nf \bullet P \geq 0 \quad \rightarrow \quad L \text{ is visible}$$

$$Nf \bullet P < 0 \quad \rightarrow \quad L \text{ is invisible} \; .$$

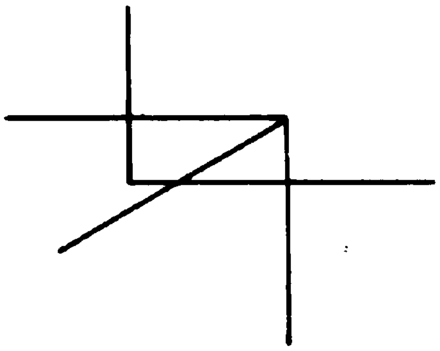
If the process described above is carried out correctly, a proper description of the scene viewed will eventually result (Figure 6-7c).

The only problem with this description is that some lines are broken up into several collinear segments. HIDDEN calls on a routine SPLICE which scans the scene for corners where two lines meet that are collinear. SPLICE replaces them with a single line. The result is the desired line drawing (Figure 6-7d).

Although the procedure as described above will work, some simple modifications can greatly improve its efficiency. One of the most time consuming parts of the procedure is the determination of the crossover points indicated in Figure 6-7b since each line must be compared with every other line for a possible intersection. Rather than do this directly, HIDDEN actually tries to determine the status (visible or invisible) of certain lines before computing any intersections. For each object in the scene it determines an "enclosing image box" based on the minimum and maximum values of x and y of the corners of the object (see Figure 6-8a). If line L belongs to the projection of a convex object (it is easy to mark the convex prototypes) or is a part of the projection of a concave object which is not hidden by the object itself, then it is clearly visible if it is outside the image boxes for all the other objects in the scene. Only the 5 lines shown in Figure 6-8b need be



(a)



(b)

Figure 6-8 . A Speed-Up Technique .

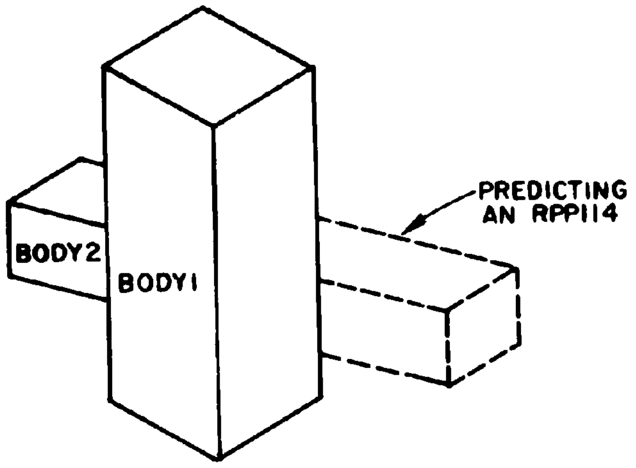
processed during phase 2 for the scene in Figure 6-8a. The other lines are known to be totally visible. Concave objects are processed individually according to the procedure of phase 2 after back lines have been removed but before the scene is considered as a whole.

VERIFICATION AND DECISION

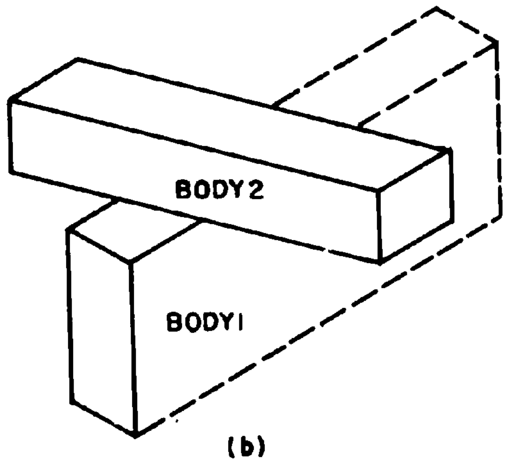
Verification is simply the process of determining how well a predicted line drawing agrees with the original input. There are two interpretations for "original input" here. First, the predicted line drawing should match the original line drawing from which it was derived. This can be confirmed by checking that for each line of the prediction there is a corresponding line in the original line drawing. If such a correspondence cannot be set up, COMPLEX concludes that either it has failed in its analysis (description) of the scene, or the original line drawing was incomplete, that is, some lines were missing. Quite often such missing lines can be detected by a statistical verification operator [40] 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 it selectively at this stage of the analysis.

COMPLEX proceeds to confirm or refute the hypothesized scene description. If all of the predicted lines can be matched to lines in the original line drawing, it assumes that the description is correct. For any predicted line which cannot be matched with a line in the original line drawing, it calls upon the statistical verification operator to check for the line (edge) in the TV data. After this has been done, if a particular body has no more than N of it predicted edges unconfirmed (N is currently set equal to 3), COMPLEX assumes that the description for this body is correct, otherwise COMPLEX assumes that the body has been recognized incorrectly. COMPLEX assumes that if a body cannot be verified, the body itself and not its occluder has been incorrectly recognized. Permitting N lines to remain unconfirmed allows for a few noisy lines in the original line drawing or some lines which simply cannot be seen due to poor lighting conditions.

Although the primary goal of verification is to confirm a hypothesis inferred from the original input data, there are occasions when there is simply insufficient data from which to generate a good hypothesis. In these cases verification performs a major function in the recognition process. Two such examples are shown in Figure 6-9. In Figure 6-9a assume that COMPLEX has recognized Body1 as an RPP114. From the little bit of Body2 which is visible assume that RECOGNIZE can determine only that the body is



(a)



(b)

Figure 6-9 . Ambiguous Scenes Resulting from Insufficient Data .

either an RPP112 or an RPP114. If COMPLEX guesses "RPP114" and makes a prediction based on this, it would find that this guess is inconsistent with the original input. The dotted section predicted to the right of Body1 does not exist. It must, therefore, be a RPP112. Humans, knowing the prototypes which make up the environment, would presumably perform a similar sort of analysis. In contrast to the situation in Figure 6-9a where the insufficient data results from occlusion, in Figure 6-9b the insufficient data arises from an incomplete line drawing. Assume that Body1 has been determined to be either an RPP122 or an RPP124. A hypothesis that the body was an RPP124 could be confirmed or rejected after prediction by checking in the TV data for the edges indicated by dotted lines.

CHAPTER 7
SYSTEM PERFORMANCE

In this chapter we examine the performance of COMPLEX by means of a number of examples. The examples have been chosen to illustrate and clarify some of the ideas presented in the previous chapters. Three of the examples are presented in detail. A number of other examples are examined more briefly, in each case to clarify a particular feature of COMPLEX. Finally, we indicate a few cases where COMPLEX is known to fail.

