(14) STAN-CS-77-591,
AIM-296

Stanford Artificial Intelligence Laboratory
Memo AIM-295

Computer Science Department
Report No. STAN-CS-77-591

(11) December 1976

(1)

(12) 246 p.

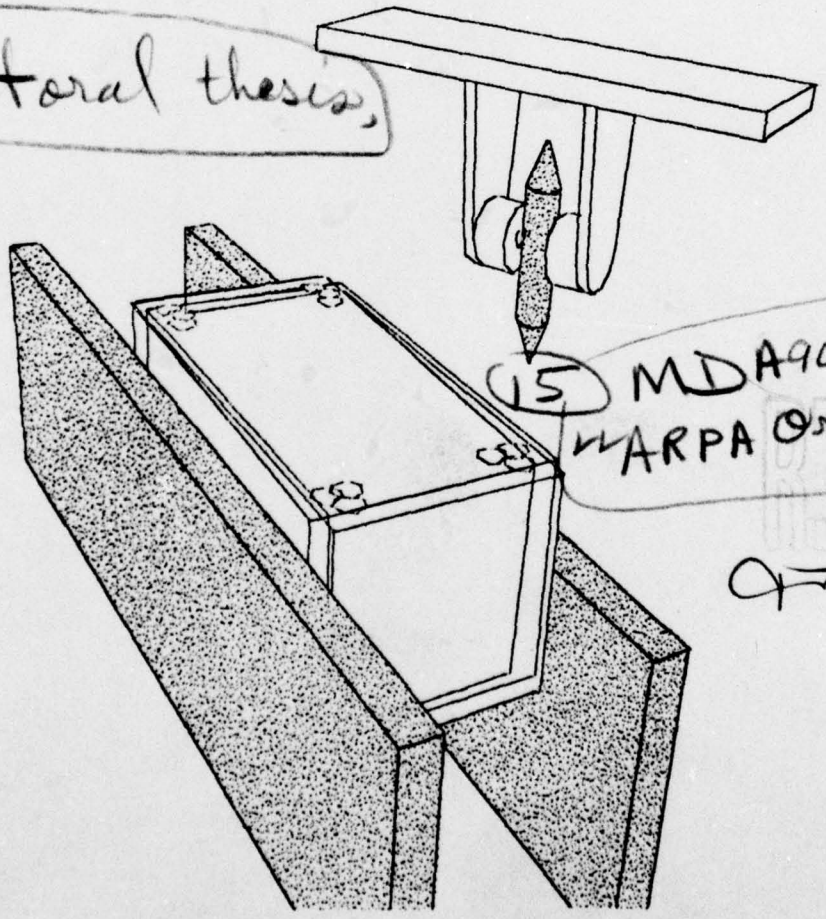
(6)

VERIFICATION VISION WITHIN A PROGRAMMABLE ASSEMBLY SYSTEM

by

(10) Robert C/Bolles

(9) Doctoral thesis,



(15) MDA903-76-C-0206,
ARPA Order-2494

COMPUTER SCIENCE DEPARTMENT
Stanford University

094120

Stanford Artificial Intelligence Laboratory
Memo AIM-295

December 1976

Computer Science Department
Report No. STAN-CS-77-591

VERIFICATION VISION WITHIN A PROGRAMMABLE ASSEMBLY SYSTEM

by

Robert C. Bolles

ABSTRACT

The research reported in this thesis concentrates on a subclass of visual information processing referred to as *verification vision* (abbreviated VV). VV uses a model of a scene to locate objects of interest in a picture of the scene. The characteristics that distinguish VV from the other types of visual information processing are: (1) the system has a great deal of prior knowledge about the type, placement, and appearance of the objects that form the scene and (2) the goal is to verify and refine the location of one or more objects in the scene. VV includes a significant portion of the visual feedback tasks required within programmable assembly. For example, locating a screw hole and determining the relative displacement between a screw and the screw hole are both VV tasks.

There are several types of information available in VV tasks that can facilitate the solution of such tasks: a model of each object in the scene, a set of initial constraints on the locations of the objects, and a set of previous pictures of this scene or similar scenes. Additional information can be obtained by applying visual operators to a current picture of the scene. [continued next page]

This research was supported by the Hertz Foundation, National Science Foundation under Contract NSF APR 74-01390-A04, and Advanced Research Projects Agency of the Department of Defense under ARPA Order No. 2494, Contract MDA903-76-C-0206. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Stanford University or any agency of the U. S. Government.

Reproduced in the U.S.A. Available from the National Technical Information Service, Springfield, Virginia 22161.

How can all of this information be used to minimize the amount of work required to perform a task? Two steps are involved in answering this question: (1) formalize the types of tasks, available information, and quantities of interest and (2) formulate combination rules that use the available information to estimate the quantities of interest. The combination rules that estimate confidences are based upon Bayes' theorem. They are designed to combine the results of operators that are not completely reliable, i.e., operators that may find any one of several known features or a *surprise*. The combination rules that estimate precisions are based upon a least-squares technique. They use the expected precisions of the operators to check the structural consistency of a set of matches and to estimate the resulting precisions of the points of interest.

An interactive VV system based upon these ideas has been implemented. It helps the programmer select potentially useful operator/feature pairs, provides a training session to gather statistics on the behavior of the operators, automatically ranks the operator/feature pairs according to their expected contributions, and performs the desired task. The VV system has also been interfaced to the AL control system for the mechanical arms and has been tested on tasks that involve a combination of touch, force, and visual feedback.

This thesis was 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.

PREFACE

The work reported in this thesis was performed while the author was a member of the hand/eye group at the Stanford Artificial Intelligence Project. The group has traditionally followed the philosophy that one of the most important stages in testing a theory is to build the necessary hardware and software and make sure that the combination performs as desired. In line with that philosophy a verification vision (abbreviated VV) system has been implemented that incorporates most of the ideas discussed in this thesis. It has been used to perform several inspection and location tasks. It has also been interfaced to the mechanical arm and the combination has been used to perform a few tasks that involve both manipulation and visual feedback.

The current implementation suffers from a well-known disease of programming: unplanned growth. Because of that, a few of the ideas, such as the heuristic to determine conservative distributions, have been implemented as separate programs. All of the examples used in the thesis are results from the VV system or one of these auxillary programs. Appendices IV and V are extensive traces of the interaction between a programmer and the VV system as the programmer prepares VV programs to perform different tasks. Comments have been added to these traces and a few minor changes have been made to clarify the explanations.

The thesis is a combination of theory and practice. Chapters three, four, and five present the theory; the rest of the chapters present the practical motivation and system aspects. The casual reader can pick up the main ideas by reading the introduction (chapter 1), the motivation chapter (chapter 2), the two traces (appendices IV and V), and the conclusion (chapter 7). A more serious reader may also want to look at the traces and conclusion before starting chapters three, four, and five.

ACKNOWLEDGEMENTS

I feel overwhelmed by the number of people who have directly helped me as I have worked on this thesis. I want to thank them all.

First, I would like to thank my advisor, Jerry Feldman, for his continual enthusiasm, encouragement, and advice. His enthusiasm helped start the hand/eye project, without which there would not have been the intellectual environment and facilities that have been so important to this work. Other members of the hand/eye group whom I have known and worked with include: Lou Paul, Tom Binford, Vic Scheinman, Bruce Baumgart, Lynn Quam, Marsha Jo Hannah, Ram Nevatia, Randy Davis, Karl Pingle, Arthur Thomas, Bruce Anderson, Eiichi Miyamoto, Charlie Wilson, Joe Zingheim, Russ Taylor, Ray Finkel, Bruce Shimano, Don Gennery, Hans Moravec, Mike Roderick, Ron Goldman, Dave Grossman, and Shahid Mujtaba. There are also several past members of the group whom I've never met, but whose code I've used. Thank you all for making it possible and exciting to work in the field of robotics.

I would especially like to thank Lou Paul for his insights and understanding. His attitude and the attitude of the other people at the A.I. Lab meant a great deal to me throughout my stay at Stanford.

I would like to thank Tom Binford for his guidance and his many careful readings of this manuscript as it was being written.

I would like to thank Dave Barstow for his friendship that has included moral and emotional support in addition to physical and intellectual exercise. Thank you.

I would like to thank my family: my wife, Jomary; my mother, father, brother, and sister; for their encouragment and support at the many decision points along the way.

Finally, I would to thank the Hertz Foundation, the National Science Foundation, and the Advanced Research Projects Agency for their financial support.

TABLE OF CONTENTS

Chapter 1. INTRODUCTION 1

Chapter 2. MOTIVATION 12
Section 1. CHECKING FOR A SCREW 12
Section 2. LOCATING A HOLE 21
Section 3. SUMMARY OF REQUIRED FACILITIES 24

Chapter 3. EXECUTION-TIME COMBINATION RULES FOR INSPECTION 30
Section 1. OPERATOR VALUE INFORMATION 30
Section 2. KNOWN ALTERNATIVES FOR A FEATURE 44
Section 3. SURPRISES 51
Section 4. MULTIPLE-VALUED OPERATORS 56
Section 5. POSITION INFORMATION 58
Section 6. ASSUMPTIONS 70

Chapter 4. EXECUTION-TIME COMBINATION RULES FOR LOCATION 81
Section 1. DETERMINING PRECISION 82
Section 2. CONFIDENCE IN THE PRECISION 87

Chapter 5. PLANNING-TIME COMBINATION RULES 93
Section 1. RANKING FEATURES BY VALUE 94
Section 2. KNOWN ALTERNATIVES AND SURPRISES 96
Section 3. COST INFORMATION 99
Section 4. LEAST-SQUARES CULLING 100
Section 5. INSPECTION 104
Section 6. PRECISION 106
Section 7. CONFIDENCE IN THE PRECISION 108
Section 8. EXPECTED COST 108

Chapter 6. PROGRAMMING TIME AND TRAINING TIME 110
Section 1. STATE THE TASK 112
Section 2. POSITION AND CALIBRATE THE CAMERA 116
Section 3. CHOOSE POTENTIAL OPERATOR/FEATURE PAIRS 126
Section 4. GATHER STATISTICS 147

Chapter 7. CONCLUSIONS AND EXTENSIONS 156
Section 1. FRAMEWORK FOR VV 156
Section 2. EASE OF PROGRAMMING 158

BIBLIOGRAPHY 161

APPENDIX I. ASSIGNMENTS FOR TWO OPERATORS 172

APPENDIX II. BEST KNOWN ALTERNATIVES 176

APPENDIX III. EXPECTED LOG-LIKELIHOOD RATIO FORMULA. 179

APPENDIX IV. SCREW-INSPECTION EXAMPLE 185
Section 1. TRACE OF A SCREW-INSPECTION TASK 186
Section 2. TASK FILE FOR THE SCREW-INSPECTION TASK 200
Section 3. CALIBRATION WITH RESPECT TO THE SCREW DISPENSER . . 202
Section 4. INTERFACE TO THE MECHANICAL ARM 205

APPENDIX V. HOLE LOCATION EXAMPLE 210

APPENDIX VI. CURVES AS FEATURES 224
Section 1. BASIC APPROACH 225
Section 2. LOCAL TESTS FOR A POINT ON A CURVE 229
Section 3. SPECIFICATION OF A CURVE 230
Section 4. REPRESENTATION OF A CURVE 235

CHAPTER 1

INTRODUCTION

Verification vision is a type of visual information processing that uses a model of a scene to locate objects of interest in a picture of the scene. The characteristics that distinguish verification vision (abbreviated VV) from the other types of visual information processing are: (1) the system has a great deal of prior knowledge about the type, placement, and appearance of the objects that form the scene and (2) the goal is to verify and refine the location of one or more objects in the scene. The following situation illustrates a typical use of VV:

> During the assembly of a pump, a mechanical arm places the pump base in a vise. The next step is to insert an aligning pin into one of the screw holes in the base. But the location of the screw hole is not known precisely enough to insert the pin directly. The programmable assembly program needs to improve its estimate for the location of the hole. VV is one way to accomplish this subtask.

In this task the pump base may be mispositioned to the extent of perhaps plus or minus half an inch and rotated plus or minus fifteen degrees, but there will not be any big surprises: the base will not be upside down or at the other end of the workstation.

The class of VV tasks can be placed in perspective by comparing it to other types of visual information processing. Baumgart distinguishes three types: *description*, *recognition*, and *verification* (see [Baumgart 74b]). These types can be conveniently defined in terms of three factors: prior knowledge about *what* can be in a scene (and can therefore appear in a picture of the scene), prior knowledge about *where* things might be with respect to the camera (and hence where they might appear in a picture), and the *goal* of the task. The three types of tasks are:

DESCRIPTION

Prior knowledge only includes the types of features that comprise the objects and how to build complex models from these features; the identity and position of the objects are unknown; the goal is to build a model that describes the scene.

If you have never seen a telephone, but you know about colors, shapes, and sizes, you might *describe* a standard, black telephone, like this: it is black and about half the size of a shoe box; there is a dumbbell-shaped crossbar on top, near the back; it has a round, disk with several holes in it on the top, near the front; the ten digits are arranged in an arc of a circle underneath the holes in the disk.

RECOGNITION

Prior knowledge includes a fixed set of possible object models and occasionally a few constraints on where the objects might occur; the goal is to identify an object in the scene and possibly quantify a few parameters about it.

If you know about telephones, what they look like, where they normally are, and what they are used for, you might identify a telephone in an office like this: you will probably first look on the desk for something black and about half the size of a shoe box. There may be several objects on the desk that you *recognize*: books, pencils, a coffee cup, and a telephone. Having recognized the phone you could roughly describe its location: it is on the far left corner of the desk, almost against the back wall. Garvey has written a program that performs this type of recognition for telephones and other office equipment (see [Garvey 76]).

VERIFICATION

Prior knowledge includes the identity of all objects in the scene and approximately where they are; the goal is to determine the precise location of one or more of the objects.

If you know approximately where the telephone is (e.g., with the same general precision as Garvey's program) and you wish to dial a number, you would *verify* that the phone really is in that general area and then locate the receiver precisely enough to pick it up and

locate the holes precisely enough to insert your finger and dial.

VV provides a way to reduce the uncertainties associated with the location of an object. It bridges the gap between the known tolerances on an object and the desired tolerances *when the initial tolerances greatly restrict the possible appearance of the object.* VV manipulates three-dimensional object models and it can determine three-dimensional corrections, but it assumes that the two-dimensional appearance of the features do not change significantly from one execution of the task to the next. Thus, VV as discussed in this thesis is restricted to quasi two-dimensional tasks. This type of feedback, however, plays an important role in several areas. Programmable assembly is one; aerial photograph interpretation and medical image processing are others.

VV has been used in several ways in the past. Possibly the best known is within the *hypothesis and test* paradigm. For example, a high-level procedure hypothesizes an edge at a certain place; the verification step verifies that the edge is there and computes its position and angle. The model includes exactly what will appear (the edge), approximately where (at some position and orientation), and approximately how it will appear (with some given contrast). There are several systems in which this type of VV plays a major role (see [Falk 70], [Shirai 73], [Tenenbaum 70], and [Grape 73]). Another area in which VV has been used is narrow-angle stereo programs. In these cases, a model is a set of correlation patches from one view of the scene and the goal is to locate these patches in the second view. Again the model states the identity (the unnamed features that produce the correlation patches), the approximate position (near the back-projection of the ray), and the appearance (a slight variation from the correlation patch). See [Quam 74], [Hannah 74], and [Pingle 74] for programs of this type.

More recently there has been considerable interest in visual perception within a programmable assembly system. Such systems provide complex, but predictable environments. For example, consider the task of inserting a screw in a hole. It can be reduced to a few subtasks, each of which could involve VV:

(1) locate the hole with the screw outside the field of view (see figure 1.1),
(2) move the screwdriver and screw into the picture and locate them against the now known background (see figure 1.2),
(3) decide how to move the screw closer to the hole,
and (4) return to step 2 if necessary.

Assume that a mechanical arm picks up the part with the hole and places it in a vise whose position is reasonably well known. In that case the hole may appear displaced in a picture at

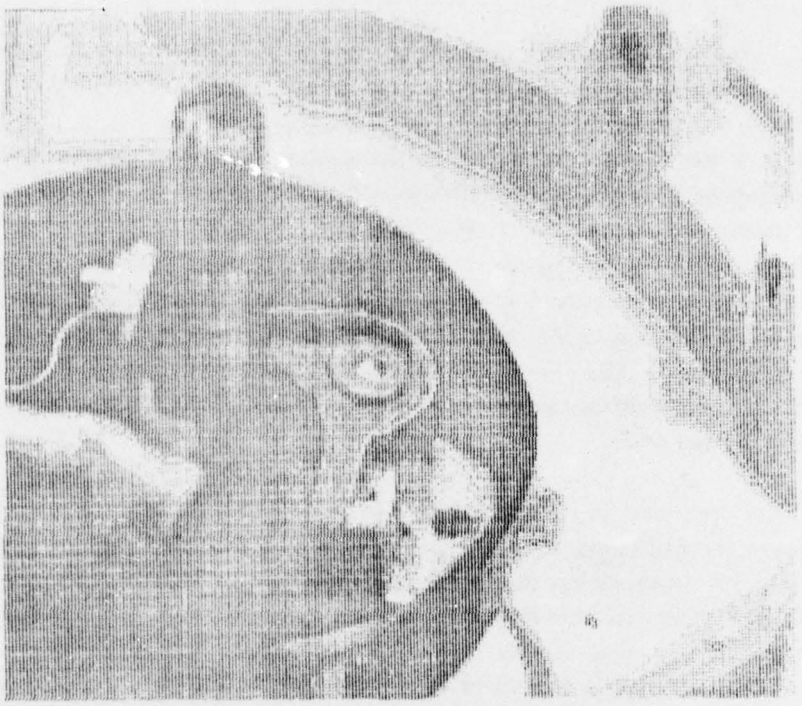


Figure 1.1

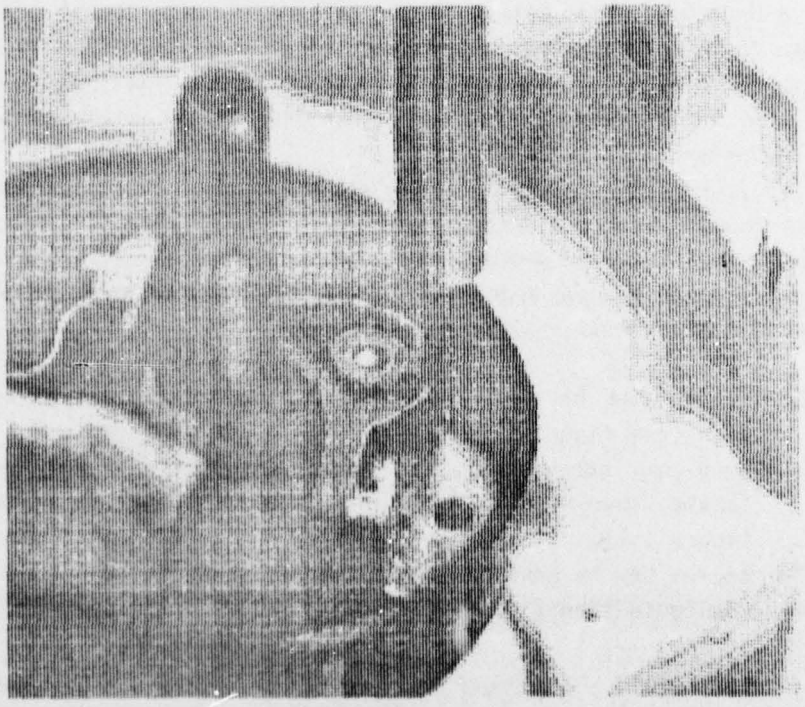


Figure 1.2

step (1) because (a) the part does not seat in the vise exactly as planned and (b) the calibration between the arm and the camera is not exact. Finding the hole in step (1) reduces these factors. Thus, there are fewer uncertainties for step (2). For step (3) the main factor contributing to the error will be the imprecision of the arm since the problem will have been reduced to an analysis of the relative displacement between the tip of the screw and the hole.