GENERATION OF EXAMPLES

The line drawing projections in this chapter all originated with real scenes. The scenes consisted of flat-white painted wooden blocks on a black cloth background. A TV camera viewed the scenes from a height of about 20 inches above the table top through either a 25mm or a 52 mm lens. Approximately 6 bits of intensity information were obtained by combining four separate 4 bit segments of the total dynamic range of the camera. The intensity matrix was then processed by an edge-follower program written by Manfred Hueckel [15] to prduce an ordered list of edge points (see Figure 1-3b). The more powerful accommodating edge-follower developed by Pingle and Tenenbaum [40] was not available at the time that these examples were generated.

A program written by G. Grape was used to transform the edge-point data into a line drawing. Unfortunately, this program is not yet able to deal with very complex scenes by itself. Although a few of the line drawings in this section were generated automatically, most of the interesting cases have had to be "touched up". Either spurious lines were deleted or some missing lines were added. In those cases where no addition or removal of lines was required, it was still necessary to manually adjust a set of 9 parameters which control the line-fitting process. It appears that in the near future an improved version of Grape's program will be capable of handling most scenes of interest with no human intervention.

The photographs included in this section were all taken directly from our Information International Inc. displays. The film used was Kodak TRI-X with an exposure of 1/8 second at f8.

SEVERAL EXAMPLES IN DETAIL

SCENE R3.CRN

Figure 7-1a is one of the line drawings generated without any touch up. The scene consists of an RPP122 and a CUBE in front of it. The intensity distributions were such

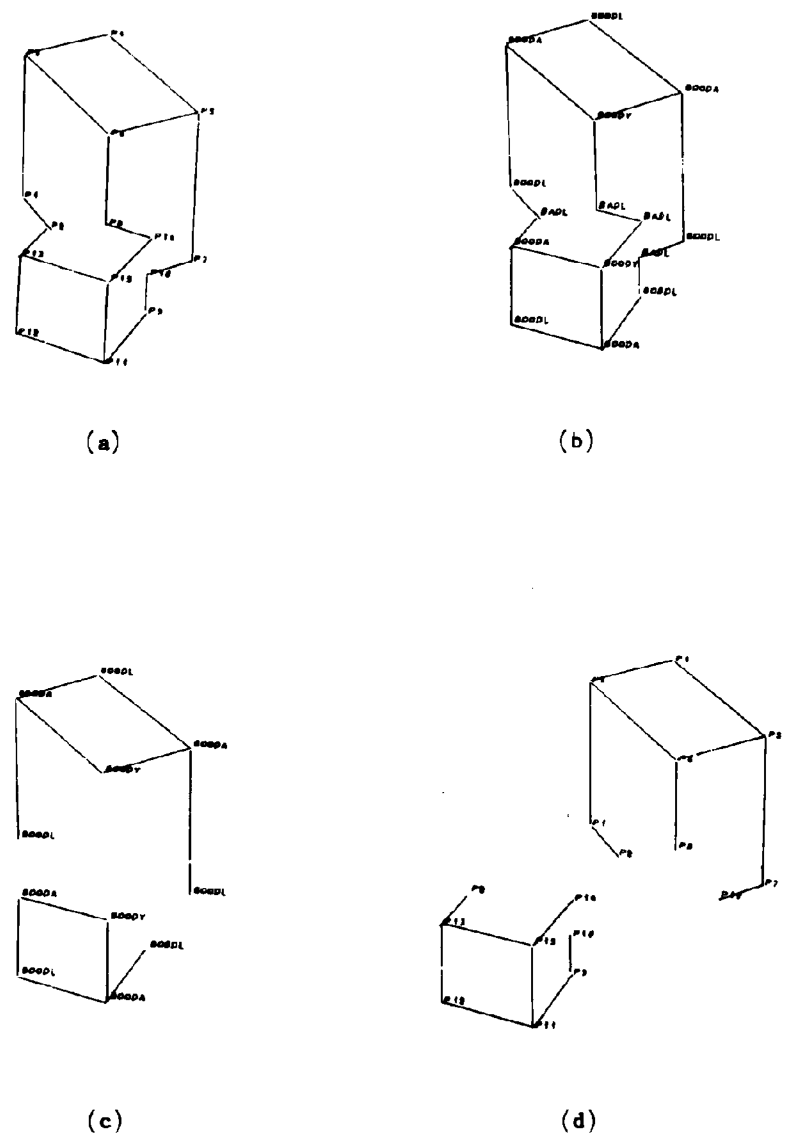
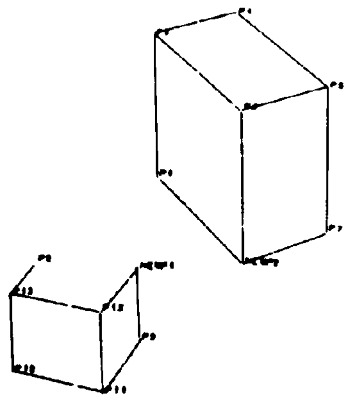
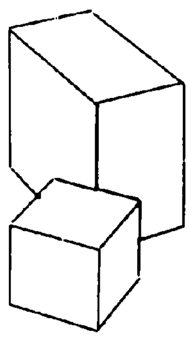


Figure 7-1 . Scene R3.CRN .

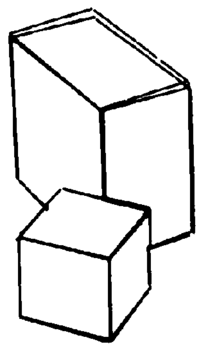
Note: GOODA = GOODARROW



(e)



(f)



(g)

Figure 7-1 . Scene R3.CRN .

that in 2 cases regions that actually correspond to 2 faces arise. From the vertex labeling in Figure 7-1b the motivation for the BADL type vertex becomes clear. BADLs generally result from T-joints where one of the collinear segments is missing (P2, P8, or P10), from an arrow vertex (P14), or from a "Y" type vertex. Although both lines meeting at point P14 belong to the same body, the lines meeting at P8 do not. There is a good chance, however, that the lines at P14 will be assigned to the proper body based on the vertices at their other ends or based on one of the special case heuristics. It is also quite probable that the bodies can be recognized even if these lines are missing from the partial body projections. Figure 7-1c shows the graph that SEGMENT sets up based on the labeling of Figure 7-1b. As mentioned earlier, neglecting the special case heuristics, the disjoint subgraphs of Figure 7-1c correspond to the individual bodies in the scene. Figure 7-1d shows these individual bodies before any processing by the line completion routines and Figure 7-1e shows them after completion. The only simple completion routine required for this scene was ADDCORNER. Note that in extending lines P14-P15 and P9-P12 ADDCORNER checks the distance of the intersection point from each of the existing object corners. If this distance is less than a prespecified tolerance, the corresponding corner is substituted for the new point.