More and more information about the expected appearance of the objects can be brought to bear as the program progresses from step to step. For step (1) the only information may be a comparison picture of this same step during a previous assembly and possibly a synthetic picture generated from the model of the expected scene. For step (2) the picture taken at step (1) is available. It contains the background that will appear throughout the task. For step (3) the earlier pictures provide information about actual glares, shadows, and light levels as the screw approaches the hole.

Thus, the three steps offer successively greater constraints on the positions of the objects and greater knowledge about the appearances of the objects. The increasing amount of information should make each successive step easier and faster. The algorithms employed by VV are specifically designed to take advantage of this type of information. VV tries to determine the cheapest, most reliable way to locate an object within a desired precision.

VV is not the only method that a programmable assembly system can use to improve its estimate of the location of an object. In the past, the most common method has been to grasp an object at two or three places and then combine the resulting position information into an estimate for the location of the object. Each grasp is equivalent to a visual operator in the information that it gathers: a value (the thickness of the object at the grasping point) and a position (the position of a point on a plane parallel to the plane of the fingers). But visual operators hold several advantages over grasping operators:

(1) Visual operators are *potentially* faster; they function at electronic speeds as opposed to the mechanical speeds that limit touch and force feedback.

(2) Visual operators are passive; they gather information about an object without disturbing it. This may be important for small, delicate parts.

(3) Visual operators offer a wider variety; some locate a corner, some locate a point on a line, some locate a point on a plane, etc.

(4) Visual operators offer a wider range of scales; the same operators can be used with microscopes or telescopes.

(5) Visual feedback offers a more global view of a situation. For example, it is virtually impossible to use force feedback to decide which way to proceed after a screw has missed a hole. A spiral search by the arm is possible, but slow.

(6) Visual information processing can often be performed concurrently with mechanical motion. For example, if the screwdriver almost always succeeds in picking up a screw from the dispenser, it may be possible to take a picture of the end of the screwdriver as it is leaving the dispenser in order to verify that the screw is present. The arm can move toward the hole while the VV system decides whether or not the screw is present. If the system decides that it is present, the arm is free to continue along its path. However, if the screw is not there, the VV system can signal the arm to return to the dispenser to try again. The economics of this parallel checking depends upon the frequency that the screw is missed and whether or not the resources are otherwise idle.

This list of advantages should not be taken as an argument for the exclusive use of visual feedback. In fact, vision is most effective when it is used in conjunction with touch and force sensing; the different systems can check each other. For example, if visual feedback is trying to servo a screw into a hole, force feedback can indicate that the screw has missed the hole.

One long-range goal of the research described in this thesis is to make it easier for programmers who are not experts in computer vision to program VV feedback. In the past, simple touch and force feedback have been understood and incorporated into mechanical arm programs, but vision has been treated like a never-never land by anyone not involved in vision research. The goal is to make vision a viable alternative within the feedback trio of touch, force, and vision. To do that requires a way to extract useful information from a picture of the workstation, to combine the results of such extractions, and finally to make a decision based upon the summarized results (see figure 1.3).

Programmers generally know how to express their vision tasks in terms of the following three quantities:

(a) the confidence that the system has found the correct object(s),
(b) the precision within which the system has located the object(s),
and (c) the cost involved in determining this information.

These concepts have to be formalized in order for a programmer to specify the goal of a task precisely. But the more difficult problem is to develop methods that produce this information from pictures of the scene. There are two aspects of this second problem: (1) the extraction

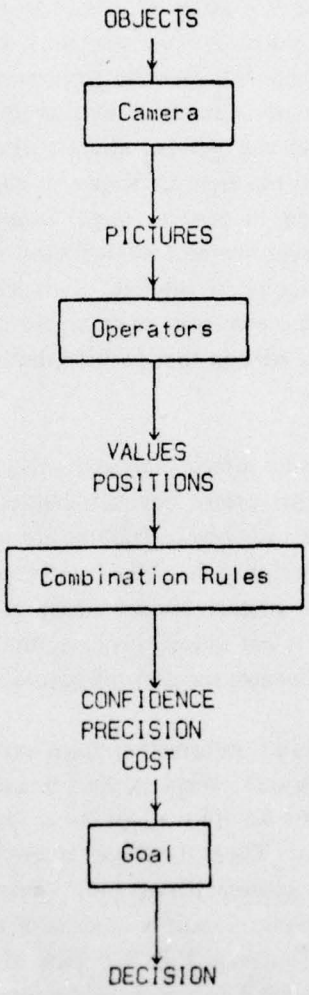

OBJECTS
Camera
PICTURES
Operators
VALUES
POSITIONS
Combination Rules
CONFIDENCE
PRECISION
COST
Goal
DECISION

Figure 1.3

of individual pieces of useful information from a picture and (2) the combination of several pieces of information to form estimates of the quantities of interest. This thesis concentrates on the latter problem.

The implementation of the VV system discussed in this thesis gathers information by applying well-known *operators*, such as edge operators, correlation operators, and region growers, that are designed to locate and describe *features*, such as line segments, correlation points, and regions. The information produced by such operators can be roughly classified into two types: *value* information and *position* information. Value information includes the value of a correlation coefficient, the contrast across an edge, and the intensity of a region. Position information, in addition to (x,y) or (x,y,z) information, may include orientation information. For example, an edge operator might return the (x,y) position of a point on a line and an estimate of the orientation of the line. This information is classified as position information. The same edge operator may compute the contrast across the edge and the confidence that there really is an edge at that position, both of which are classified as value information.

The distinction between value information and position information is natural because often it is reasonable to assume that values from different operators are independent, but it is seldom reasonable to assume that positions of features are *independent* (especially features of rigid objects). Independence means that the value of one operator (such as a correlator) does not affect the expected value for another operator (such as an edge operator). The position information, on the other hand, is *not* independent because the location of one point or the orientation of one line greatly influences the possible positions for other features.

Given the position and value information from several operators, what is the best estimate of the location of an object? What is the precision associated with that estimate? What should the combination rules be? In the past the combination rules have been designed for specific operators and/or tasks. There have been a few special-purpose programs written that perform VV tasks within programmable assembly environments (e.g., see [Bolles 73] and [Dewar 73]) and a few programs that handle a subclass of the VV tasks (e.g., see [Agin 75], [Fischler 71b], [Chien 75] and [Holland 75]), but none of these programs concentrates on precision and confidence and is general enough to accomplish a wide variety of VV tasks.

Part of this thesis describes a set of mathematical tools that form a set of combination rules for the class of VV tasks. A least-squares technique is used to combine available position information to form a current, best estimate of the location of the object (plus a tolerance about that estimate). Bayesian probability is used within a sequential decision scheme to compute the necessary confidences. These techniques are well-known, but they combine particularly nicely to answer the various needs of a VV system.

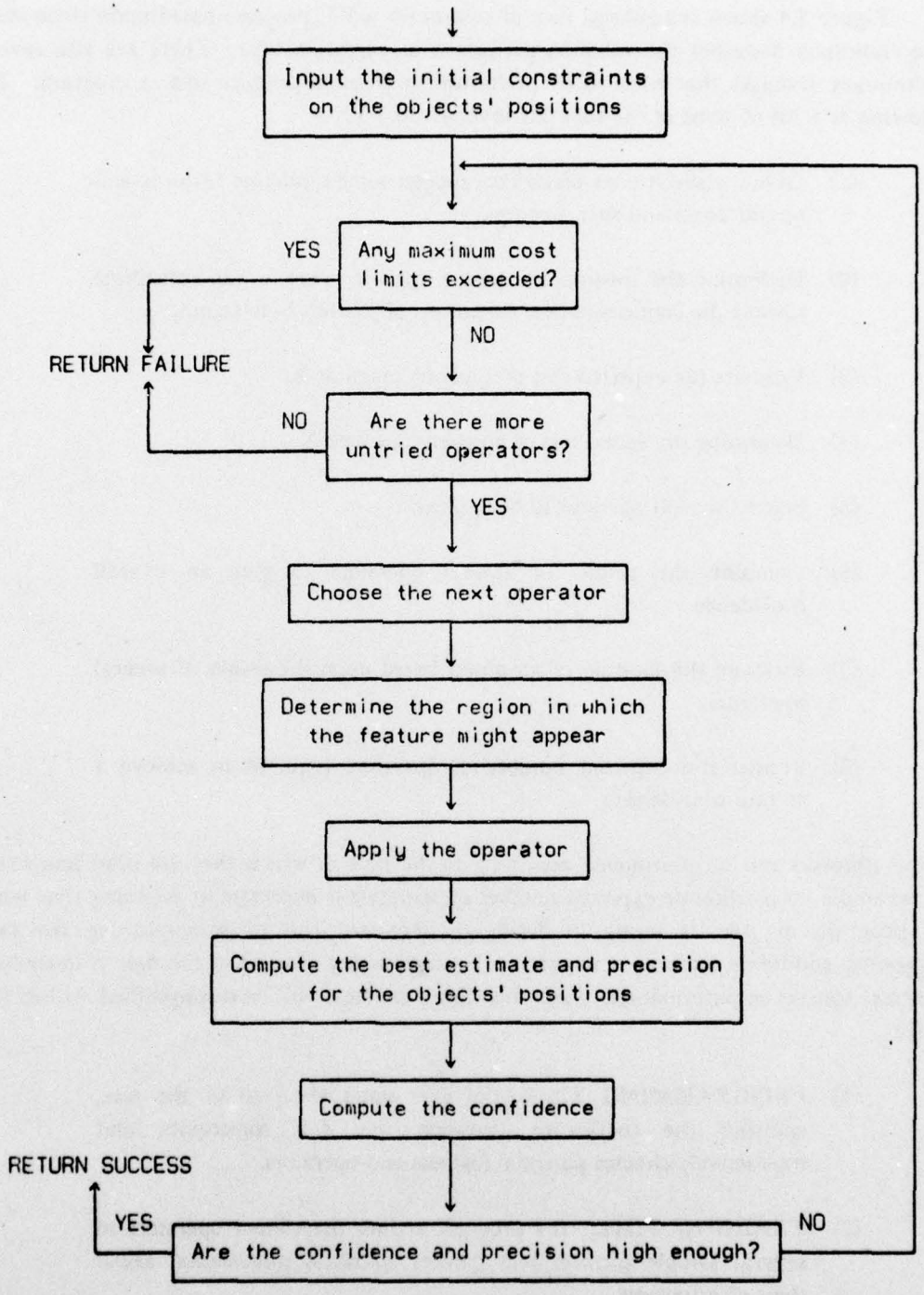

Input the initial constraints on the objects' positions
Any maximum cost limits exceeded?
YES
NO
RETURN FAILURE
Are there more untried operators?
NO
YES
Choose the next operator
Determine the region in which the feature might appear
Apply the operator
Compute the best estimate and precision for the objects' positions
Compute the confidence
RETURN SUCCESS
YES
Are the confidence and precision high enough?
NO

Figure 1.4

Figure 1.4 shows the general flow of control for a VV program based upon these ideas. The flowchart describes the subtasks performed at execution time. There are also several preliminary subtasks that have to be performed in order to produce such a program. The following is a list of some of the subtasks involved in VV:

(1) Given a specific set of objects, suggest some candidate features and operators to find such features.

(2) Determine the information that a specific operator can contribute toward the confidence that the correct object has been found.

(3) Estimate the expected cost of applying operator X.

(4) Determine the actual cost of applying operator X.

(5) Select the next operator to be applied.

(6) Combine the results of several operators to give an overall confidence.

(7) Estimate the location of an object based upon the results of several operators.

(8) Predict the expected number of operators required to achieve a certain confidence.

These subtasks can be partitioned according to the time at which they are most important. For example, to predict the expected number of operators is important at *planning time* when the program or user is trying to decide the expected cost of accomplishing the task. Suggesting candidate features is important at *programming time* when the user is describing potential sources of information. Four such times or stages will be distinguished within this thesis:

(1) PROGRAMMING TIME: the user states the goal of the task, specifies the confidence, precision, and cost constraints, and interactively chooses potential features and operators.

(2) TRAINING TIME: the program applies the chosen operators to several sample pictures and gathers statistical information about their effectiveness.

(3) PLANNING TIME: the system ranks the operators according to their expected contribution, determines the expected number of operators to be needed, and predicts the cost of accomplishing the task.

(4) EXECUTION TIME: the system applies operators one at a time, combines the results into confidences and precision, and stops when the desired levels have been reached or a cost limit has been exceeded.

To accomplish a VV task requires progress from one stage to the next in the order shown above. However, for clarity, these stages will be discussed in a different order: execution time, planning time, programming time, and training time. The execution-time discussion describes how the results of operators are combined for the two types of tasks, inspection and location; the planning-time discussion describes how the application order and number of the operators is determined; the programming-time discussion describes how an operator/feature pair is chosen as a potential source of information; the training-time discussion describes how the potential benefits of an operator are characterized from several trial runs. The thesis concludes with a description of some possible extensions.

This thesis relies heavily upon the domain of programmable assembly for its examples and motivation (e.g., see [Finkel 75] and [Finkel 76]). The techniques are discussed in the context of a highly controlled environment in which mechanical arms are performing assembly task's. Some of the techniques have been optimized to take advantage of specific properties of this environment, but the basic methods used to produce location and confidence information from the results of several visual operators are more widely applicable. Other possible areas include photo-interpretation and medical image processing.

There is one other general remark that should be made at the beginning of this discussion. Although most of the examples and techniques described here are based upon conventional television cameras and their images, there is no reason why the same or similar techniques could not be used within systems based upon direct ranging devices, laser trackers, or multiple touch sensors.

CHAPTER 2

MOTIVATION

The purpose of this chapter is to outline the general requirements and the semantic structures necessary for a VV system that can be easily programmed (see [Bolles 75]). Two example tasks and their solutions are used to introduce the desired capabilities. Later chapters will discuss specific mechanisms in detail. Appendices IV and V contain traces of a programmer using the current implementation of the VV system to set up and test VV programs that perform the two tasks discussed in this chapter.

The two example tasks are (1) check for a screw on the end of the screwdriver and (2) locate a screw hole in a part that has been placed in a vise. The goal of the first task is to make a *yes* or *no* decision: Is the screw present? VV tasks of this type will be referred to as *inspection* tasks. The goal of a *location* task, on the other hand, is to estimate the current location of the object, or equivalently to estimate the displacement between the current location of the object and its planned location. The success of an inspection task is usually measured by its *confidence*; the success of a location task is usually measured by its *precision*.

Section 1
CHECKING FOR A SCREW

Consider visually deciding whether or not there is a screw on the end of the screwdriver. One idea for a solution is to aim a camera at the exit of the screw dispenser, take a picture of the end of the screwdriver as it leaves the dispenser, apply a series of operators to the picture, and use the values of the operators to decide whether or not the screw is on the end. There are several decisions to be made before this idea can be converted into an algorithm to perform the desired task: Which visual operators should be applied? Where should they be applied? In what order should the operators be applied? How should the results of several operators be combined? How much can an operator contribute toward

the final decision? These and other questions will be briefly discussed below.

1. What are some potentially useful operator/feature pairs?

Since the goal of this task is to decide whether or not the screw is on the end of the screwdriver, each operator chosen should contribute toward this decision. Consider figure 2.1.1, which shows typical pictures of the two expected situations: one with the screw on the end of the screwdriver and one with the screw missing. An operator that can distinguish the texture formed by the screw threads is a potentially useful operator, because the presence of screw threads distinguishes the two situations. An operator that locates a feature on the screwdriver would not directly contribute to a decision because the screwdriver is present whether the screw is there or not. But this operator may still be useful to improve the system's estimate of the location of the screwdriver tip and hence make it easier for the texture operator to find the screw threads. Thus, operators may directly or indirectly contribute to the final decision.