Each block is recognized in an identical manner since both of them are supported by the table. The 3 lowest corners of each are identified as base points and are located in 3-space using the image-table collineation. Points P15 and P6 are located in 3-space by assuming that they lie on edges normal to the table since lines P11-P15 and NEWP2-P6 are vertical in the image. The projection pairs (Number of Faces, Number of Vertices) imply in each case that the corresponding prototype must be a parallelepiped. The lengths of the 3 edges connecting the 4 points mentioned above are then sufficient to specify the prototype. Although the original line drawing was incomplete, each of the resulting individual body projections is complete. This is, of course, not generally the case as the next example will show.

The line drawing of Figure 7-1f results after locating each object in space and predicting how the "hypothesized" scene would appear from the camera's point of view. The determination of the transform that properly positions the appropriate model in space proceeds exactly as described in the previous chapter based on the 4 points (forming a "peak") just mentioned. As show in Figure 7-1g, the prediction and image fit quite well except for the 3 lines missing from the original line drawing. Most probably these lines could be found by explicitly looking for them in the TV image with the sensitive verification operator. Even if they could not be detected, enough of the lines match so that COMPLEX accepts the hypothesized scene description.

SCENE P3.CRN

Figure 7-2a shows a scene slightly more complex than the previous one. The scene actually consists of 3 blocks, an LBEAM, a WEDGE122, and a RPP112. Again some of the lines separating faces of different bodies are missing due to the lighting conditions. In addition, several lines are displaced from their "ideal" positions. In particular, the T-joint at P21 has its tops bent in such a way that it is almost an arrow. SEGMENT classifies this vertex properly, as shown in Figure 7-2b, by tolerating a small deviation from collinearity (approx. 30 degrees) for the 2 tops of a T-joint. The little segment at P23 is also not in its proper position since the corner finder in the Grape program merged with P17 the T-joint where this segment originally intersected line P17-P19. The result is a MULTI (Figure 7-2b). In this particular case, it makes no difference since both lines meeting at P23 will be omitted from the individual body descriptions (see below).

Several things are worth noting in Figures 7-2b, c, and d. In Fgure 7-2c we note that two unlabeled nodes have been created based on "matching BADLs". If the one resulting from P1 and P14 had not been created, the two edges meeting

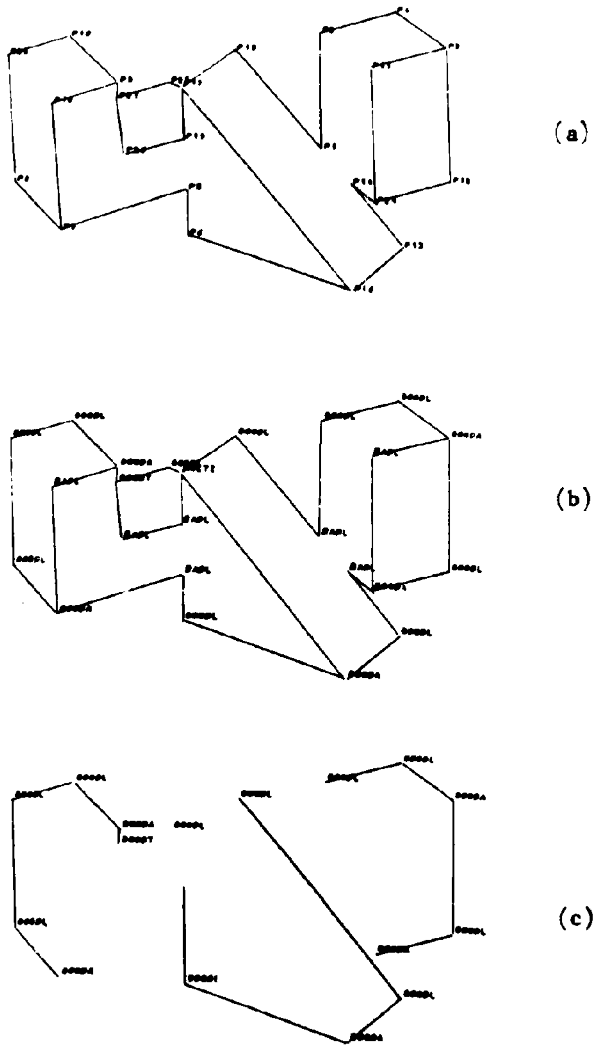
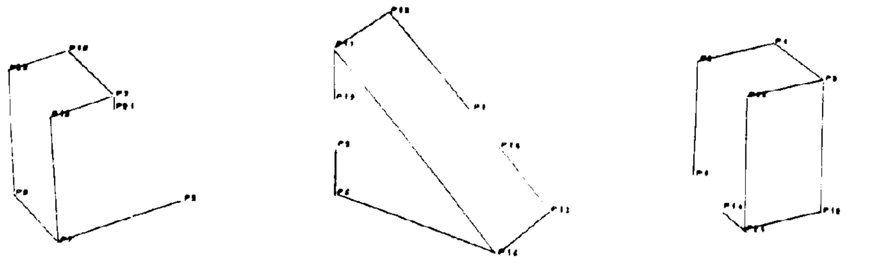
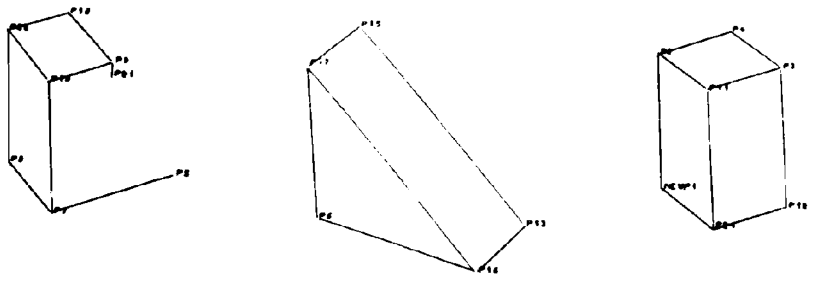


Figure 7-2 . Scene P3.CRN

Note: GOODA = GOODARROW

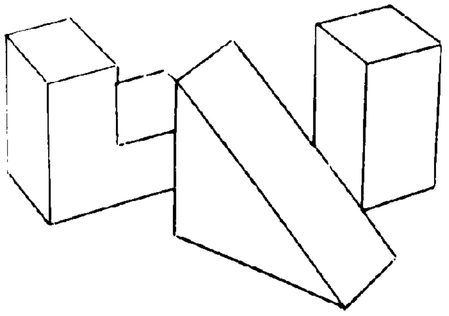


(d)

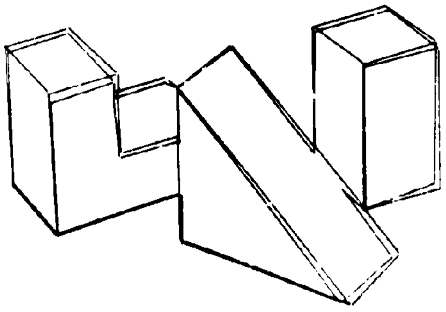


(e

Figure 7-2 . Scene P5.CRN .