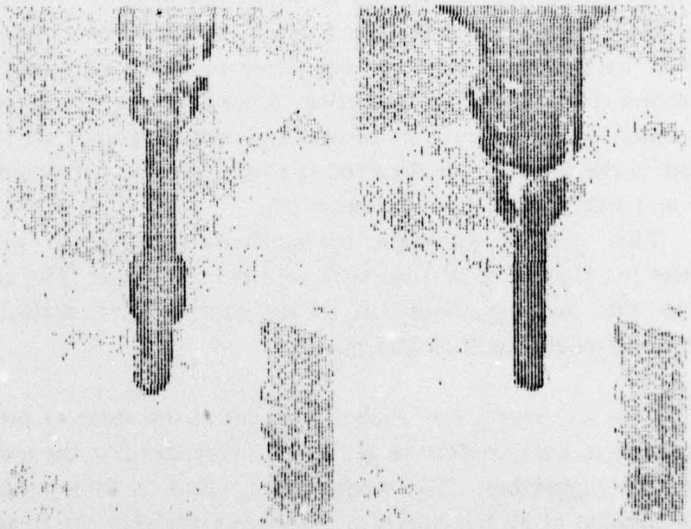


Figure 2.1.1

In general, the objects in a scene have several visual features: colors, textures, sizes, edges, corners, and holes. The more unique features an object has, the easier it is to find. One reason the *blocks world* is difficult to analyze visually is that blocks do not have very many distinctive features. All blocks have edges, corners, and planes so it is difficult to distinguish one from another. Programmable assembly, on the other hand, offers a wide variety of features.

Given a wide variety of features, which ones would be useful to locate? Which operators should be applied to find such features? A VV program can provide a user with three levels of assistance in order to help him choose potentially useful operator/feature pairs for a specific task:

(1) The VV system can provide a convenient environment within which to experiment with different operators. The user might describe features such as lines, corners, and correlation patches and apply various operators to locate them in trial pictures. For example, the user could describe the corner formed by the head of the screw and the shaft of the screwdriver (see figure 2.1.2), try a corner-finder to find it, try a correlation operator to find it, choose the more promising operator, and add the chosen operator and corner to the list of potentially useful operator/feature pairs.

(2) The VV system can analyze a typical picture of the scene to produce a ranked list of potential operator/feature pairs. For example, an edge operator may be applied to the picture in order to pick out all pairs of line segments that form a corner of a certain minimum size. One of these corners might be the corner formed by the head of the screw and the shaft of the screwdriver. Since both a corner-finder and a correlation operator can locate corners, two operator/feature pairs can be added to the list of suggestions for each corner: a corner-finder/corner pair and a correlation operator/corner pair.

This type of automatic operator/feature suggestion procedure reduces the amount of detailed work required of the user. The user only has to filter out suggestions that are difficult to locate reliably or that produce unreliable position information.

(3) The VV system can analyze a model of the scene to produce a ranked list of operator/feature pairs. The more complete the model, the better the suggestions. The model might include a three-dimensional representation of all the objects in the scene, a model of the camera, and a model of the light sources. For example, a hidden-line elimination scheme can be used to predict visible corners from models of the screw and screwdriver. One of these corners might be the one formed by the head of the screw and the shaft of the screwdriver.

This type of automatic suggestion procedure can analyze potential operator/feature pairs at a higher level than the system mentioned above. Potentially it can bring to bear all the knowledge associated with real objects: their appearance, their structure, and their function. However,

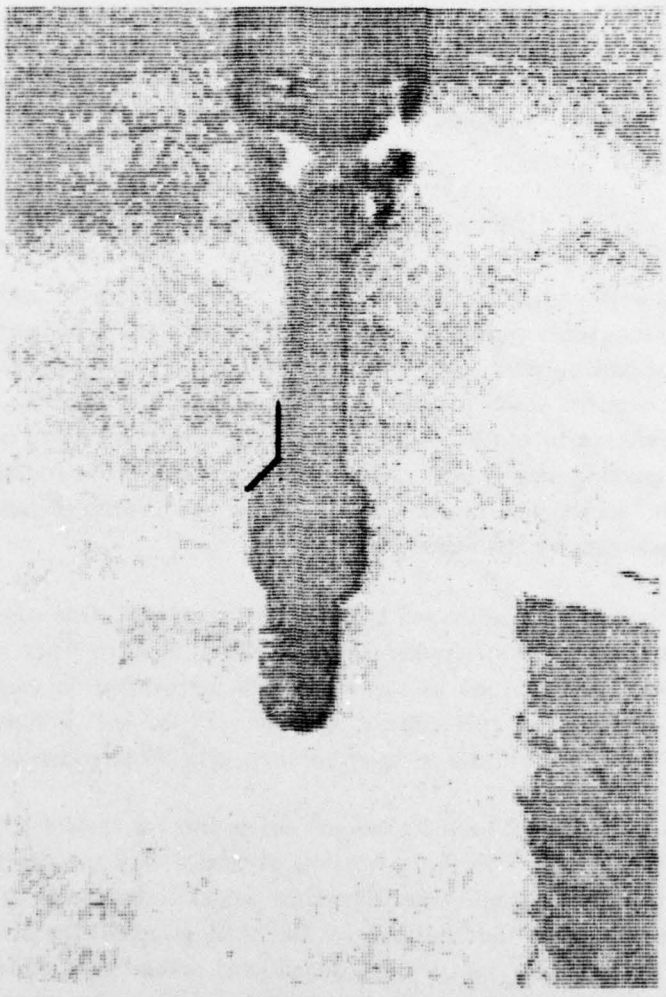


Figure 2.1.2

the results are only as good as the model, whereas in (2) the results are only as good as the training picture.

This list is ordered according to the sophistication of the proposed system; the later procedures require more information about the task and they can analyze the potential suggestions more efficiently. For example, the second level might suggest a corner feature, one side of which is formed by a shadow. If the shadow changes from one trial picture to the next, the feature is probably not a good one. The only way that the second level system can arrive at this conclusion is (1) to monitor the relative positions of several features in several trial pictures and (2) to notice that the corner changes position with respect to the other features. The third level system could directly determine that one side is formed by a shadow and discard this feature immediately.

2. *Where should an operator be applied?*

In the screw inspection task, since the arm and camera are not exact, the end of the screwdriver will sometimes appear at one point in the picture and sometimes at another. If the total range of possible positions is only a small portion of the picture, there is no reason to apply operators over the whole picture. The region of possible positions for a feature in a picture will be referred to as the *tolerance region* for the feature. It can be determined once during a programming session and can remain fixed during the training, planning, and execution phases. In order to find the feature during one of these phases, only its tolerance region must be scanned, not the whole picture.

How can the tolerance region for a feature be determined? One way would be to have the arm attempt to get a screw from the dispenser several hundred times, take a picture after each attempt, mark the position of the end of the screwdriver in each of the pictures, incorporate all of the occurrences into a continuous region (e.g., a convex polygon), and declare that region to be the tolerance region for the end of the screwdriver.

Another way to determine the tolerance region about a feature is to ask the user to specify the constraints that limit the uncertainty associated with the position of the feature. For the screw-checking example the constraints would include such information as the accuracy of the arm and the accuracy of the hand grasping the screwdriver. These constraints can be translated into a three-dimensional volume that represents the possible positions of the feature. This volume will be referred to as the *tolerance volume* of the feature. The tolerance region for the feature can be formed by projecting its tolerance volume onto the screen.

In order to project the three-dimensional volume onto the camera screen a camera calibration is needed. Camera calibrations are designed to provide (1) a transform that maps

a point in the workstation coordinate system onto a point in the screen coordinate system and (2) a transform that maps a point in the screen coordinate system into a ray in the workstation coordinate system. Some additional piece of information (e.g., the height of a point) is necessary to map a point on the screen into a point in the workstation coordinate system (e.g., see [Sobel 74]). Camera calibrations are not exact, so tolerance regions should be expanded to accommodate for this additional inaccuracy. The application of a camera calibration to move back and forth between the screen coordinate system and the workstation coordinate system will be used several times throughout the remainder of this discussion.

Given a feature, its tolerance region, and an operator to find the feature, where should the operator be applied within the region to locate the best match for the feature? The search strategy depends upon several factors, including the type of feature, the operator being used, the size of the tolerance region, and the feature's expected distribution of positions within the tolerance region. If a correlation operator is used to locate the corner formed by the head of the screw and the shaft of the screwdriver, an exhaustive search may be reasonable if the tolerance region is small. When an edge operator is used to locate a point on a line segment, a few linear scans across the tolerance region are often sufficient. A set of *search techniques* and methods to predict their expected cost in different situations are required in order to choose a good search strategy.

3. How should the results of an operator be interpreted?

Consider a hypothetical texture operator that ranks local regions in a picture according to their similarity to the texture formed by the screw threads. It may return a value of .95 when applied to an example picture with the screw present (see figure 2.1.1) and .57 when applied to an example picture with the screw missing. If the same operator is applied to a new picture to decide whether or not the screw is present, and it returns a value of .86, what is the probability that the screw is on the end of the screwdriver? How can the *a priori* probabilities be incorporated into the computation?

If one operator implies that there is a probability of .76 that the screw is present and another operator implies that there is a probability of .84 that the screw is present, what is the overall probability of the screw being present? How are the results of the two operators related? In general, how can the values of several operators be integrated into one estimate for the probability that the screw is present?

There are two errors that can be made in a task of this sort: (a) the operator values may imply that the screw is present when it is not, and (b) the operator values may imply that the screw is missing when, in fact, it is present. These errors are referred to as errors of the first and second type, respectively [Shewhart 39]. How can an assembly engineer set the limits on the acceptable number of errors of each type? Given such limits and a set of operators,

what is the expected number of operators that will have to be applied in order to make a decision that satisfies these limits?

The combination rules are complicated by the fact that visual operators are not completely reliable. Sometimes the best match they find is not the intended match. For example, consider the correlation operator shown in figure 2.1.3. If it is applied throughout the tolerance region shown in figure 2.1.3.b, it will probably find two good matches, as shown in figure 2.1.3.c. If the operator happens to prefer match B, it will return an incorrect match. In general, when an operator is applied at several positions within a tolerance region, it returns a set of different values. Which position is the best match? What is the chance that the best match is really the correct match? How can the combination rules adjust for this unreliability?

The possibility of unreliable operators means that (1) the user should check for potential confusions at programming time; (2) the training-time subsystem should gather statistics on the reliability of the operators; (3) the planning-time subsystem should reduce the desirability of unreliable operators; and (4) the execution-time combination rules should reduce the contribution of any operator known to be unreliable.

4. Which operator should be applied first?

Some operators find their matches more easily than others; some operators contribute more toward the final decision than others. In what order should the operators be applied? For example, in the screw-checking task the screw thread operator may be faster than the corner-finder, but the corner-finder may produce more information than the screw thread operator. Which one should be applied first?

One possible choice mechanism is to apply the operator with the largest expected value for the ratio

$$\frac{\langle \text{contribution toward the final decision} \rangle}{\langle \text{cost} \rangle}$$

first. But what is the *expected* contribution of an operator? At execution time the combination rules compute a specific contribution for a specific value of an operator. The planning mechanism needs the *average* contribution of the operator, which depends upon the distribution of the values of the operator as well as the contribution derived from each value.

As mentioned earlier, operators can contribute indirectly as well as directly toward the final decision. How can these indirect contributions be incorporated into the ranking of the

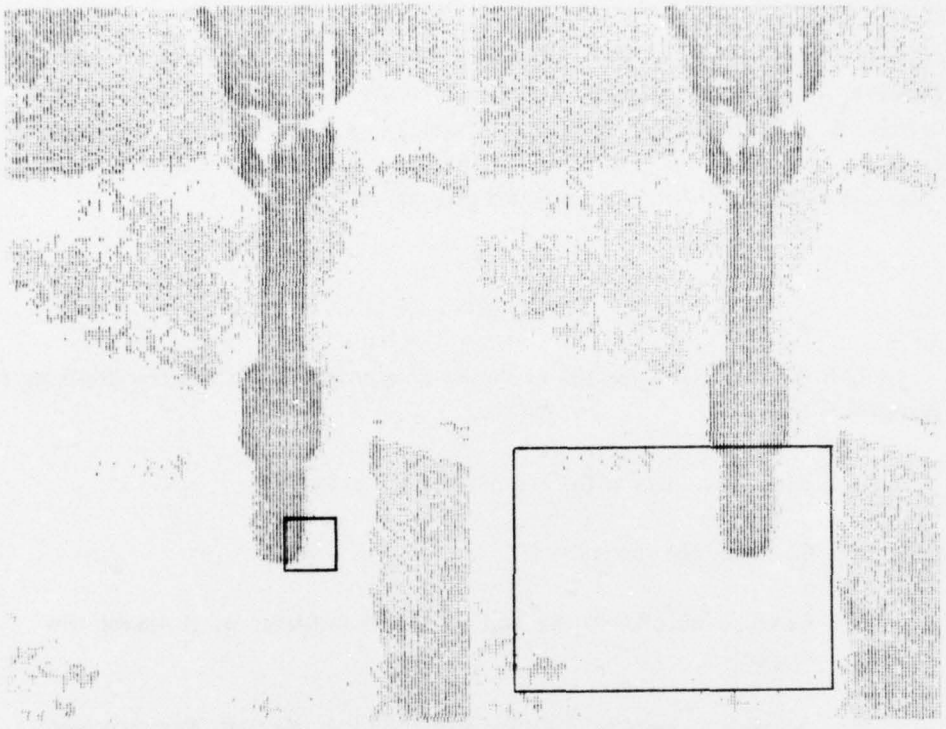


Figure 2.1.3.a

Figure 2.1.3.b

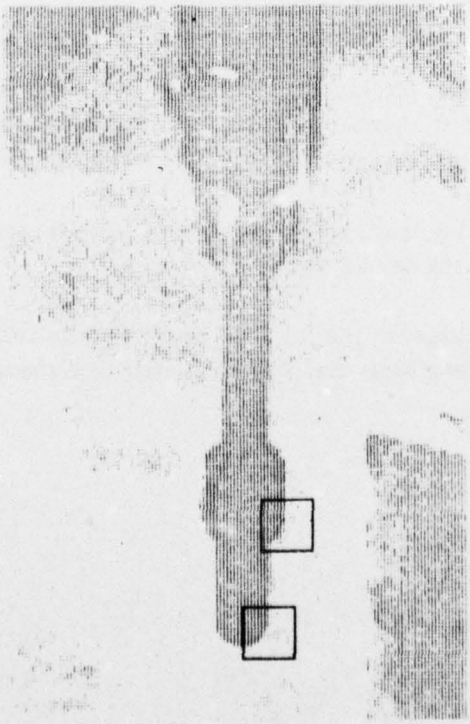


Figure 2.1.3.c

operators? For example, an edge operator that locates a point on the side of the shaft of the screwdriver may be the cheapest operator to apply in the screw-checking task. It only indirectly contributes to the task by increasing the constraints on the position of the screwdriver and screw, but it still may be the best first step toward the final decision. How can the strategist take this type of expected progress into account?

5. *A refined VV program*

In light of this discussion the execution-time program for the screw-checking task can be restated as follows:

(a) Aim the camera at the exit of the screw dispenser.

(b) Calibrate the camera.

(c) Take a picture of the end of the screwdriver as it leaves the dispenser.

(d) Apply one operator at a time, using the best one first. For each one, employ the appropriate search technique to locate the best match within the tolerance region of the feature.

(e) Incrementally incorporate the results of each operator into an estimate of the probability that the screw is present. Be aware of possible confusions and adjust the probabilities accordingly.

(f) Stop applying operators and make a decision as soon as one can be made with the desired confidence.

The calibration steps (steps a and b) may be performed only once for each series of assemblies. The remaining steps are performed each time the arm tries to obtain a screw from the screw dispenser.

Section 2
LOCATING A HOLE

Consider visually locating a screw hole in a part that has been placed in a vise. Such a task is a location task; the user is more concerned about the location and precision associated with the hole than about the confidence that the hole is present. There is no question about *what* is sought, just an uncertainty about *where* it is.

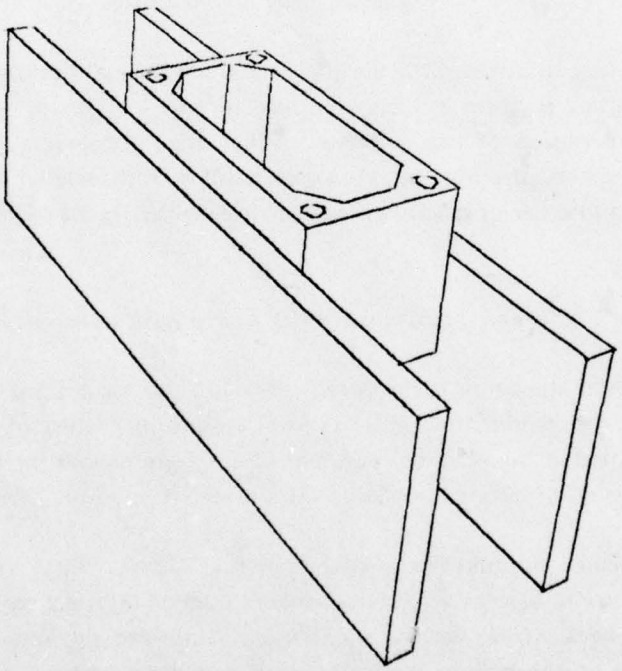


Figure 2.2.1

Assume that a flat side of the part is placed in contact with one of the vise jaws (see figure 2.2.1). Then the part can only be rotated and translated in the plane parallel to the jaws. If the vise location is well-known, the uncertainties associated with the location of the part can be described as a planar transformation, which is a function of three parameters: dx,

dy, and $d\alpha$. The parameters dx and dy are unknown translations in the plane of the vise jaws; $d\alpha$ is an unknown rotation about a vector perpendicular to the plane of the vise jaws. Constraint resolving routines can combine the accuracy of the arm and the accuracy of grasping the part to produce *a priori* limits on these parameters. For example,

$$-1.32 \text{ cm} \leq dx \leq 1.32 \text{ cm}$$
$$-.76 \text{ cm} \leq dy \leq .76 \text{ cm}$$
$$-10.4 \text{ degrees} \leq d\alpha \leq 10.4 \text{ degrees}.$$

The goal of the task is to improve these limits to a prescribed value. For example, the arm control program might need to know the location of the screw hole to within the following tolerances:

$$-.15 \text{ cm} \leq dx \leq .15 \text{ cm}$$
$$-.15 \text{ cm} \leq dy \leq .15 \text{ cm}.$$

A general way to accomplish this goal is to apply several operators and combine their position information to form a better estimate for the location of the screw hole and a measure of the precision of that estimate. What are the combination rules necessary to produce such estimates, and how can the precision of each individual operator be taken into account? These and other questions will be briefly discussed in the following subsections.