(f)



(g)

Figure 7-2 . Scene F5.CRN .

at P15 would not have been identified as part of the wedge due to the MULTI at P17. The node corresponding to the GOODL at P23 is discarded since after all the merging and special case heuristics have been applied, it still has an associated set containing only 2 lines. SEGMENT correctly assumes that this node does not correspond to a separate body, and that the partial projection to which these two lines should be associated can be analyzed without them.

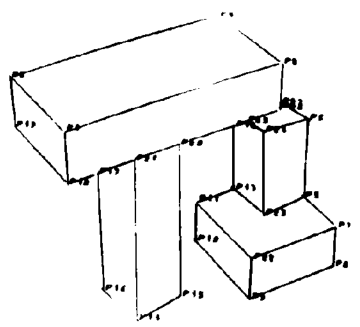
The completion routines add many of the missing lines for the individual objects. As indicated in Figures 7-2d and e, procedure JOIN modifies the WEDGE122 replacing lines P17-P19 and P5-P6 by line P17-P6 and lines P1-P15 and P13-P14 by line P13-P15. ADDCORNER also fixes up the lower left-hand corner of the RPP112. ADDLINE adds lines between P18 and P22 for the LBEAM and between P2 and P11 for the RPP112. The result of the modifications is that both the WEDGE122 and the RPP112 projections are complete. Although much of the LBEAM is missing, the visible edge lengths turn out to be sufficient for identification and positioning. Recognition is again straightforward since all three objects are determined to rest on the table.

The prediction based on the hypothesized scene description is shown in Figure 7-2f and again with the original line drawing in Figure 7-2g. The discrepancies between the predicted lines and the original ones that are apparent in this example arise from several factors. First and orobrobably most important, the original line drawing is not an ideal perspective projection of the scene but rather a perspective projection modified by the noise in the original data and the peculiarities of the preprocessors. Any transformation derived from only part of the data (a few local features) will, in general, not agree exactly with the rest of the data. Some of the errors undoubtedly arise from our simplified model of the picture taking process. Finally, the author's ability as a carpenter is subject to question. Consequently, some of the do-it-yourself blocks do not exactly match the specifications of the prototypes. Since these errors have not interfered with the operation of COMPLEX, not much effort has as yet been spent on precisely locating and removing their sources. The actual error is usually less than 0.15 inches.

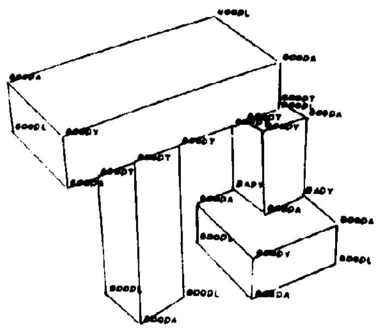
SCENE R9.CRN

In contrast to the previous two examples where all the blocks rest on the table this example, Figure 7-3a, illustrate the complications arising from other forms of support.

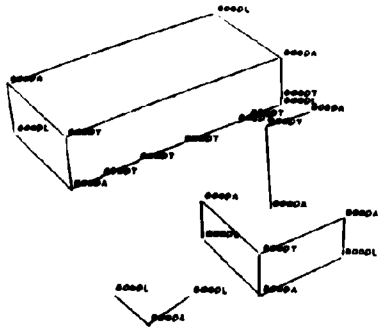
The only interesting points to be observed in the labeling of Figure 7-3b are the BADYs at P5 and P13. It is relatively clear why one wants to think of BADYs as a degenerate T-joints. If the top block at the BADY were moved



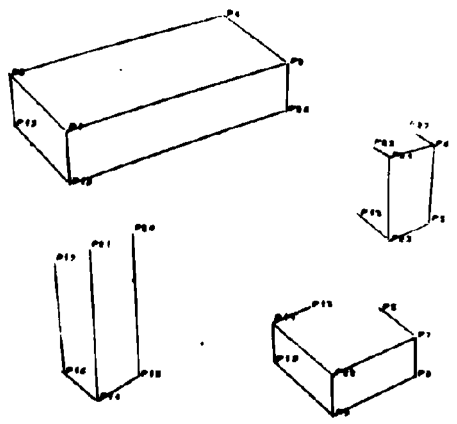
(a)



(b)



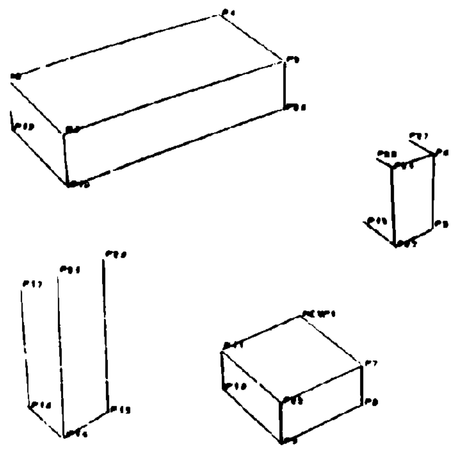
(c)



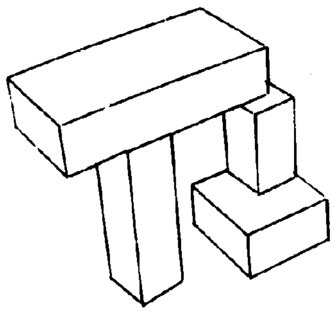
(d)

Figure 7-3 . Scene R9.CRN .

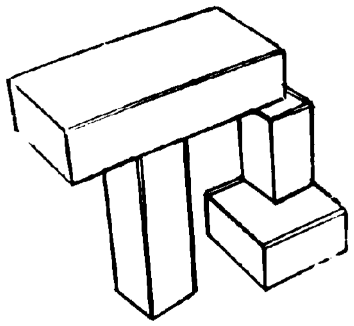
Note: GOODA = GOODARROW



(e)



(f)



(g)

Figure 7-3 . Scene [illegible].CRN .

slightly, one or both of the BADYs would become T-joints. In Figure 7-3d we can see that SEGMENT has omitted the line between P12 and P13. It is not associated with any node of the graph in Figure 7-3c, and none of the special case heuristics cause it to be assigned to a body. This presents no problem, however, since enough of the RPP112 is visible without it. For simplicity we shall refer to bodies by their proper names even though they have not yet been identified by the program.