1. How can position estimates and precisions be computed?

In the hole-locating task each operator locates a specific point on the part in the vise. The function of the combination rules is to determine the values of dx, dy, and $d\alpha$ that transform the expected (or planned) positions of the features into the observed positions of the features. This set of combination rules will be referred to as the *fitting scheme*.

Fitting schemes encounter several complicating factors. First, visual operators locate matches in a picture of a scene and hence produce position information within the coordinate system of the camera screen, not the workstation. Since the arm control program needs to know the location of the hole in the workstation coordinate system, the position information has to be transformed by the camera calibration into the workstation coordinate system. If two or more cameras are being used, stereo techniques can directly locate features in the workstation coordinate system. With one camera, however, additional position information is needed to determine a feature's workstation coordinates. In the screw hole task, since the uncertainties have the form of a planar transform, the distance from a feature to the known plane is sufficient. That is, given the screen coordinates for the image of a feature and the height of the feature, it is possible to locate the unique, three-dimensional position for the feature. Stereo is preferable, but useful information can be derived from a single camera and

some additional information.

The second complication is that some operators are more precise than others. For example, a correlation operator may locate the corner formed by the head of the screw and shaft of the screwdriver to within 1.5 pixels and the corner-finder may locate the same corner to within half a pixel. (A *pixel* is one resolution element in a digitized picture.) The fitting scheme should be able to weight the results of these operators appropriately.

A third complication is that some operators do not locate a specific point on the part; they locate a piece of a line or a portion of a region. For example, the edge operator that looks for the side of the shaft locates a point on a line segment and the hypothetical screw thread operator locates a portion of a small region. The fitting scheme should be general enough to incorporate these different types of position information.

A fourth complication is that an operator may locate a decoy instead of the correct match. If the decoy looks locally like the desired match, the only way to determine that it is a decoy is to check the global structure of the matches. Since the features of a rigid object are expected to remain at fixed relative positions, it should be possible to check a set of feature matches for their *structural consistency*. Do the matching positions correspond to some reasonable transformation of the object? Structural consistency can also be incorporated into the confidence computations used in inspection tasks. If the program is checking for a vertical screw, and a pair of operators imply that the tip of the screw and the top of the screw are side by side, one ought to be suspicious.

This discussion of decoys and unreliable operators suggests two general conclusions: (1) the confidence that an operator has located the correct match is just as important within location tasks as it is within inspection tasks, and (2) position information (i.e., structural consistency) can make a significant contribution in an inspection task toward the overall confidence in a decision. In other words inspection tasks should be concerned with position information in addition to confidences, and location tasks should be concerned with confidences in addition to position information. Thus, structural consistency and confidence are important within both types of tasks.

Finally, the fitting scheme should be able to incorporate new information incrementally. As more operators are applied and matches are found, their results should be combined with previous results. Intermediate estimates can be used to reduce the amount of searching required to locate a new feature. For example, after a point on the shaft of the screwdriver has been located, its position and precision can be combined with *a priori* limits associated with the location of the screw to form a new, smaller tolerance region about the corner formed by the head of the screw and the side of the shaft. The use of intermediate estimates to reduce tolerance regions re-emphasizes the importance of the order of the operators and the need for a sequential strategy to locate the desired object to within the desired precision.

2. A refined VV program to locate the screw hole

The execution-time VV program to locate the screw hole can be restated as follows:

(a) Aim the camera at the vise.

(b) Calibrate the camera.

(c) Apply one operator at a time in the most strategic order.

(d) Use each operator's position information to update the estimates for the parameters, dx, dy, and $d\alpha$.

(e) Use the estimates for the parameters to determine the expected position of the next feature and a new, improved tolerance region about that position.

(f) Be cautious about the implications drawn from the position information because the operators are not completely reliable.

Section 3
SUMMARY OF REQUIRED FACILITIES

This section summarizes the facilities required by a VV system. It lists the main components, states their role within the complete system, and briefly describes the current state-of-the-art in each area.

1. Calibration Techniques

Calibration routines provide transformations back and forth between the screen coordinate system and the workstation coordinate system. Almost every computer vision system has its own methods for calibrating the cameras. The basic techniques can be found in [Sobel 74], [Sproull 73], and [Baumgart 74b].

Calibration can itself be described as a VV task: place a known object at a known position, visually locate the object, and compute the current position of the camera. The advantage of this formulation is that the same VV program can be used to calibrate the

cameras and to perform the desired feedback task; a special-purpose calibration subsystem is not necessary. Another advantage of this approach is that the camera can directly calibrate itself with respect to the stationary objects in the scene (e.g. the vise or the screw dispenser) instead of a special calibration object. Gill used his corner-finder to develop a version of this type of calibration system for the blocks world (see [Gill 72]).

A calibration routine may also include an estimate for the camera noise and a description of the lens distortion. These quantities are important factors in predicting the appearances and positions of features. Gennery and Moravec have developed a program to estimate these quantities [Moravec 76].

2. *Visual Operators*

A visual operator locates the image of a feature and returns an estimate for its location and a description of its appearance. For example, a region grower might locate a small, dark, elliptical region inside a larger, grey region; an edge operator might locate a distinct, high-contrast edge with some particular orientation at a certain location.

There are several well-known types of operators: edge operators (e.g. see [Roberts 63], [Horn 71], and [Hueckel 69]), line followers (e.g. see [Tenenbaum 70] and [Shirai 73]), corner finders (e.g. see [Gill 72] and [Perkins 73]), correlation operators (e.g. see [Quam 71], [Hannah 74], and [Moravec 76]), region growers (e.g. see [Brice 70], [Yakimovsky 73a], [Agin 75], and [Garvey 76]), and texture operators (e.g. see [Bajcsy 72], [Lieberman 74] and [Marr 75a]). There is, however, still a need for a wider variety of more powerful operators. Some of the most useful would be operators to grow textured regions and locate boundaries between two textured regions.

3. *Search Strategies*

The purpose of a search strategy is to choose where to apply an operator in order to locate the best (or at least a good) match as cheaply as possible. Strategies may include heuristics to avoid the cost of an exhaustive search. For example, Moravec uses a two-dimensional binary search to locate a good match for a correlation operator (see [Moravec 76]). Exhaustive searches can also be avoided when looking for extended features, such as lines and regions. The larger the feature, the easier it is to find.

If an operator is known to be unreliable, a search strategy may be used to produce an ordered list of the best three or four matches for the operator. A VV program may try one match after another until it finds a structurally consistent one.

4. A Light Model

A light model describes the position, brightness, and spectrum of each light source at the workstation. It can be used in conjunction with the object models to predict the appearance of an object. The angle of incidence, the reflectance of the object, and the angle of observation are the basic variables that determine the appearance of a point on an object.

There are several programs that use object models and light models to produce synthetic pictures. Some of the most complete graphic display systems have been developed at the University of Utah (e.g., see [Gouraud 71]). Such systems are general enough to handle objects with smooth, curved surfaces (e.g. car bodies), transparent objects (e.g. wine glasses), and shadows. The two main restrictions associated with these programs are that (1) the object models are difficult to construct and (2) they require a great deal of computation time.

Horn at MIT [Horn 70] has investigated techniques to reverse this process. That is, use pixel intensities, contours between the intensity levels, object models, and reflectance properties of light to recognize an object in a scene. It is difficult to analyze complex pictures because of the interaction of several light sources and the reflection from several small surfaces.

5. Object Models

Object models are used to predict the positions and appearances of features. For example, if the model includes descriptions of the shapes and surfaces of the object, it is possible to predict the positions, shapes, and colors of the corresponding regions in a picture of the object.

Object modelling at this level of detail is quite complex. The representation depends upon the purpose of the model. It makes a difference whether the object is going to be looked at, picked up, or painted. Models may include one or all of the following facets: a rigid inter-affixment of features (e.g. see [Finkel 74], [Taylor 76], and [Lieberman 75b]), a structural description (e.g. see [Agin 72], [Nevatia 74], [Baumgart 74a], [Grossman 75a], [Miyamoto 75], and [PADL 74]), an articulation description (e.g. see [Nevatia 74]), and a description of the surfaces (e.g. see [Coons 67], [Gorden 72], and [Gould 72]). These individual components are reasonably well understood, but there are no existing systems that provide all of them.

6. A Constraint Resolving Algorithm

A constraint resolving algorithm takes as input an object model and a set of constraints on the object, and produces a list of the remaining degrees of freedom and the resulting

constraints associated with those degrees of freedom. For example, if a coffee cup is known to be sitting upright on the table and a vision program locates the center of the hole on top, the constraint system produces a locus of points that describes the possible positions for the handle of the coffee cup.

A completely general constraint resolving system is difficult to program because an object may have as many as six nonlinearly related degrees of freedom (three displacements and three rotations) and the loci are often quite complex subregions of this 6-space. Ambler and Popplestone have investigated a limited set of constraints for cylinders and V-blocks (see [Ambler 73]). Taylor has developed a linear programming system that handles a wider range of constraints and produces specific tolerance information (see [Taylor 76]). Grossman has combined a graphics model with a Monte Carlo simulation technique to produce the distributions formed by a set of input constraints (see [Grossman 76]). All of these programs are steps in the right direction, but there is still a great deal of work to be done.

7. Combination Rules for Probabilities

Combination rules use *a priori* probabilities and the value and position information provided by the operators to produce estimates for various probabilities. Typical probabilities of interest are the probability that an object is present (e.g. a screw), the probability that the current match is the correct match, and the probability that a set of matches is structurally consistent.

Probability analyses have been used for numerous tasks. Some of the formulations that are the closest to the one developed in this thesis are: medical diagnosis models (e.g. see [Shortliffe 75], and [Davis 76]), a fault detection model (e.g. see [Nilsson 75]), and parts recognition models (e.g. see [Duda] and [Rosen 74]). All of these models use Bayes' theorem or a closely related mechanism to compute the desired *a posteriori* probabilities.

All of these systems have to make some strong independence and conditional independence assumptions in order to convert Bayes' theorem into a usable form. Similar assumptions are made in this thesis. The validity of the results depend upon the validity of these assumptions.

8. Combination Rules for Position Information

Combination rules for position information use position and orientation estimates from individual operators to produce an overall estimate for the location of the object. The rules can also use the precisions associated with each position estimate to produce an overall precision estimate for the location of the object. For example, if it is known that a screw dispenser is sitting upright on the table and if a vision program locates two features on the dispenser, then the combination rules should be able to produce an estimate for the location

of the dispenser. The precision of this estimate depends upon the precision of the position information of the operators and on the structural relationship between the two features. If the line joining the two features happens to be perpendicular to the table, the overall contribution of the two features will be less than if the line is parallel to the table, because the plane of the uncertainties is parallel to the table.

Combination rules of this type are closely related to constraint resolving procedures. Both reduce a set of constraints into a single, overall description of the remaining uncertainties. The difference is that the combination rules are designed to deal with unreliable operators. In particular, they have to deal with inconsistent pieces of information, some of which are seriously in error, and should be discarded, and some of which are only slightly inconsistent and should be included in the computation.

Some fitting schemes, such as least-squares, are well-known and have been used in a wide variety of tasks. Almost from the very beginning of computer vision research, fitting schemes have been used to optimize various criteria. Roberts used a least-squares routine in one of the intermediate steps involved in deciding which block prototype best fit the observed data (see [Roberts 63]). Perkins used a least-squares fitting technique within his corner-finder to locate the lines that form the corner [Perkins 73]. Gennery used a least-squares fitting routine to calibrate a stereo pair of cameras (see [Gennery 75]).

Fischler and Elschlager have taken a slightly different approach to the combination rules problem. They use a linear programming scheme to determine the optimum matching position for a structure of features (see [Fischler 71b]). Their method can weight each feature and the links between the features. Their technique, however, is more concerned with the position of the best overall match than with the precision and confidence associated with that match.

9. A Stategist

A strategist tries to construct the most efficient program to accomplish a task by ranking the available operators and choosing the most appropriate control structures. It bases its decisions upon the expected contributions and costs of the operators when applied to the specific task. For example, a strategist may decide to try to locate a point on a large ellipse and use the information from that operation to locate the desired hole. Or the strategy program may decide to locate a point on the ellipse, follow the ellipse a short distance in order to refine the system's estimate of the location of the ellipse, and then try to locate the desired hole.

The artificial intelligence community has been working on the strategy problem for a long time. The basic approaches can be found in [McCarthy 58], [Nilsson 71], [Fikes 71], [Sacerdoti 75b], and [Taylor 76]. Feldman and Sproull have developed one of the most

comprehensive systems to deal with costs, constraints, and confidences (see [Feldman 75] and [Sproull 77]). Their system is based upon a decision theoretic model of strategies. Other examples of the use of strategy programs within computer vision research can be found in [Yakimovsky 73a] and [Garvey 76].

10. An Interactive Programming System

A VV system that can be easily programmed requires a human engineered, top-level system that interacts with the user. This type of control program provides an environment in which a user can experiment with different operators, arrange VV programs, and test them on trial pictures.

Recently considerable interest has been shown in teach-by-doing programming techniques for arm programs (e.g. see [Rosen 74]). Some of this enthusiasm has extended into vision research. For example, an interactive parts recognition system has been implemented at SRI [Rosen 74]. Two other vision systems that provide this type of user interface are described in [Agin 75] and [Garvey 76].

11. Conclusions

The purpose of this chapter was to outline the set of capabilities required within a VV system. Systems already exist that provide several of these capabilities, such as camera calibration and object modelling. However, two of the most important capabilities are missing: combination rules for probabilities and combination rules for position information. The next three chapters develop a set of these combination rules for VV.

CHAPTER 3

EXECUTION-TIME COMBINATION RULES FOR INSPECTION

The introductory chapters have stressed the importance of combining the results of several, possibly unreliable tests. Chapters 3, 4, and 5 derive combination rules for VV. The derivations are fundamental, but the reader may wish to look at some examples first. Appendices IV and V contain extended examples of the application of these combination rules. The examples provide a general feeling for the behavior of the rules.

There are two types of combination rules: execution-time rules and planning-time rules. *The former is concerned with combining the actual* results of the operators as they find features. The latter is concerned with computing and combining the *expected* contributions of the operators. In this chapter we deal with execution-time rules needed to accomplish inspection. Execution-time rules for location are discussed in the next chapter.

The combination rules are incrementally developed in conjunction with a sequence of examples designed to incorporate increasing levels of complexity. Each section, except the last one, extends the rules to cover one additional aspect of the problem. The last section discusses some of the important assumptions that are made in the first five sections in order to derive the combination rules.

Section 1
OPERATOR VALUE INFORMATION

Consider the standard inspection task: decide whether or not there is a screw on the end of the screwdriver. For simplicity assume that normalized cross-correlation is the only type of operator known to the VV system. Correlation uses patches from a planning picture as features to be found in a test picture. Figure 3.1.1 shows a planning picture with the screw on the end of the screwdriver and several sample pictures, some with the screw present, some with it missing. Figure 3.1.2 shows several correlation patches outlined on top of the planning