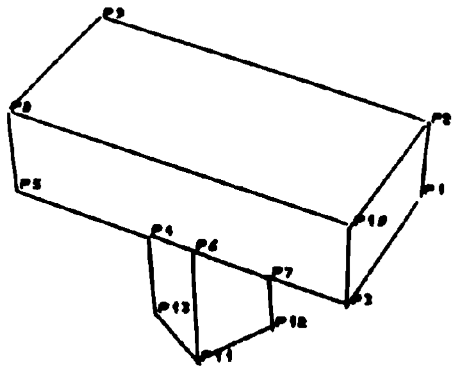
Recognition, as indicated earlier, proceeds in a bottom-to-top fashion. To begin with COMPLEX might decide to analyze either the RPP114 or the RPP122 which are both resting on the table. Assume that it examines the RPP114 first. COMPLEX then tries to recognize the RPP124 on top of the RPP114 but fails since not all the potential supporters of the RPP124 have yet been recognized. The next body processed, therefore, must be the RPP122 resting on the table. After RECOGNIZE identifies and locates it in 3-space, the plane of its top face can be used as the support plane for the RPP112. RECOGNIZE can then identify and locate this body. Now knowing the positions of the RPP114, the RPP122, and the RPP112, COMPLEX can determine that the true support for the RPP124 is provided by the RPP114. This last body is then identified and located in space.

The prediction is shown in Figure 7-3f and with the original input superimposed in Figure 7-3g. In this case the two line drawings match quite well.

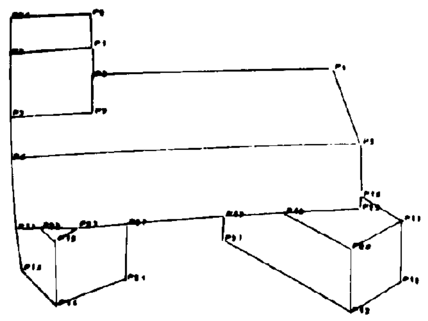
COMMENTS ON OTHER INTERESTING EXAMPLES

The scene of Figure 7-4a, VIEW3.CRN, illustrates how the constraint of known model size can be used in recognition. From the visible portion of the bottom block RECOGNIZE can only determine that it is either a RPP112 or a RPP114 (it is actually an RPP112). COMPLEX proceeds by hypothesizing that it is a RPP114. This allows the support plane of the top object to be determined and its base edge lengths to be computed. These lengths, however, do not correspond to any of the lengths of edges of models which topologically match the top object. Such a failure by RECOGNIZE signals COMPLEX that it should try the other possible prototype, the RPP114, as the identity of the supporting object. With this hypothesis the analysis, prediction and verification proceed without difficulty.

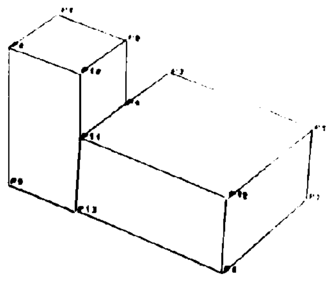
Scene 05.CRN shown in Figure 7-4b is a slightly more complex scene illustrating one of the unfortunate situations that arise in practice. While in theory degenerate views occur only from single points of view, the difficulty that the preprocessors have in detecting extremely narrow regions causes degenerate line drawings of objects to be produced



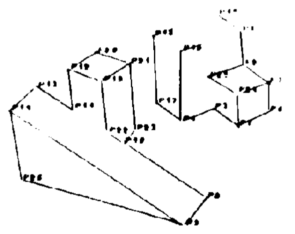
(a)



(b)

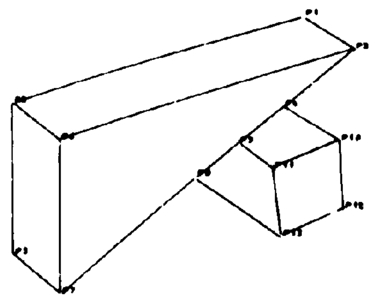


(c)

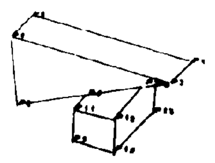


(d)

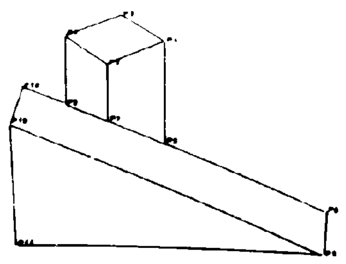
Figure 7-4 . Other Interesting Examples



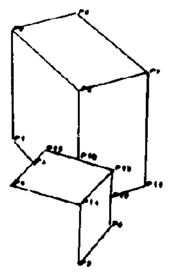
(e)



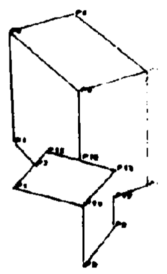
(f)



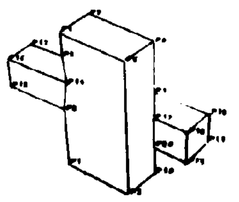
(g)



(h)



(i)



(j)

Figure 7-4 . Other Interesting Examples .

over a relatively large range of viewing angles (perhaps 20 degrees or more). This means that Q5.CRN is not atypical. The necessity of interpreting the T-joints at P1, P8, and P6 as "degenerate arrows" while interpreting those at P7 and P17 as BADTs motivates one of the special case heuristics mentioned earlier in the description of our segmentation procedure. During the prediction phase one must be careful not to predict a face that would ideally be visible but is invisible due to a "pseudo degeneracy" (i.e. the face determined by P6-P7-P17). This is accomplished simply by interpreting as visible only those faces of an object whose normal dotted into the line of sight in less than some small negative value rather than zero.

Figure 7-4c illustrates on of the problems that arises in the analysis of scenes where objects abut. The line drawing of Q1.CRN was generated without any touch up. The identification and location of the objects proceed without difficulty. The problem is that a small error in the predicted position of either of the objects drastically changes the topology of the predicted line drawing. For this example, the two objects are predicted to appear separated by about 1/10 of an inch. Although the error does not cause enough lines to be mismatched so that the correct hypothesis is rejected in this case, one might consider "forcing abutment" during recognition if such a relationship was detected during structural analysis. Prime indicators of abutment are the BADY, KJOINT, and X-joint type vertices. Winston considers the determination of this structural relationship further in [43].

Figure 7-4d, CL2.CRN, illustrates another problem caused by abutment. Each object in the scene is recognized quite simply after segmentation and partial completion. The problem arises from the fact that RECOGNIZE as it currently exists has no concept of the volume of space occupied by an object. Consequently, a small error in the calculated position of the cube can cause edges P3-P25 and P2-P25 to be physically located inside (behind the front face of) the RPP122. The resulting prediction will have these two edges missing. Ideally, one should detect body intersections and correct them by modifying the transforms associated with the offending bodies. For expediency, however, we have simply modified the criteria for determing if a line segment is visible. HIDDEN says that a line surrounded by one or more faces is visible if its midpoint is in front of or within a specified small distance behind all surrounding faces.

CL2.CRN also illustrates another problem that arises. For the RPP112 partially occluded by the wedge, only a small segment, P13-P23, of one of its base edges in visible. This edge plus edge P19-P22 are extended by ADDCORNER to form a new corner at their intersection. Since the computed location of the RPP112 depends quite directly on the location of this new corner, any error in the input data corresponding to P10-P23 may lead to relatively large errors between the predicted and original line drawings. It may at times become necessary to get a close-up view of the two edges before a predicted corner location is estimated. In this example, it was necessary to relax the matching tolerances before the object could be identified as an RPP112.

Q4.CRN shown in Figure 7-4e is the first of two "leaning" examples. The line drawing here was again produced without touch-up. It turns out that for this particular leaning example, both objects get recognized correctly without resorting to stereo ranging. The explanation is interesting. Only edge P5-P7 is identified as a base edge of the wedge; edge P3-P7 is ruled out because of the downward pointing arrow vertex at P3. Both bodies are then assumed (incorrectly) to be resting on the table. The verticality of edge P6-P7 is used to locate corner P6 in 3-space. Assuming the top face of the wedge to be horizontal (which turned out to be true), RECOGNIZE then determines the location of corner P3. Corners P5, P7, P6, and P3 are sufficient to specify the "top object" as a WEDGE124 and allow it to be positioned properly in space. From the small portion of the bottom object which is visible, it is only possible to determine that it is either an RPP112 or an RPP114. Assuming that it is an RPP114, PREDICT generates a line drawing with part of the RPR114 sticking out behind the WEDGE124. Since this prediction does not match the original line drawing, the description set up for the bottom body is deleted and it is re-identified as an RPP112. This time the prediction matches the original line drawing and the hypothesized description is accepted.

In example LEAN.CRN of Figure 7-4f we have a slightly different situation of one object leaning on another. For the wedge, edges P2-P3 and P3-P4 are identified as base edges, and consequently the wedge is assumed to be supported on the top face of the lower object. Based on this assumption, however, the wedge cannot be identified with any prototype because the predicted base edge lengths do not match any adjacent pair of model edges. COMPLEX currently exits with a "Failure in Recognition" message typed out when this situation occurs. When the nearly complete stereo-focus depth package becomes available, COMPLEX will check such situations for base points with different z coordinates to detect leaning configurations.

Scene R2.CRN shown in Figure 7-4g is like the previous example in the sense that although COMPLEX does not exactly fail to describe the scene, it does not completely succeed either. In this example none of the base edges of the occluded RPP112 are visible. Since the location of the base of an object plays such as important role in our one-eyed reconstruction scheme, COMPLEX currently detects such cases and indicates that either the scene should be rotated on the lazy susan or the occluding object(s) should be physically moved. Using stereo or focus ranging, however, one would not have to give up in these situations.

The scene T1.CRN shown in Figure 7-4h is the final example we shall mention in this section. The interesting feature of this example is the fact that the two lines that are missing are part of the exterior boundary. As described previously, COMPLEX was designed under the assumption that exterior lines would usually be present. Although segmentation will often fail if this is not the case, T1.CRN is analyzed successfully.

COMMENTS ON SOME SYSTEM FAILURES

In this section we consider situations where COMPLEX fails. These failures occur because of insufficient data or bad heuristics. Such heuristics rule out the correct answer in the process of limiting the search space. Clearly, if the input line drawing is complete and COMPLEX cannot identify an object correctly, we would say that the program has failed. If the line drawing is incomplete, however, it is not clear whether to assign the cause of failure to COMPLEX or to the poor input.

T2.CRN shown in Figure 7-4i is identical to T1.CRN of Figure 7-4h except for a single missing internal edge. A problem arises in Figure 7-4i because the "Y" vertex of the front object gets labled BADY because none of its lines connect to arrow vertices. This causes the front object to be identified as two separate bodies. Although we could have modified the heuristics of SEGMENT to handle this case, we have chosen not to do so. We shall consider this a situation where our local heuristics are inadequate for segmentation. The responsibility for this failure, system or input, is left for the reader to assign.

The scene of Figure 7-4j is an example of a situation that is possible although unlikely. The two hidden bodies are interpreted as halves of the same body and are merged by the segmenter. Since this long body matches none of the protypes, a recognition failure occurs. In a future reincarnation COMPLEX would presumably have the ability to unmake incorrect merges such as this.

CHAPTER 8

CONCLUSION

RESULTS OF THIS THESIS

The principal result of our research is a heuristic program, COMPLEX, capable of interpreting line drawings as a three-dimensional scene. The only previous three-dimensional scene description system comparable to COMPLEX 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. COMPLEX is able to deal with considerably more complex scenes than was the Roberts system. COMPLEX allows objects to be supported by one another as well as by the table. The most distinctive feature of COMPLEX, however, is its ability to interpret a scene based on imperfect data. In addition to tolerating inaccuracies in its input, COMPLEX can analyze degenerate 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 of the program. COMPLEX uses its line drawing input and a known set of models to suggest and test hypotheses. We believe that this approach to machine perception will prove to be a fruitful one.

We have also presented some preliminary results concerning the constraints in projections of planar-faced solids. These results begin to answer the question "How much does a single view of an object imply about the shape and position of the object viewed"? More importantly, however, they indicate the power of external constraints in the interpretation of inherently ambiguous line drawing data.

SUGGESTIONS FOR FUTURE WORK

The ideas which we presented in Chapter 3 only begin to answer the question of how prior knowledge or environmental constraints can be utilized to aid in the interpretation of two-dimensional data. To be more specific, it would be interesting to characterize the class of trihedral projections for which 4 points completely specify the visible object. One might also wish to look at classes of objects (not necessarily trihedral) and projections that are specified by N independent points. It would be worthwhile to consider in more detail the possibility of guessing (adding) lines or line segments to the projection as mentioned previously. Finally, constraints other than the point plane incidence constraints should be investigated in considerably more detail.

In the area of designing a more reliable vision system, we have already indicated the need for an organization which integrates better the advantages of both the model driven and data driven approaches. Portions of COMPLEX are still too data-driven to be effective in analyzing very noisy scenes. We also need to come to a clearer understanding of the most effective way to utilize both monocular and binouclar depth cues. Many new problems worthy of investigation will undoubtedly become apparent as more experience is accumulated concerning the complete Hand-Eye system.

While probably not effecting the reliability of the vision system, it would be desirable to be able to "learn" new structural descriptions of models and/or decision mechanisms for distinguishing between them. Both of these things are assumed to be known a priori in our current implementation.

In the anaysis of extremely complex scenes it will undoubtedly be the case that more than a single view is reauired for complete scene description. People, when confronted with a fairly complex scene, walk around and analyze the scene from several different points of view. As far as we know, no one has as yet considered the problem of efficiently analyzing a second (wide angle) view of a scene based on a partial analysis of a previous view.