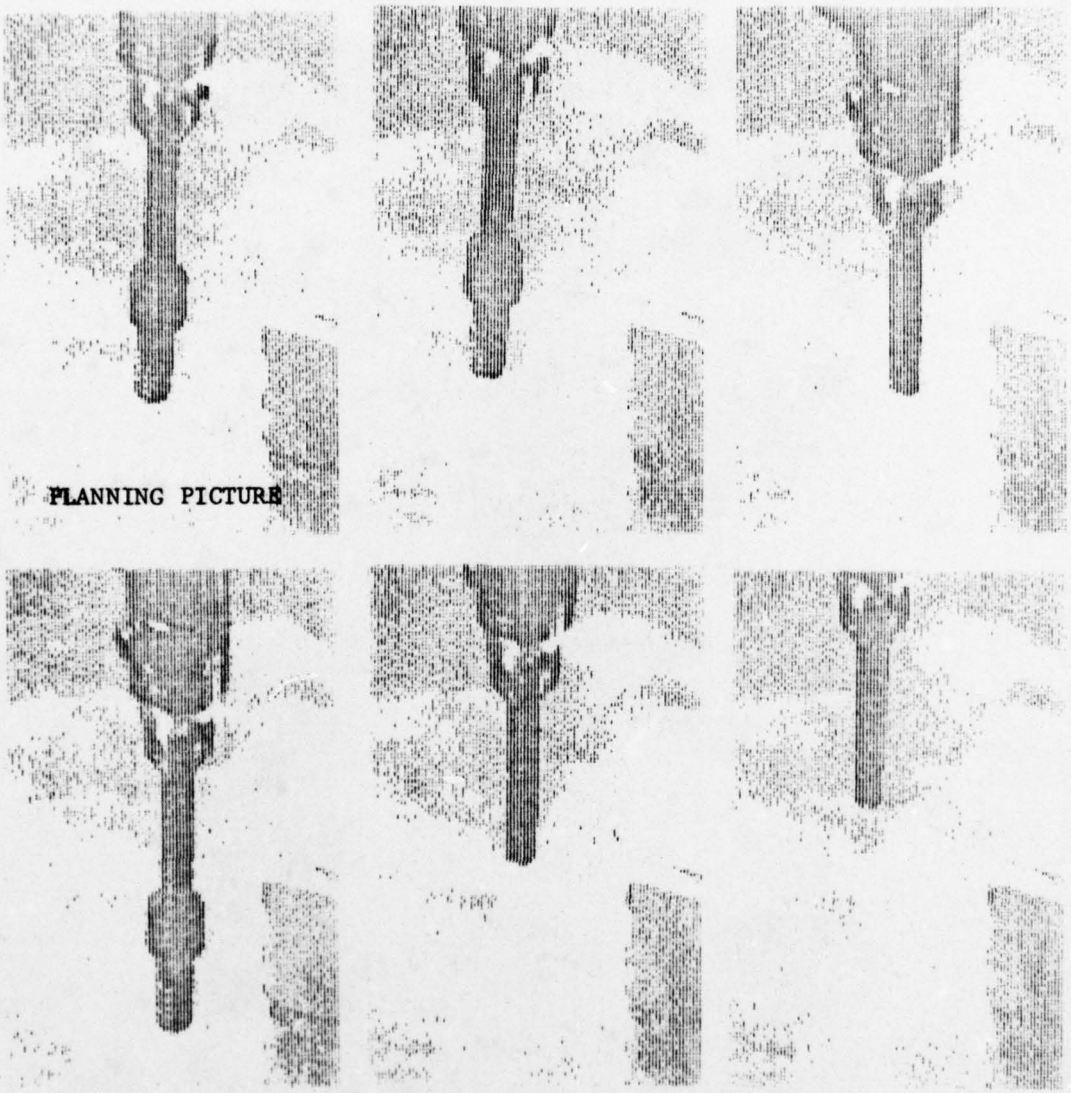


Figure 3.1.1

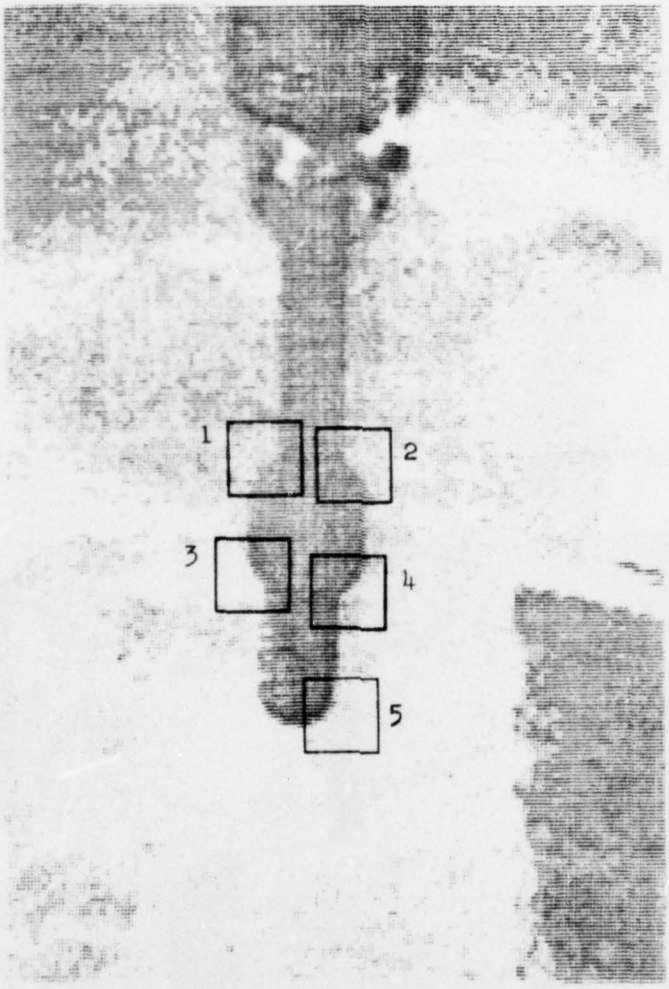


Figure 3.1.2

picture. Whenever operator 1 is applied to a sample picture it locates a best match with a certain value, which is a function of the correlation coefficient.

If operator 1 is applied to a sample picture in which the screw is missing, no portion of the picture will match operator 1 very well. Operator 1 will still locate a best match, but the correlation coefficient will be lower. Thus, operator 1, if reliable, will (1) match the correct piece of the screw, if the screw is there, and (2) match some other feature (with a lower correlation value) if the screw is not there. This performance difference is the basis for deciding whether the screw is there or not.

Figure 3.1.3 shows the results of applying operator 1 to ten different sample pictures where the screw is present. If the frequency of these values is *assumed* to follow a normal distribution, the corresponding distribution can be approximated from the experimental mean and standard deviation of these values. The fitted sample distribution is shown in figure 3.1.4. If operator 1 is applied to several pictures without the screw, the resulting values will form some other distribution. A table of ten such trials and the corresponding distribution (again assuming a normal distribution) are shown in figure 3.1.5. The two frequency functions are superimposed in figure 3.1.6. The assumption of normality is *not* necessary for the rules derived in this chapter. The assumption is convenient for displaying example distributions and it simplifies some of the planning-time computations, but it is not necessary for the execution-time formulas. Section 3.6 will outline the potential advantages of normally distributed values.

If operator 1 is applied to a test picture to determine whether the screw is there, the operator will find a best match with some value, say .93. Based solely upon operator 1, should the system say that the screw is there or not? In probabilistic terms, what is the probability that the screw is there, given that operator 1 has a value of .93? Denote that quantity as follows:

(3.1.1) P[<screw there> | <value of operator 1 is .93>].

Let

(3.1.2)
```
On  ≡ <the screw is on the end of the screwdriver>
Off ≡ <the screw is not on the end of the screwdriver>
v1  ≡ <operator 1 returns the value v1>
```

then the basic *a priori* probability diagram is

.80
.88
.77
.85
.83 Sample mean = .85
.89
.96 Sample standard deviation = .05
.86
.85
.82

Figure 3.1.3

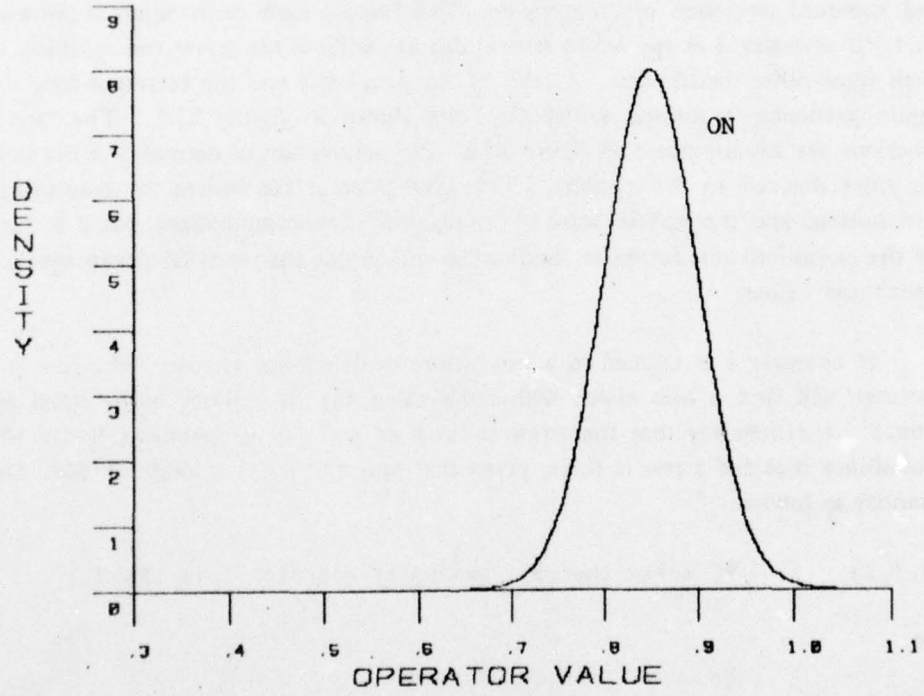


Figure 3.1.4

.60
.71
.50
.57
.67 Sample mean = .65
.70
.61 Sample standard deviation = .07
.76
.65
.71

Figure 3.1.5.a

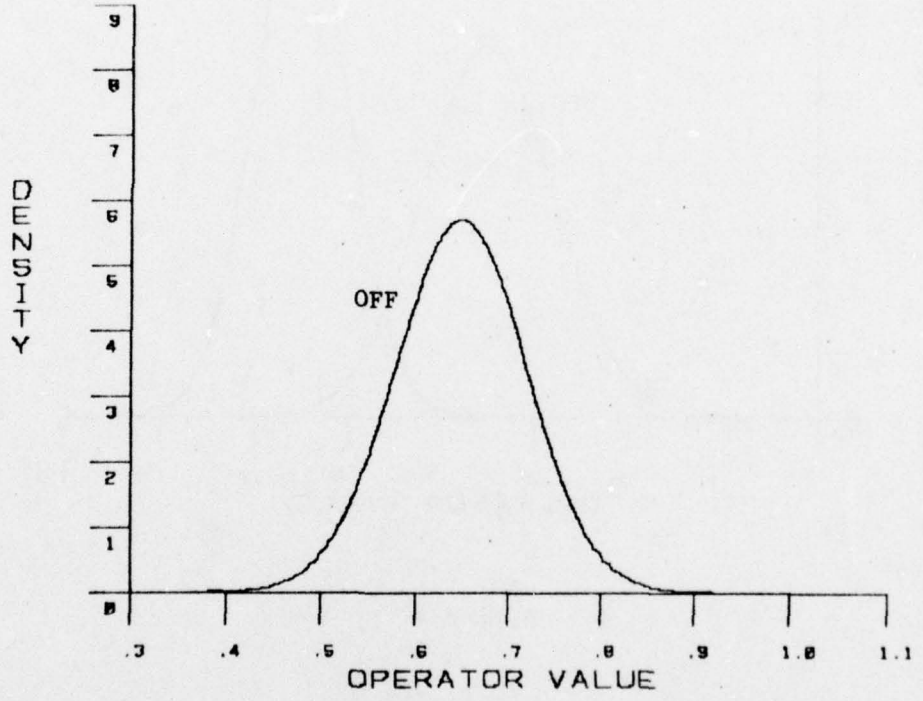


Figure 3.1.5

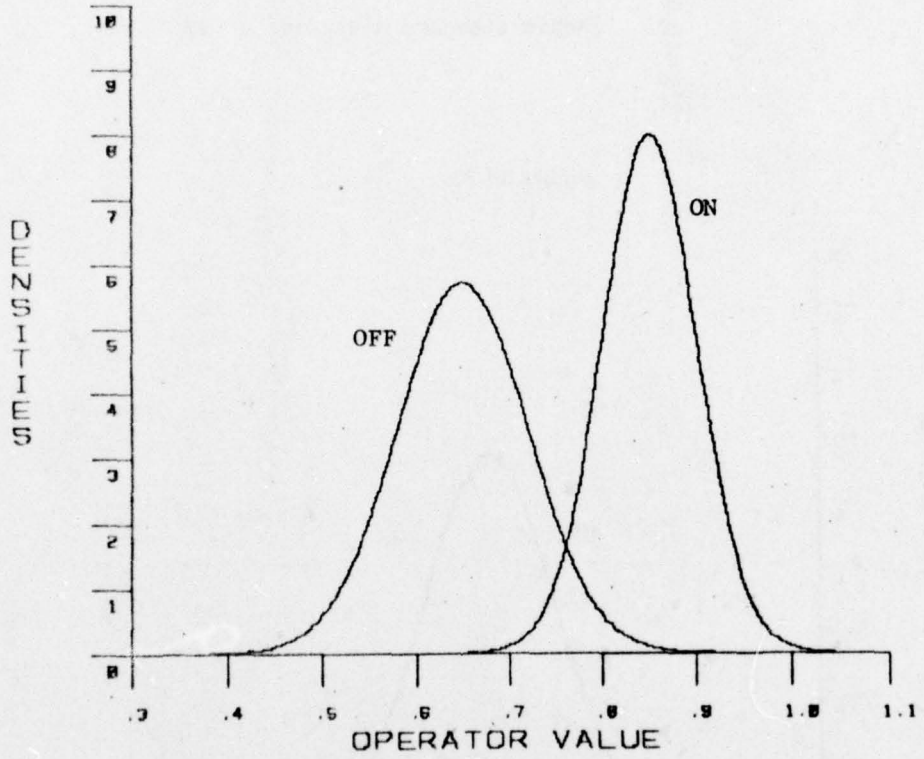
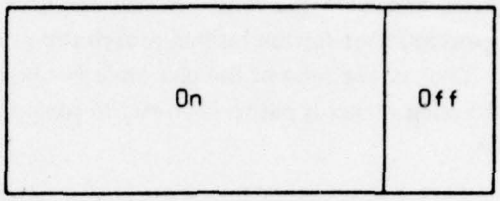
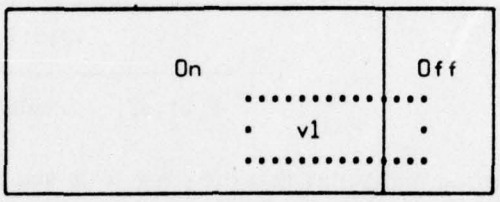


Figure 3.1.6



and the inclusion of P[v1] produces



Bayes' theorem (e.g., see [Hoel 71]) is a convenient way of combining this *a priori* probability information with the distribution information to produce an estimate for the probability that the screw is On. Bayes' theorem expresses the desired *a posteriori* probability in terms of the *a priori* and conditional probabilities as follows:

$$P[On|v1] = \frac{P[v1|On]*P[On]}{P[v1|On]*P[On] + P[v1|Off]*P[Off]} \tag{3.1.3}$$

or

$$P[On|v1] = \frac{1}{1 + \frac{P[v1|Off]*P[Off]}{P[v1|On]*P[On]}}. \tag{3.1.4}$$

These formulas state the desired probability in terms of probabilities that are often more readily computed. The *a priori* probabilities are based upon measured statistics or the experience of the assembly engineer. For example, if the screwdriver correctly acquires a screw nine-tenths of the time, P[On] is .9. The density functions shown in figure 3.1.6 can be used to compute the conditional probabilities, P[v1|Off] and P[v1|On]. Since the functions are density functions, the probability of the operator producing any one particular value is zero. But the probability of the operator producing a value within a certain range is the integral of the function over that range. Thus one way of estimating the above ratio for a specific value of the operator is to consider a small range about the value, compute the two

probabilities by integration, and form the ratio. Notice, however, that as the width of the region decreases, the approximations for the ratio approach the ratio of the two values of the density functions at X. That is, the ratio of the probabilities can be replaced by the ratio of the densities. This observation makes it particularly easy to compute the appropriate ratio for any value of the operator.

Bayes' theorem can be extended to combine the values of several operators:

$$(3.1.5) \qquad P[On|v1,v2,\ldots vN] = \frac{1}{1 + \dfrac{P[v1,v2,\ldots vN|Off]*P[Off]}{P[v1,v2,\ldots vN|On]*P[On]}}.$$

Let P[v1,On] represent the probability that the screw is On and that operator number one produces the value v1. Since

$$(3.1.6) \qquad P[v1|On] \equiv \frac{P[v1,On]}{P[On]}$$

and

$$(3.1.7) \qquad P[v1,v2|On] = \frac{P[v1,v2,On]}{P[On]} * \frac{P[On,v2]}{P[On,v2]} = P[v1|On,v2] * P[v2|On],$$

then, more generally, the conditional probabilities can be expanded into:

$$(3.1.8) \qquad P[v1,v2,\ldots vN|On] = P[v1|On,v2,v3,\ldots vN] * P[v2|On,v3,v4,\ldots vN] *$$

$$\ldots$$

$$* P[v(N-1)|On,vN] * P[vN|On].$$

If the vj's are assumed to be conditionally independent, that is,

$$(3.1.9) \qquad P[vj|On,vj+1,\ldots vN] = P[vj|On] \quad \text{(for all j's)}$$

then these probabilities reduce to

$$(3.1.10) \qquad P[v1,v2,\ldots vN|On] = P[v1|On] * P[v2|On] * \ldots * P[vN|On],$$

and (3.1.5) becomes

$$(3.1.11) \qquad P[On|v1,v2,\ldots vN] = \frac{1}{1 + \frac{P[Off]}{P[On]} * \prod_{i=1}^{N} \frac{P[vi|Off]}{P[vi|On]}}.$$

The conditional *independence* assumption stated in (3.1.10) is a major assumption. It makes it possible to reduce the interdependences significantly. But it also restricts the type of operators that can be used within a VV system that is based upon formulas such as (3.1.11). If the value of an operator is not approximately conditionally independent of the values of other operators, it can not be used.

Both (3.1.4) and (3.1.11) make it clear that the *contribution* of an operator is the value of the ratio:

$$(3.1.12) \qquad \frac{P[vi|Off]}{P[vi|On]}.$$

The contribution of an operator determines its influence on the estimate of P[On|v1,v2,...vN]. The inverse of ratio 3.1.12,

$$(3.1.13) \qquad \frac{P[vi|On]}{P[vi|Off]},$$

is known as the *likelihood ratio*. The logarithm of the likelihood ratio is also important, as the chapter on planning-time combination rules will show. The larger the likelihood ratio, the stronger the evidence that the screw is present. This formulation agrees with one's intuition in several ways. Consider figure 3.1.7 in which three values of the operator have been indicated: W, X, and Y. If the operator happens to produce the value W, the likelihood ratio is 1.0, and the estimate for the probability that the screw is there is unchanged. Any value to the left of W implies a likelihood ratio less than 1.0, and thus decreases the estimate of the probability that the screw is there. Both X and Y are to the left of W; both suggest that the screw is *not* there, but Y does so more strongly, as expected.