Finally, we must keep in mind that the problem which we have been considering is to produce a complete scene description of a given line drawing. If only a partial interpretation of the line drawing were required, the analysis would undoubtedly be quite different. For example, if the question were "Are there any wedges in the scene?", one might be able to do better than analyzing the entire scene and then checking if any wedges have been identified. Task dependent partial interpretation of pictures appears to be another area worthy of further study.

APPENDIX I
A SIMPLE CAMERA MODEL

Image-Table Top Coordinates:

Our approach is to find a mapping from the image coordinates to the plane of the table top. The result of this is that any point in the image that corresponds to to a real world point actually lying on the table top (i.e. P1' in Figure 3-1) will be mapped into its correct 3-space location (i.e. P1) in the table plane. Those points that do not correspond to points on the table (i.e. P2') will be mapped into the point on the table intercepted by a ray passing through the camera center (also called the lens center or center of projection) and the point in the image (i.e. P2T).

Without getting engrossed in the details of the projective geometry involved, let us say a little more about this mapping. We see from Figure 3-1 that what we have is simply a projective transformation of one 2 dimensional space into another with the camera center as the center of projection. If we represent points in each plane using 3-dimensional homogeneous coordinates, then we can represent the transformation from the image system to the table top system by a single 3x3 matrix A such that:

$$V = A V'$$

where V' =(image x' coordinate, image y' coordinate, 1) and V =(wx,wy ,w). By the Fundamental Theorem of Projective Geometry there exists a unique transformation mapping 4 points in one plane into 4 points in another plane providing that no 3 of the original points are collinear, thus 4 points are sufficient to specify T.

Prior to our analysis of a scene, these points are laid out on the table and A is determined. Consequently, for any point in the image we can compute its "corresponding" point on the table.

The Lens Center and How To Find It!

We can determine C(Xc,Yc,Zc), the lens center location, fairly simply by the arrangement shown in Figure A-1. Here we assume that we know the locations of calibration points P0 and P1 in the table (X,Y,Z) system. Knowing A and where P0 and P1 appear in the image (not shown), we can determine P0T and P1T respectively. Knowing 2 points along each of these rays we know the rays and can consequently determine C, their intersection. A problem arises in practice from the fact that due to measurement errors for the 4 points P0,P1,P0T,and P1T, the two rays do not intersect but merely come close. The way that we shall

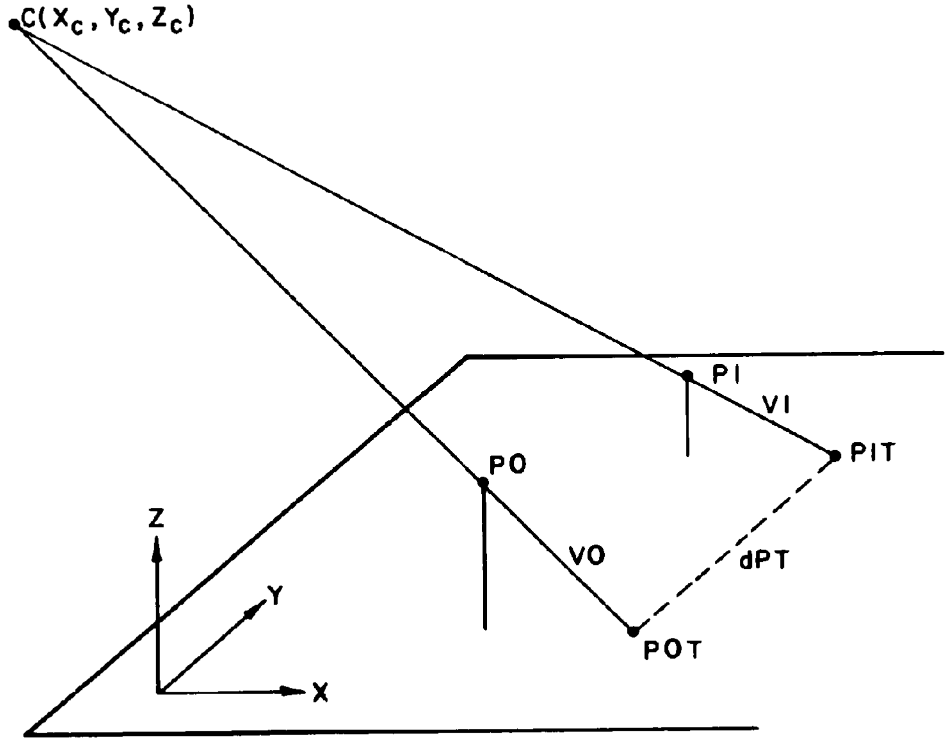


Figure A-1 . Arrangement for Determining the Lens Center proceed is to assume that the rays do not intersect and determine that point in 3-space where the perpendicular distance between them is a minimum. We shall call this point the "best intersection" of two (possibly) skew rays.

We define L0 to be the ray from C through P0 and correspondingly L1 to be the ray from C through P1. We represent L0 and L1 parametrically as:

$$L0(t) = P0T + (P0-P0T)t$$

$$L1(t') = P1T + (P1-P1T)t'$$

where if $0 \le t \le 1$ then the corresponding point is between PJT and PJ. (Note: We are dealing with vector equations here.) Let us define:

$$V0 = P0-P0T$$

$$V1 = P1-P1T$$

$$dPT = P1T-P0T.$$

We want to find values t=t0 and t'=t0' such that $D(t,t') = [L0(t)-L1(t')]^2$ is minimized. Setting $\partial D/\partial t=0$ and $\partial D/\partial t'=0$ yields the following set of equations to be solved for t0 and t0':

$$\begin{bmatrix} |V0|^2 & -V0 \cdot V1 \\ -V0 \cdot V1 & |V1|^2 \end{bmatrix} \begin{bmatrix} t0 \\ t0' \end{bmatrix} = \begin{bmatrix} V0 \cdot dPT \\ -V1 \cdot dPT \end{bmatrix}$$

We will then take $P_{best} = 1/2[L0(t0)+L1(t0')]$. This procedure gives a good estimate of the camera location; the size of D gives a measure of the error. If better accuracy is needed one could do several such calculations of C and then take an average. The question of determining an accurate and consistent model of the camera system constitutes a large portion of a recent dissertation by Sobel[37].

BIBLIOGRAPHY

1- Ahuja, D.V. and Coons,S.A. [1968] Geometry for construction and display. IBM Systems Journal, Vol. 7, Nos. 3 and 4, pp. 188-206.

2- Binford, T. [1970] A visual preprocessor. Internal Report, Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts (April).

3- Duda, R. O. and Hart, P. [1969] Perspective transformations. SRI Project No. 7494, Artificial Intelligence Group Technical Note No. 3, Stanford Research Institute, Menlo Park, California (February).