Not all values to the right of W will necessarily suggest that the screw is there. Consider figure 3.1.8. It emphasizes the difference between the two standard deviations so that it becomes clear that there can be a region to the right of W in which the likelihood

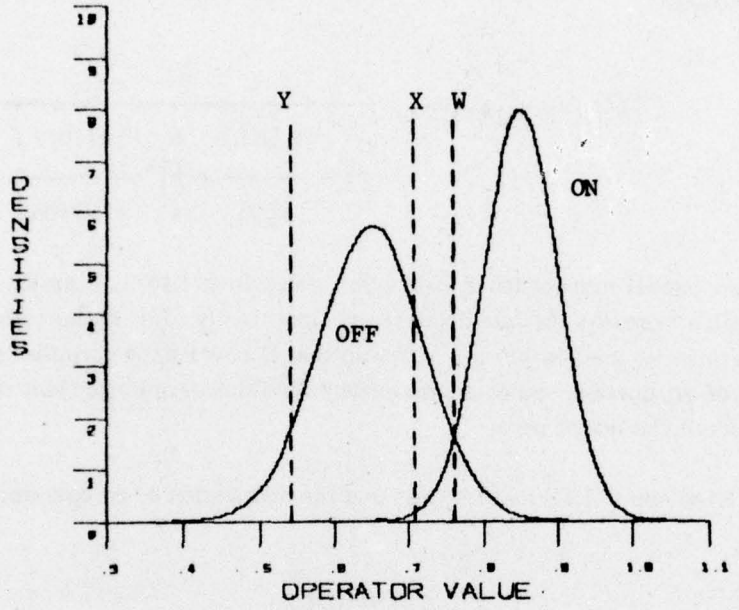


Figure 3.1.7

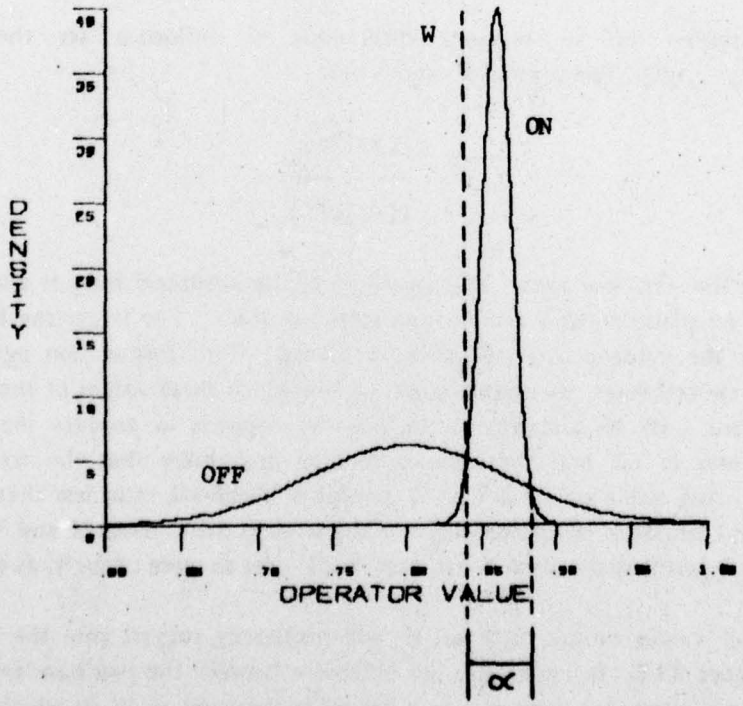


Figure 3.1.8

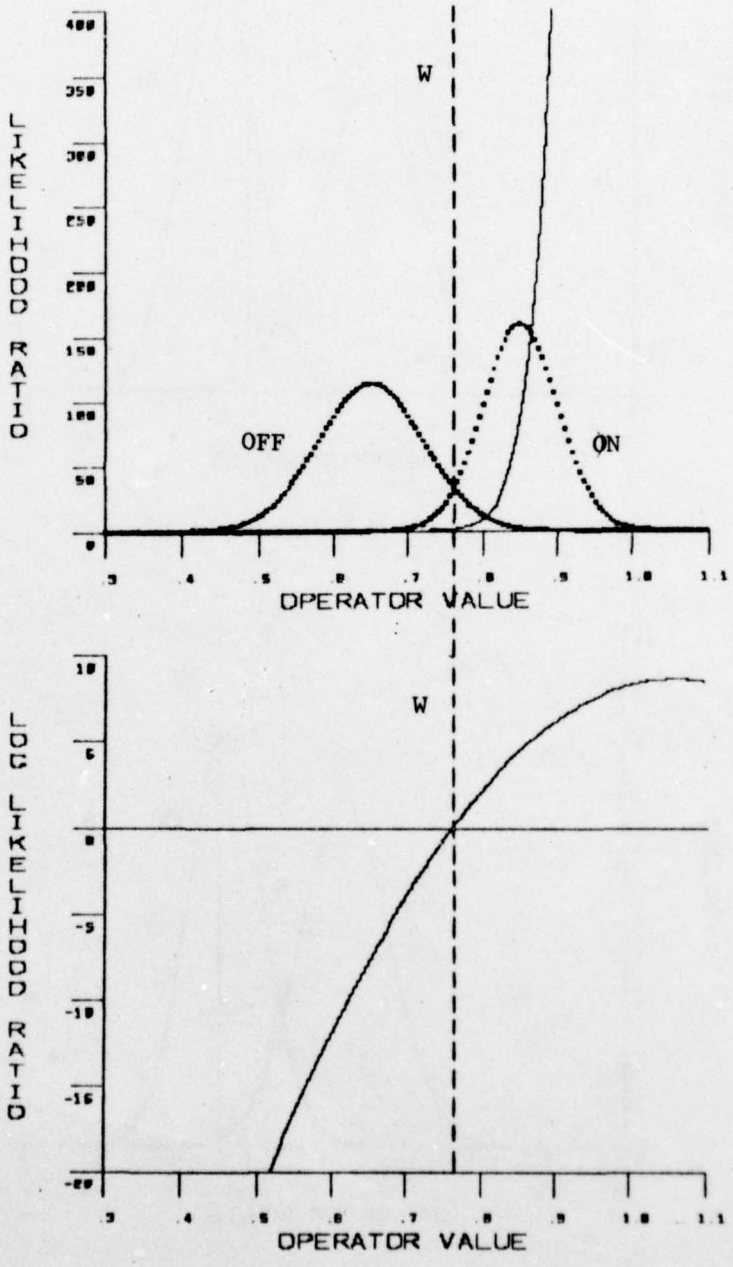

LIKELIHOOD RATIO
W
OFF
ON
OPERATOR VALUE
LOG LIKELIHOOD RATIO
W
OPERATOR VALUE

Figure 3.1.9

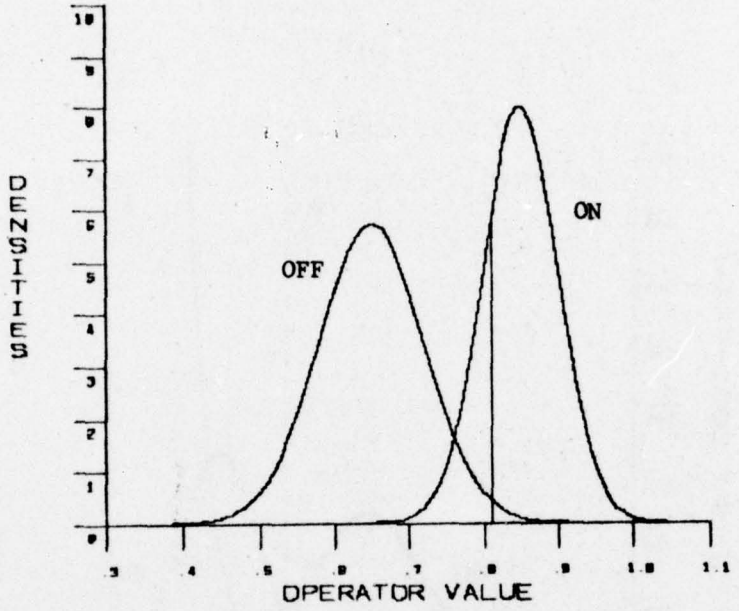


Figure 3.1.10

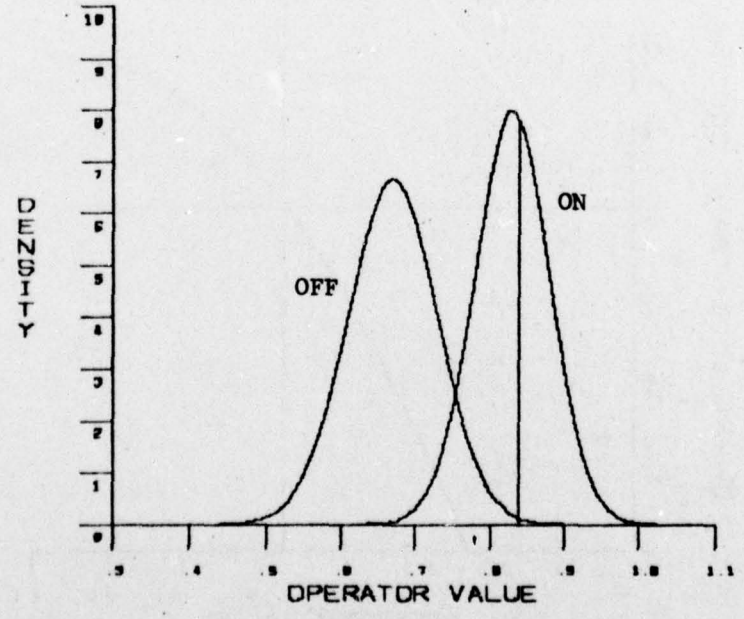


Figure 3.1.11

ratios are less than one. Only a small interval (labeled α) contains values that suggest that the screw is there. The likelihood ratios for every value in α are greater than one. All other values for the operator produce likelihood ratios less than one. Figure 3.1.9 shows the likelihood ratios and the log likelihood ratios associated with the distributions shown in figure 3.1.7.

The formulation of (3.1.11) is computationally convenient. For example, define

$$t(0) = \frac{P[Off]}{P[On]} \tag{3.1.14}$$

$$\text{and} \quad t(j) = \frac{P[vj|Off]}{P[vj|On]} * t(j-1) \qquad (\text{for } j > 0)$$

then

$$P[On|v1,\ldots vj] = \frac{1}{1 + t(j)}. \tag{3.1.15}$$

This set of formulas gives a straightforward way to incorporate the results of sequentially applied, conditionally independent operators *incrementally*. In fact, it is a powerful way to combine the value information of operators into a probability that an object is present.

For example, consider the screw checking task. If P[On] is .90, then t(0) is .11111. Assume that the density functions shown in figure 3.1.10 correspond to operator 1. If it returns a value of .810 (represented by the vertical line in the figure), then t(1) is .0080 and P[On|v1] is .9920. If the desired confidence is less than this amount, the program can stop applying operators and make a decision: the screw is present. If the desired confidence is .999, more information is needed. Consider the density functions for operator 2 shown in figure 3.1.11. If the second operator returns a value of .840, then t(2) is .000476 and P[On|v1,v2] is .9995, which is sufficient to make a decision.

Section 2
KNOWN ALTERNATIVES FOR A FEATURE

In the previous section, an operator was applied over some portion of a picture under the assumption that there are only two possible results: (1) the screw is present and the operator locates the appropriate piece of the screw or (2) the screw is not present and the operator locates some other best match. It was also assumed that the operator was applied over the whole region before returning the best match. In effect, these assumptions guarantee that the value returned by the operator belongs to one of the two density functions, On or Off. This result is pleasant if true, but there are several reasons why these assumptions might be unwarranted:

(1) There may be similar features in the same local area that sometimes appear better to the operator than the proper match. If a similar feature appears regularly enough in sample pictures so that the system can determine the corresponding density function, the feature will be called a *known alternative*. In that case the desired feature is itself considered to be one of the known alternatives. Every time a best match is found, the VV program must decide which alternative is being matched. If a similar feature occurs infrequently and unpredictably, it will be referred to as a *surprise*.

(2) Each application of the operator may be so expensive that it is prohibitive to scan it over the complete area in order to choose the best match. Instead, it has to be sequentially applied until some reasonably good match is found. If there are a few similar features in the local area, a reasonably good match may not be the best match, and hence the value produced by the operator may not belong to one of the two density functions.

(3) The measurements made by the operator may not immediately single out the best match. A correlation operator is a special type of operator; it only returns one value, so it is easy to determine which match is best. Other operators may return values along several scales. The best match is experimentally defined to be the one that produces values closest to the training values. For example, an edge operator may return both the distinctness of the edge and the contrast across the edge. If the desired line is a fuzzy line with a high contrast, it is not clear how to determine the best match. A metric has to be defined.

(4) The desired feature may not be in the portion of the picture scanned by the

operator. This problem may occur if the program has incorrectly restricted the tolerance region for a match, or if the feature has been obscured for some reason. The operator still returns the location and value for the best match it can find, but such a value does not belong to either of the densities; no conclusions can be drawn about P[On|v1,v2,...vN].

(5) Some global factor may change (e.g., one of the workstation's lights may be out) so that the feature appears quite different, even though it is in the correct area. In this case the values of the operator may be radically different than planned.

In this section the combination rules are extended to include *known alternatives*. Later sections will discuss the extensions necessary for surprises, multiple-valued operators, sparsely applied operators, and global changes.

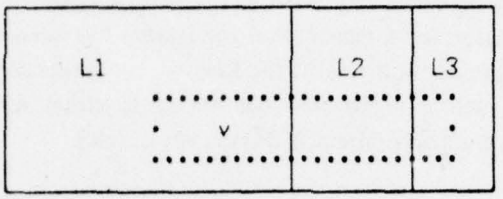
Consider the problem of correctly deciding which of three possible line segments an edge operator has located. There are several sources of information (orientation, fuzziness, contrast, etc.), but for the time being consider only one dimension (e.g., contrast). Assume that during the training session the system gathered enough statistics about the three lines to approximate the three density functions associated with their contrast values. If an edge is found with a certain contrast in an actual picture, which line is the operator on (assuming that there are only three possibilities) and what is the confidence associated with that decision? This question can be answered by computing three probabilities: the probability that the operator has located line 1, the probability that the operator has located line 2, and the probability that the operator has located line 3. Let

(3.2.1)
$$L1 \equiv \langle \text{operator 1 has located a point on line 1} \rangle$$
$$L2 \equiv \langle \text{operator 1 has located a point on line 2} \rangle$$
$$\text{and} \quad L3 \equiv \langle \text{operator 1 has located a point on line 3} \rangle.$$

Then Bayes' theorem states that

(3.2.2)
$$P[L1|v] = \frac{1}{1 + \dfrac{P[v|\neg L1]*P[\neg L1]}{P[v|L1]*P[L1]}}.$$

Consider the following diagram



Since

(3.2.3) $\neg L1 = L2 \oplus L3$ { $\oplus$ stands for exclusive OR },

(3.2.4) $P[v|\neg L1] = P[v|L2]*P[L2|\neg L1] + P[v|L3]*P[L3|\neg L1]$.

Formula 3.2.2 reduces to

$$(3.2.5) \qquad P[L1|v] = \frac{1}{1 + \frac{P[v|L2]*P[L2,\neg L1]}{P[v|L1]*P[L1]} + \frac{P[v|L3]*P[L3,\neg L1]}{P[v|L1]*P[L1]}}.$$

Since L2 and L3 form ¬L1, each is contained in ¬L1. Therefore, (3.2.5) can be further reduced to

$$(3.2.6) \qquad P[L1|v] = \frac{1}{1 + \frac{P[v|L2]*P[L2]}{P[v|L1]*P[L1]} + \frac{P[v|L3]*P[L3]}{P[v|L1]*P[L1]}}.$$

When there are N known alternatives formula 3.2.6 can be generalized to

$$(3.2.7) \qquad P[Lj|v] = \frac{1}{1 + \sum_{i \neq j} \frac{P[v|Li]*P[Li]}{P[v|Lj]*P[Lj]}}. \qquad (\text{for } 1 \leq j \leq N)$$

This formula is convenient because it states the desired probability in terms of *a priori* probabilities and likelihood ratios. Given the value of an operator that has several known alternatives, the probability of each alternative is computed, and the alternative with the largest probability is the best match.

This derivation is also useful for an *inspection* task in which there are two or three known alternatives when the object is there and two or three known alternatives when the object is not there. In this case the program is less concerned about which alternative is the best match than about the overall probability that the object is there. A derivation similar to the one used above produces the formula needed in this situation. Let f1, f2, ... fM be the known alternatives that might be matched when the object is there and let g1, g2, ... gN be the alternatives that are possible when the object is not there. Bayes' theorem states:

$$P[On|v] = \frac{1}{1 + \frac{P[v|Off]*P[Off]}{P[v|On]*P[On]}} \qquad (3.2.8)$$

By assumption

$$P[On] = P[f1] + P[f2] + \ldots + P[fM] \qquad (3.2.9)$$

and

$$P[Off] = P[g1] + P[g2] + \ldots + P[gN]. \qquad (3.2.10)$$

Formula 3.2.8 can be expanded into

(3.2.11)

$$P[On|v] = \frac{1}{1 + \frac{P[v|g1]*P[g1] + P[v|g2]*P[g2] + \ldots + P[v|gM]*P[gN]}{P[v|f1]*P[f1] + P[v|f2]*P[f2] + \ldots + P[v|fM]*P[fM]}}$$

or

$$P[On|v] = \frac{1}{1 + \frac{\sum_{1 \le i \le N} P[v|gi]*P[gi]}{\sum_{1 \le i \le M} P[v|fi]*P[fi]}} \qquad (3.2.12)$$

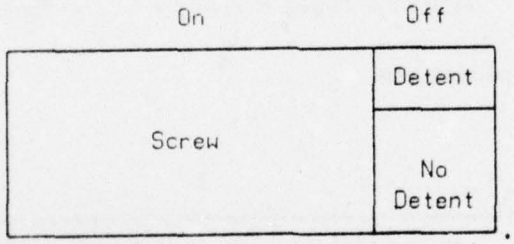
This formula gathers all of the evidence for and against the proposition On and forms a ratio between them. To use this formula requires a great deal of knowledge about what can

be expected in a runtime picture. In particular, this knowledge includes the set of possible alternatives, their values, and their *a priori* probabilities. Within the context of programmable assembly this information is often available because the environment is highly constrained and a program has the opportunity to watch several examples of the assembly.

Consider the application of formula 3.2.12 to the screw checking example. When the screw is not on the end, one of the operators (say operator 1 that tries to locate the tip of the screw) may return a different set of values depending upon whether or not the detent at the end of the screwdriver is showing. (The detent is the spring-loaded ballbearing that holds the screw on the end.) It appears in the picture only if the screwdriver is oriented in such a way that aims the detent at the camera. Thus, there are two known alternatives for operator 1 when the screw is not on the end: (1) the detent is showing and (2) the detent is not showing. When the screw is present, there is still only one known alternative. Thus, the *a priori* probabilities for operator 1 are:

(3.2.13)
$$P[On] = P[Screw]$$
$$P[Off] = P[Detent] + P[No\ Detent],$$

which can be diagramed as follows:



This diagram is just for operator 1. The other operators may not be affected by the detent.

If operator 1 is the first operator to be applied, formula 3.2.12 simplifies to:

(3.2.14)
$$P[On|v1] = \frac{1}{1 + \frac{P[v1|Detent]*P[Detent] + P[v1|No\ Detent]*P[No\ Detent]}{P[v1|Screw]*P[Screw]}}.$$

Assume that

(3.2.15)

P[Screw] = .90
P[Detent] = .03
P[No Detent] = .07

and that the density functions shown in figure 3.2.1 are the functions for operator 1. If operator 1 returns a value of .81, then P[On|v1] is .9910. Figure 3.2.2 shows the three known alternatives for operator 1 and three typical values produced by operator 1. Each column of the table contains P[Tip|v], P[No Detent|v], and P[Detent|v]. Given a value, v1, for operator 1, the known alternative with the largest value of P[<known alternative> | v1] is the best match for that value. For example, given the value .47, the best match is the Detent. That is, based upon the *a priori* probabilities and the training information, a value of .47 for operator 1 implies that the most likely feature being matched is the detent in the end of the screwdriver. The second best choice is the end of the screwdriver without the detent.

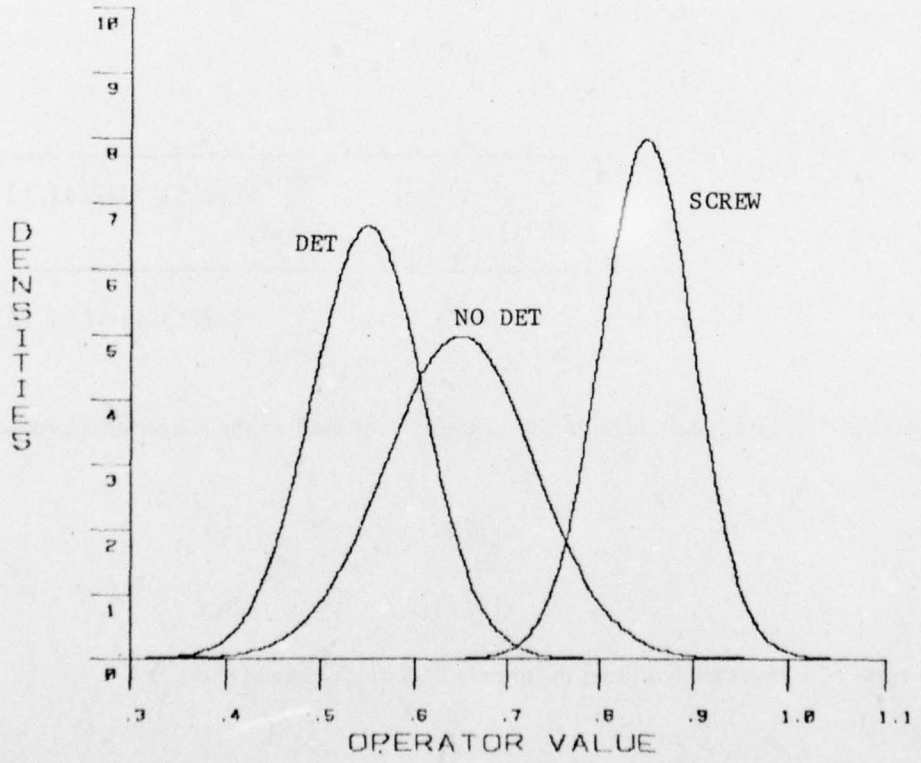


Figure 3.2.1

		VALUE OF OPERATOR 1		
		.81	.63	.47
KNOWN ALTERNATIVE	Screw	.3910	.0011	.0000
	No Detent	.0090	.8040	.2530
	Detent	.0000	.1949	.7470

Figure 3.2.2

Formula 3.2.12 can be easily extended to incorporate the results of several operators, all of which may have known alternatives. Assume that there are K operators. Let fj,1; fj,2; ... ;fj,Nj be the Nj known alternatives for the jth operator when the object is present. Let gj,1; gj,2; ...;gj,Mj be the Mj known alternatives for the jth operator *when the object is not present.* Then

(3.2.16)

$$P[On|v1,v2,\ldots vK] = \frac{1}{1 + \frac{P[On]^{(K-1)}}{P[Off]^{(K-1)}} * \prod_{j=1}^{K} \frac{\sum_{1\le i\le Nj} P[vj|gj,i]*P[gj,i]}{\sum_{1\le i\le Mj} P[vj|fj,i]*P[fj,i]}}.$$

The exponent (K-1) appears because the expression for each of the K operators produces a factor of

(3.2.17)
$$\frac{P[On]}{P[Off]},$$

and the ratio of *a priori* probabilities in formula 3.2.8 cancels one of them.

Section 3
SURPRISES

The main assumption of the last section is that all of the alternatives are known and characterized in advance. Sometimes, however, operators match unknown features and return unusual values. Such unknown and unexplained matches will be referred to as *surprises*. The values produced by surprises can not be accounted for by the usual density functions.

There are several possible causes for an unusual match (some global change, the feature is not present, or a surprise), so the values produced for such a match should *not* be used to alter the overall confidence estimate. The values may contribute to other considerations (such as a global error decision), but they should not be blindly cranked through the combination rules.

There are two ways to deal with unusual values: (1) filter out particularly bad values and (2) scale down the potential contribution (in the probability computations) of any operator that is known to find surprises. The first method involves a check on each value produced by an operator to make sure that it is reasonable for at least one of the known alternatives. For example, if a value is not within three standard deviations of the mean of at least one known alternative, classify it as an *unusual value*. This method is generally only applicable when some unexpected event occurs, such as one of the lights burns out at the workstation so that the pictures of the scene are significantly different than expected.

The second method is to lower the possible contribution of unreliable operators. This method is based upon a simple rule: an operator that finds surprises should not be trusted as much as one that doesn't. The assumption that all of the alternatives are known has been expressed in formula 3.2.9.

(3.3.1) $$P[On] = P[f1] + P[f2] + \ldots P[fN].$$

If the operator occasionally locates surprises, a better model is

(3.3.2) $$P[On] = P[f1] + P[f2] + \ldots P[fN] + P[s]$$

where P[s] is the *a priori* probability of finding a surprise. To reflect this model in the probability computations requires a density function to be associated with the surprises. What should the form of this density be? If surprises can randomly produce any value for the operator, one reasonable assumption is that the surprise density has a rectangular distribution. If the filtering method (i.e., method 1 mentioned above) is applied, the range of

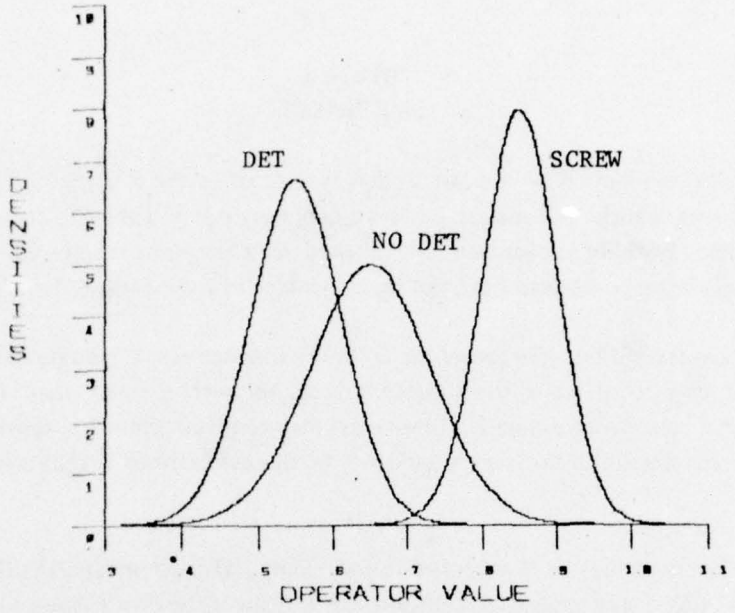


Figure 3.3.1.a

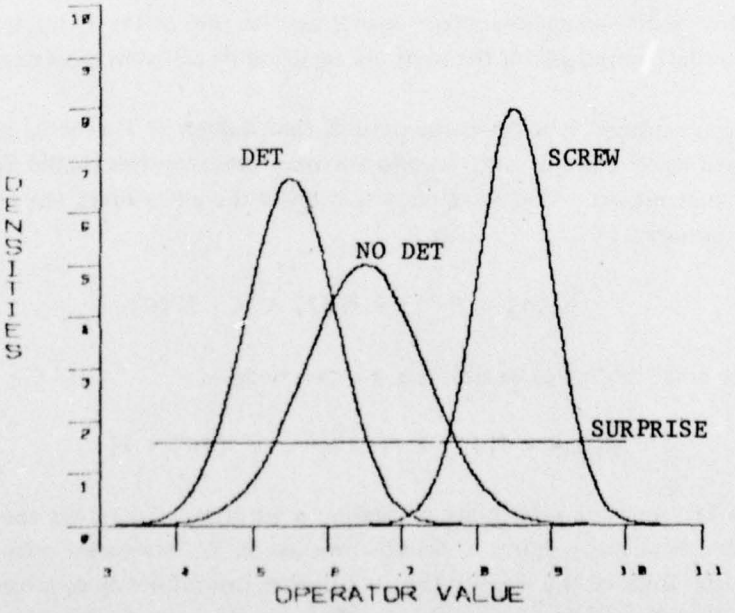


Figure 3.3.1.b

the rectangular distribution can be restricted to the interval between the smallest reasonable value for the operator and the largest reasonable value. *Figure 3.3.1.a shows three density functions, one for the case in which the screw is present and two known alternatives when the screw is not present. If the operator occasionally locates surprises, a rectangular density function is added, as shown in figure 3.3.1.b.*

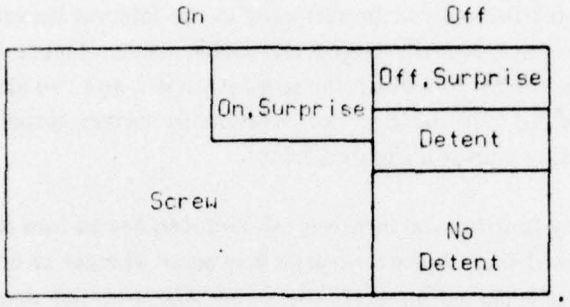
The density function for surprises can be incorporated into the confidence computation in a straightforward way. Since a surprise may occur whether or not the object is present, the new possibility is included in both the numerator and the denominator. However, the probability of a surprise may be different when the screw is present than when it is not. That is, P[Surprise|On] may be different from P[Surprise|Off]. In order to provide for this possibility, different quantities are included in the numerator and denominator. If s represents a surprise, formula 3.2.12 can be restated as:

$$(3.3.3) \qquad P[On|v] = \frac{1}{1 + \dfrac{P[v|s,Off]*P[s,Off] + \sum_{i=1}^{N} P[v|gi]*P[gi]}{P[v|s,On]*P[s,On] + \sum_{i=1}^{M} P[v|fi]*P[fi]}}.$$

The additional density function restricts the contribution of the suspect operator. The operator can not be as strongly for or against the proposition On as it could be when all of the alternatives were known. For example, consider operator 1 mentioned in the last section. The *a priori* probabilities are

$$(3.3.4) \qquad \begin{aligned} P[On] &= P[On,Surprise] + P[Screw] \\ P[Off] &= P[Off,Surprise] + P[Detent] + P[No\ Detent] \end{aligned}$$

and the corresponding diagram is



Formula 3.3.3 reduces to

(3.3.5) $P[On|vX] =$

$$\frac{1}{1 + \dfrac{P[vX|Off,Surp]*P[Off,Surp] + P[vX|Det]*P[Det] + P[vX|No\ Det]*P[No\ Det]}{P[vX|On,Surp]*P[On,Surp] + P[vX|Screw]*P[Screw]}}.$$

Assume that

(3.3.6)
$$P[Screw] = .88$$
$$P[Det] = .02$$
$$P[No\ Det] = .06$$
$$P[Off,Surp] = .02.$$
$$\text{and } P[On,Surp] = .02.$$

Since P[Off,Surp], P[On,Surp], P[v1|On,Surp], and P[v1|Off,Surp] are constants, formula 3.3.5 reduces to

(3.3.7)

$$P[On|v1] = \frac{1}{1 + \dfrac{.0318 + P[v1|Det]*P[Det] + P[v1|No\ Det]*P[No\ Det]}{.0318 + P[v1|Screw]*P[Screw]}}.$$

The maximum value of P[On|v1] is achieved when operator 1 returns a value that minimizes both P[v1|Det] and P[v1|No Det]. At that point P[On|v1] is .7754. This value is significantly less than the maximum value of 1.0, which is possible if the surprises are not incorporated into the formula. So incorporating surprises into the formulas for operators

known to produce surprising values reduces the operators' contributions toward the overall probabilities, thereby implementing method 2 above.

If the density functions shown in figure 3.3.1.b are the functions for operator 1, and operator 1 returns a value of .63, the conditional probabilities associated with the alternatives are:

(3.3.8)

$$P[Tip|.63] = .0011$$
$$P[Det|.63] = .1338$$
$$P[No\ Det|.63] = .7096$$
$$\text{and } P[Surprise|.63] = .1556.$$

The best alternative is No Detent and the overall probability that the screw is on is .0789.

The incorporation of surprises also means that sometimes the best match may be a *surprise*. For example, if operator 1 happens to return a value of .40, P[Surprise|.40]=.8869, which means that the best match is the surprise.

Formula 3.3.3 can be extended to combine the results of several operators, each of which may have known alternatives and/or surprises. Let $f_{j,0}$ be the surprise associated with the jth operator when the object is there and let $g_{j,0}$ be the surprise for the jth operator when the object is not there. Then the formula can be written as:

(3.3.9)

$$P[On|v1,v2,\ldots vK] = \frac{1}{1 + \frac{P[Off]}{P[On]} * \prod_{j=1}^{K} \left(\frac{P[On]}{P[Off]} * \frac{\sum_{0\le i\le Nj} P[vj|gj,i]*P[gj,i]}{\sum_{0\le i\le Mj} P[vj|fj,i]*P[fj,i]} \right)},$$

This extension to include surprises means that there are three possible outcomes whenever an operator is applied: (1) the value is outside the reasonable range, (2) the value is reasonable, but the best match is a surprise, or (3) the value is reasonable and the best match is a known alternative. The higher-level interpretation, if any, of the unusual values and surprises will be discussed in a later chapter.

Section 4
MULTIPLE-VALUED OPERATORS

Some operators return more than one value; the description of what they have found contains values along several scales. For example, a texture operator may describe a local region in terms of its size, density, and periodicity. It has already been mentioned that edge operators often return two or three values. When dealing with such operators one wants to combine all of the available information into one probability that the object is present, or to determine the best alternative. Again Bayesian probability provides a way to make this combination. Consider an inspection task and one operator that returns M values, x1, x2, ... and xM. Then the standard Bayesian formula is

$$(3.4.1) \qquad P[On|x1,x2,...xM] = \frac{1}{1 + \frac{P[x1,x2,...xM \mid Off]}{P[x1,x2,...xM \mid On]} * \frac{P[Off]}{P[On]}}.$$

If the values happen to be conditionally independent of each other, the usual reduction yields

$$(3.4.2) \qquad P[On|x1,x2,...xM] = \frac{1}{1 + \frac{P[Off]}{P[On]} * \prod_{i=1}^{M} \frac{P[xi|Off]}{P[xi|On]}}.$$

These formulas can be extended to include several operators, each of which may return several values. Assume that there are N operators and each operator returns Mj values ($Mj \geq 1$). Let xj,1; xj,2; ... xj,Mj be the Mj values returned by the jth operator. If the values for one operator are interdependent, but the values of separate operators are conditionally independent, then

(3.4.3)

$$P[On \mid (x1,1;\ x1,2;\ \ldots\ x1,M1),\ \ldots,\ (xN,1;\ xN,2;\ \ldots\ xN,MN)\ ] =$$

$$\frac{1}{1 + \dfrac{P[Off]}{P[On]} * \prod_{j=1}^{N} \dfrac{P[\ (xj,1;xj,2;\ldots;xj,Mj) \mid Off]}{P[\ (xj,1;xj,2;\ldots;xj,Mj) \mid On]}}.$$

If all of the values are conditionally independent of each other this formula collapses back to the previous formula (with a suitable renumbering of the x's).

Formula 3.4.3 can be further extended to include operators that have several known alternatives and even surprises. Assume that the values for one operator are interdependent, but that the values of separate operators are conditionally independent. Let there be K operators. Let the jth operator have Mj known alternatives when the object is there, and Nj known alternatives when the object is not there. Let M0 and N0 represent the surprises. Assume that the jth operator returns Rj values as a description of what it finds. Then the appropriate formula is:

(3.4.4)

$$P[On \mid (x1,1;x1,2;\ldots;x1,R1),\ \ldots,\ (xN,1;xN,2;\ldots;xN,RN)] =$$

$$\frac{1}{1 + \dfrac{P[Off]}{P[On]} * \prod_{j=1}^{K} \left(\dfrac{P[On]}{P[Off]} * \dfrac{\sum_{0 \le i \le Nj} P[(xi,1;xi,2;\ldots;xi,Ri) \mid gj,i] * P[gj,i]}{\sum_{0 \le i \le Mj} P[(xi,1;xi,2;\ldots;xi,Ri) \mid fj,i] * P[fj,i]} \right)}.$$

To use operators that return several interdependent values requires enough information to approximate the multi-dimensional density functions. Once this has been done, the ratio of density values can be used in place of the ratio of probabilities, just as in the one-dimensional case.

Since the expression (xj,1; xj,2; ... xj,Rj) can be validly substituted for vj in any of the derivations that follow, the remaining derivations will only be concerned with single-valued operators. The formulas apply to multiple-valued operators, but for notational

simplicity they will not be stated in their full generality.

Section 5
POSITION INFORMATION

The local value information produced by an operator is important, but the relative structure of the matches is crucial in verification vision. This section describes two methods that incorporate structural information into the execution-time formulas for inspection. One method uses a set of matches to determine the most consistent location for the object and then decides how likely that location is in light of the initial task constraints and the training information. The second method simply determines the number of reasonably consistent matches, given the fact that the features are part of a rigid, three-dimensional object. A major drawback of the first method is that the amount of computation goes up exponentially as the number of operators increases. The second method is a simple heuristic to avoid this large amount of computation, but it does not take full advantage of the available information.