4- Duda, R. O. and Hart, P. [1971] Forthcoming book on pattern recognition and picture processing.

5- Feigenbaum, E.A. and Feldman, J. (Ed.) [1963] Computers and Thought. McGraw-Hill, New York.

6- Feldman, J.A., Feldman, G.M., Falk, G., Grape, G., pearlman, J., Sobel, I., Tenenbaum, J.M. [1969] The Stanford hand-eye project. Proc. International Joint Conference on Artificial Intelligence (May), pp. 521-526.

7- Fu, K. S. and Swain, P. H. [1970] Nonparametric and linguistic approaches to pattern recognition. LARS Information Note 051970, TR-EE 70-20, Purdue University, Lafayette, Indiana, (June).

8- Grape, G.R [1969] Computer vision through sequential abstractions. Internal Memo, Stanford Artificial Intelligence Project, (June).

9- Grape, G. R. [1970] 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, California (March).

10- Gray, J.C. [1967] Compound data structure for computer aided design. Proc. 22nd National Conference ACM, P-67,Thompson Book Co., Washington , D.C., pp. 355-365.

11- Gregory, R.L. [1966] Eye and Brain Mc-Graw Hill , New York

12- Guzman, A, Some aspects of pattern recognition by computer. MAC-TR-37 (THESIS), Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts, (February).

13- Guzmán,A. [1968] Computer recognition of three-dimensional objects in a visual scene. MAC-TR-59 (THESIS), Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts, (December).

14- Hart, P, [1969] Stereographic perception of 3-dimensional scenes. Final Report, SRI Project 7642, Stanford Research Institute, Menlo Park, California, (August).

15- Hueckel, M. [1969] An operator which locates edges in digitized pictures, Artificial Intelligence Memo No. 105, Stanford University, Stanford, Calif. (October).

16- Ho, Y. and Agrawala, A.K. [1968] On pattern classification algorithms, introduction and survey, Proc. IEEE, 56, (December), pp. 2101-2114.

17- Hochberg, J. [1964] Perception. Prentice-Hall, New Jersey.

18- Huffman, D. A. [1969] Logical analysis of pictures of polyhedra. Artificial Intelligence Group, Technical Note No. 6, SRI Project 7494, Stanford Research Institute, Menlo Park, California (May).

1[illegible] Kelly, M.D. [1970] Edge detection in pictures by computer using planning. Artificial Intelligence Memo No. 108, Stanford University, Stanford, California, (January).

20- Luccio, F. and Vigna, P. D. [1969] Some aspects of the recognition of convex polyhedra from two plane projections. Relazione interna n. 69-21, Instituto di Elettrotecnica ed Elettronica del Politecnico di Milano (September).

21- McCarthy, J., Earnest, L.D., Reddy, D.R., and Vicens, P.J. [1968] A computer with hands, eyes, and ears. Proc AFIPS 1968 Fall Joint Computer Conference, Vol. 33, Thompson Book Co., Washington, D.C., pp. 329-338.

22- McCarthy,J. et. al. [1970] Project technical report. Stanford Artifical Intelligence Project Memo AIM-117, Stanford University, Stanford, California (April).

23- Miller, W.F. and Shaw, A.C. [1968] Linguistic methods in picture processing: a survey. Proc. AFIPS 1968 Fall Joint Computer Conference, Vol. 33, Thompson Book Co., Washington, D.C., pp. 279-290.

24- MIT Project MAC. [1969] Progress report VI. Massachusetts Institute of Technology. Cambridge, Massachusetts (July).

25- Moorer, A. [1969] Stanford A-I Project monitor manual. Artificial Intelligence Operating Notes Nos. 54 and 55, Stanford University, Stanford, California, (September).

26- Nagy, G. [1968] State of the art in pattern recognition. Proc. IEEE. 56, (May), pp. 836-862.

27- Narasimhan, R. [1966] Syntax-directed interpretation of classes of pictures. Communications of the ACM, 9, (March), pp. 166-173.

28- Nilsson, N.J. [1969] A mobile automaton: an application of artificial intelligence techniques. Proc. International Joint Conference on Artificial Intelligence, (May), pp. 509-515.

29- Orban, R. [1970] Removing shadows in a scene. Project MAC, Artifical Intelligence Memo No. 192, Massachusetts Institute of Technology, Cambridge, Massachusetts (April).

30- Paul, R., Falk, G., and Feldman, J.A. [1969] The computer representation of simply described scenes. Artificial Intelligence Memo No. 101. Stanford University, Stanford, California, (October).

31- Reddy, D. R. [1966] An approach to computer speech recognition by direct anaysis of the speech wave. Technical Report No. CS49, Artificial Intelligence Memo No. 43, Stanford University, Stanford, California (September).

32- Reddy, D.R. [1969] On the use of environmental, syntactic and probabilistic constraints in vision and speech. Artificial Intelligence Memo No. 78, Stanford University, Stanford, California (January).

33- Roberts, L.G. [1963] Machine perception of three-dimensional solids. Technical Report No. 315, Lincoln Laboratory, M.I.T. (May). Also in Optical and Electro-Optical Information Processing, Tippett, J.T. et al (Ed.), MIT Press, Cambridge, Mass., 1965, pp. 159-197.

34- Rommey, G. W. [1969] Computer Assisted Assembly and Rendering of Solids. Rome Air Development Center Techical Report 69-365, University of Utah, (September).

35- Rosenfeld, A. [1969a] Picture Processing by Computer. Academic Press, New York.

36- Shaw, A.C. [1968] The formal description and parsing of pictures. PhD Thesis, Report CS 94, Computer Science Department, Stanford Univ., Stanford, Calif.

37- Sobel, I. [1970] Camera models and machine perception. PhD Thesis, Artificial Intelligence Memo No. 121, Stanford University, Stanford, California (May).

38- Sproull, R. F. [1970] System manual for hand-eye hackers. Internal Report, Stanford Artificial Intelligence Project, Stanford University, Stanford, California (June).

39- Sproull, R. F. and Swinehart, D. [1970] SAIL. Stanford Artificial Intelligence Project Operating Note No. 57.1, Stanford University, Stanford, California (April).

40- Tenenbaum, J.M. [1970] Accommodation in computer vision. Forthcoming Ph.D. Thesis, Electrical Engineering Dept., Stanford University.

41- Vincens, P. [1969] Aspects of speech recognition by computer. PhD Thesis, Technical Report No. CS127, Artificial Intelligence Memo No. 85, Stanford University Stanford, California (April)

42- Warnock, J. E [1968] A hidden line algorithm for halftone picture representation. University of Utah Computer Science Technical Report 4-15, (May).

43- Winston, P. H. [1970] Learning structural descriptions from examples. PhD Thesis, Massachusetts Institute of Technology, Cambridge, Massacusetts (January).