Figure 3.5.1.a shows the positions of four typical features in a planning picture. Assume that the task is to determine the change from the planning picture to the test picture and that the change is mainly an X-Y shift. If the four operators are applied to a test picture and they locate their best matches at the positions shown in figure 3.5.1.b, the relative structure of the matches appears to be correct. A least-squares fitting routine (or some other fitting routine) can be used to produce an estimate for the shift and an estimate for the goodness of fit. In this example, the residual errors are quite small (as shown in figure 3.5.1.c). Since this is a good fit, one would say that the matches are *structurally consistent*. However, if the four matches are found at the positions shown in figure 3.5.1.d, the best fit would contain large errors (see figure 3.5.1.e). In this case one would probably be suspicious of at least one of the matches.

The implication is that the residual errors are a function of the structural consistency of the set of matches. The less consistent the matches, the larger the errors. The sum of the squares of the errors is commonly used to measure this type of consistency. It is a convenient measure because there are well-known techniques for minimizing it. It is also appealing because the distribution of the sum of the squares of the errors is known to be a Chi-square distribution if the errors are normally distributed [Graybill 61]. Since measurement errors are known to be normally distributed for a large number of situations, the use of least-squares techniques looks quite promising.

The theorem that specifies the distribution of the sum of the squares of the errors can be stated as follows:

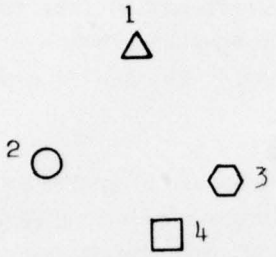


Figure 3.5.1.a

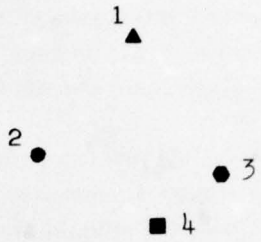


Figure 3.5.1.b

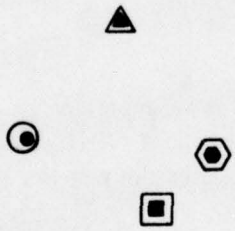


Figure 3.5.1.c

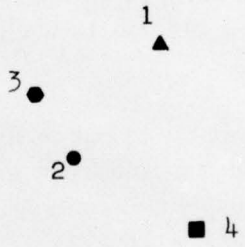


Figure 3.5.1.d

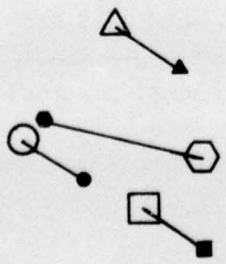


Figure 3.5.1.e

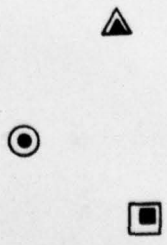


Figure 3.5.1.f

THEOREM: If there are N linear equations relating the actual matching positions with the planned positions and if there are R parameters to be adjusted in the transformation from the planned to actual positions, the sum of the squares of the errors (for normally distributed errors) forms a Chi-square distribution with (N-R) degrees of freedom.

One implication of this theorem is that a Chi-square test can be applied to a particular sum of squares to determine whether it represents a consistent transformation between the planned and actual positions. If the test indicates that the set of matches is *not* consistent, it is possible to determine which match is the least consistent. For example, in figure 3.5.1.e, the hexagon is the least consistent feature. The least consistent match can be temporarily left out of the solution and another least-squares fit can be computed. If the new fit is significantly better than the previous one, the least consistent match can be permanently removed from the set of matches. This culling of *bad* matches can continue until a consistent set of matches has been found. Thus, another measure of the consistency of a set of matches is the percentage of matches deemed consistent by this culling procedure. Figure 3.5.1.f shows the best fit after the least consistent match has been discarded. *The remaining three matches are structurally* consistent.

As expected, the concept of *structural consistency* is an important aspect of verification. But how should it be integrated with the value information? It is possible to extend the conditional probability formulas to include the positions of the matches in addition to the values of the matches. Let

(3.5.1) pi ≡ <the position of operator i's match>,

then Bayes' theorem states:

$$(3.5.2)\quad P[On|v1,\ldots vN,p1,\ldots pN] = \frac{1}{1 + \frac{P[v1,\ldots vN,p1\ldots pN|Off]}{P[v1,\ldots vN,p1\ldots pN|On]} * \frac{P[Off]}{P[On]}}.$$

If the vi's are assumed to be conditionally independent of the pi's (and each other), formula 3.5.2 reduces to:

(3.5.3)

$$P[On|v1,\ldots vN,p1,\ldots pN] = \frac{1}{1 + \frac{P[p1\ldots pN|Off]}{P[p1\ldots pN|On]} * \frac{P[Off]}{P[On]} * \prod_{i=1}^{N} \frac{P[vi|Off]}{P[vi|On]}}.$$

The assumption that the vi's are conditionally independent of the pi's means that the value of an operator is independent of the location of the match. That is, if an operator locates the same feature at different positions in different test pictures, it is expected to produce the same value. In programmable assembly this is generally a reasonable assumption except when different positions consistently produce different lighting conditions. For example, if a shadow happens to fall on a feature when the object is oriented in a certain way, the value of the operator that tries to find that feature will depend upon the orientation of the object.

The assumption one does *not* want to make is that the pi's are conditionally independent of each other. Such an assumption would completely ignore structural consistency, which is precisely what the mathematics is intended to capture. But what is the value of

(3.5.4)
$$\frac{P[p1,p2,\ldots pN \mid Off]}{P[p1,p2,\ldots pN \mid On]} ?$$

First, consider a simpler version of (3.5.4) that is only concerned with the positions of two operators:

(3.5.5)
$$\frac{P[p1,p2 \mid Off]}{P[p1,p2 \mid On]}.$$

The probability P[p1,p2|Off] is the probability that operator 1 will find its match at the position p1 *and* that operator 2 will find its match at the position p2. Ratio 3.5.5 can be rewritten as

(3.5.6)
$$\frac{P[p1,p2,Off]*P[On]}{P[p1,p2,On]*P[Off]}.$$

If P[Off] could be expressed simply as:

(3.5.7) $$P[Off] = P[X] + P[Y] + P[Z],$$

then P[p1,Off] could be expressed as

(3.5.8) $$P[p1,Off] = P[p1,X] + P[p1,Y] + P[p1,Z]$$

and P[p1,p2,Off] could be expressed as

(3.5.9) $$P[p1,p2,Off] = P[p1,p2,X] + P[p1,p2,Y] + P[p1,p2,Z].$$

However, P[Off] is more complicated than that. It is a sum of several conjunctions. Assume that the first operator has m1 known alternatives when the screw is Off and n1 known alternatives when the screw is On. Let g1,1; g1,2; ... g1,m1 be the m1 known alternatives when the screw is Off and let f1,1; f1,2; ... f1,n1 be the n1 known alternatives when the screw is On. Similarly let g2,1; g2,2; ... g2,m2 and f2,1; f2,2; ... f2,n2 be the known alternatives for the second operator. Then P[Off] and P[On] can be expressed as follows:

(3.5.10) $$P[Off] = \sum_{j=1}^{m1} \sum_{k=1}^{m2} P[g1,j;\ g2,k].$$

Given this expression for the probability P[Off], the analogous formula to (3.5.9) is

(3.5.11) $$P[p1,p2,Off] = \sum_{j=1}^{m1} \sum_{k=1}^{m2} P[p1;p2;g1,j;g2,k].$$

Then ratio 3.5.6 can be rewritten as

(3.5.12) $$\frac{P[On]}{P[Off]} * \frac{\sum_{j=1}^{m1} \sum_{k=1}^{m2} P[p1;\ p2;\ g1,j;\ g2,k]}{\sum_{j=1}^{n1} \sum_{k=1}^{n2} P[p1;\ p2;\ f1,j;\ f2,k]},$$

Thus, ratio 3.5.4 has been reduced to the evaluation of several expressions of the form P[pj;pk;f1,j;f2,k], which represents the probability that operator 1 has found known alternative f1,j at position pj and operator 2 has found known alternative f2,k at position pk.

In order to see how estimates for probabilities such as P[p1;p2;f1,j;f2,k] might be computed, consider figure 3.5.2, which is a picture of a screw taken during the standard screw checking task. In that task the location of the tip of the screw is assumed to be known within plus or minus one quarter inch along each axis and the orientation of the screw is assumed to be within plus or minus fifteen degrees of vertical. Figure 3.5.2 shows two features on the screw. If two operators trying to locate these features happen to find their best matches at the locations shown in figure 3.5.3, what is the probability that they have located the correct features? The chance is probably small because it would require the screw to be rotated 140 degrees from vertical, which is highly unlikely since the initial uncertainty is only plus or minus fifteen degrees. If the two matching positions are at the locations shown in figure 3.5.4, the probability that the matches are correct is significantly higher since the implied displacements and rotation are within the expected ranges. The important idea is that the estimate of the probability P[p1;p2;f1,j;f2,k] can be based upon the likelihood that the object would be at the location implied by the set of matches.

The likelihood that an object is at a specific location can be expressed in terms of the parameters that describe the location. For example, consider a task in which the uncertainties are pseudo-planar (i.e., the unknown parameters are X and Y displacements, dx and dy; a rotation change, dt; and a scale change, ds). Statistics can be gathered at training time that describe the expected density functions for these parameters (see figure 3.5.5). If two matches imply that dx=.1, dy=-.2, dt=-2.3, and ds=1.02, the probability P[p1;p2;f1,j;f2,k] can be expressed as

(3.5.13) P[p1; p2; f1,j; f2,k] = P[dx=.1;dy=-.2;dt=-2.3;ds=1.02].

To a first approximation the parameters can be treated independently. Thus,

(3.5.14) P[p1;p2;f1,j;f2,k] = P[dx=.1]*P[dy=-.2]*P[dt=-2.3]*P[ds=1.02].

The value of each of the component probabilities can be determined by integrating the appropriate density function over a small interval centered about the nominal value.

This step completes the demonstration that the position information can be incorporated into the conditional probability formulas and that there are reasonable techniques for estimating the necessary probabilities. The final formula for two operators is:

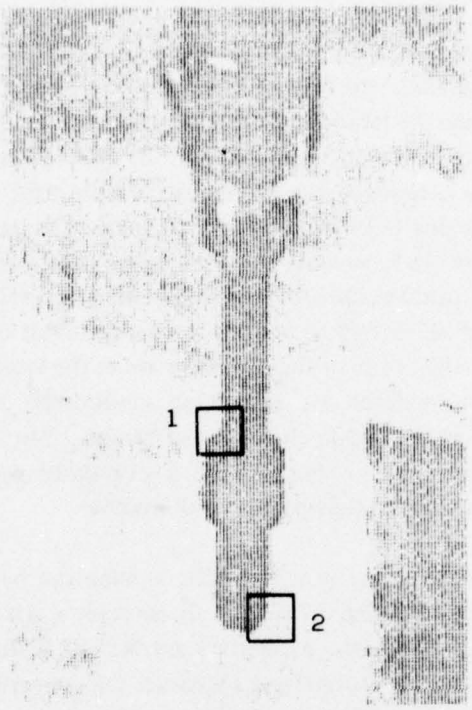


Figure 3.5.2

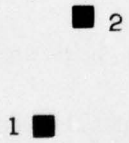


Figure 3.5.3

Figure 3.5.4

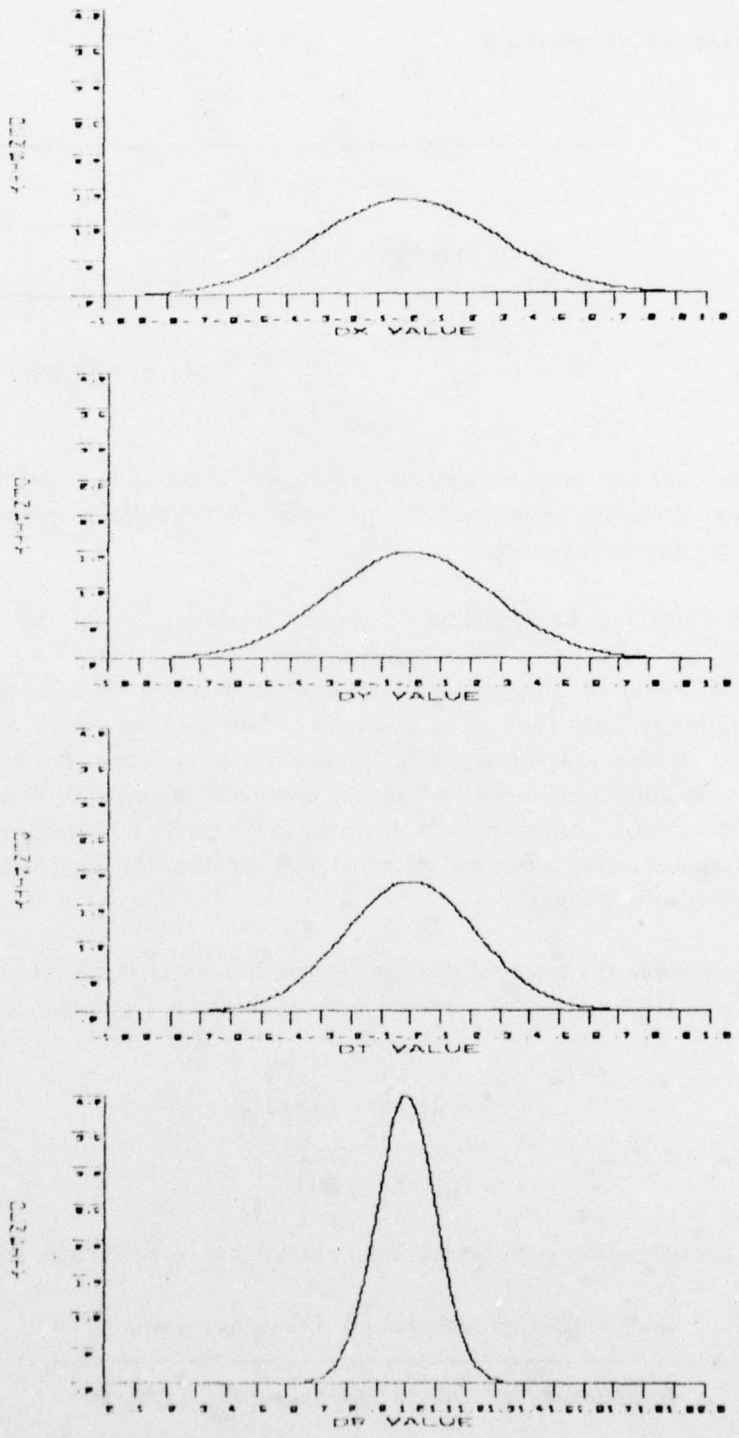


Figure 3.5.5

(3.5.15) $$P[On|v1,v2,p1,p2] = \frac{1}{1 + \prod_{i=1}^{N} \frac{P[vi|Off]}{P[vi|On]} * \frac{\sum_{j=1}^{m1}\sum_{k=1}^{m2} P[p1; p2; g1,j; g2,k]}{\sum_{j=1}^{n1}\sum_{k=1}^{n2} P[p1; p2; f1,j; f2,k]}}.$$

The value terms are based upon the operators' evaluations of the appearances of the features and the position terms are based upon the likelihood of the object being at the location implied by the position of the matches.

Formula 3.5.15 can be expanded to include N operators, but the combinatorial explosion limits its usefulness. For example, if five operators each have three known alternatives when the screw is present and three alternatives when the screw is missing, 496 separate probabilities would have to be computed. Ten operators would require 118118 terms. This fact suggests that formula 3.5.15 should only be used to evaluate the structural consistency of small subsets of the total number of operators. Section 4.2 will discuss the use of this type of formula to assign known alternatives to the results of two or three operators. In effect, the formula makes it possible to choose the most consistent pattern of alternatives for a small number of operators.

One way to avoid the computation costs inherent in the straightforward application of formula 3.5.15 to a large number of operators is to use some other technique to approximate the ratio

(3.5.16) $$\frac{P[p1,p2 \mid Off]}{P[p1,p2 \mid On]}.$$

One heuristic that has proved to be experimentally useful is to replace (3.5.16) by

(3.5.17) $$\frac{\langle \text{percentage of consistent features, given Off} \rangle}{\langle \text{percentage of consistent features, given On} \rangle}.$$

This ratio is only a crude approximation to the ratio of the probabilities, but it is useful

because it contributes a factor based upon the structural consistency of the matches.

Since the total number of possible matches is the same for the two cases, On and Off, the ratio of percentages reduces to

$$(3.5.18) \qquad \frac{\langle \text{number of consistent features, given Off} \rangle}{\langle \text{number of consistent features, given On} \rangle}.$$

Thus, the contribution of structural consistency in the probability formulas can be transformed into a ratio of the numbers of consistent matches.

In order to compute a value for the ratio in (3.5.18) the least-squares culling routine is applied twice: once assuming that the screw is present and once assuming that the screw is missing. The input for each application is a list of pairs:

(3.5.19) <planned feature position, matching position of the operator>.

The planned feature positions for the operators depend upon the assumption about the screw. If the screw is assumed to be present, one set of planning positions is used. If the screw is assumed to be missing, a different set of planning positions is used, because the operators are expected to find different features when the screw is missing.

The set of features for each assumption forms a geometric pattern (or structure). The least-squares culling routine is used to measure the agreement between a planning pattern and an actual pattern of matches. The ratio in (3.5.18) measures the relative agreement. This relative measure is the contribution of the position information toward the overall confidence.

In most cases the structure of the planning features is significantly different when the screw is On than when the screw is Off. This difference essentially guarantees that the ratio in (3.5.18) will not be close to 1.0. As the number of matches included in the fit increases, the structures for the two situations become more distinct. This behavior is important if the position information is to distinguish between the two situations.

An important assumption of this discussion so far has been that the operators match unique features, one for On and one for Off. If there are several known alternatives for an operator, the program has to decide which is the best match for that operator and assign that match to the operator so the least-squares culling routine can be called. The basic formula used to determine the best alternative was developed in section 3.2. It is

$$(3.5.20) \qquad P[Lj|v] = \frac{1}{1 + \sum_{i \neq j} \frac{P[v|Li]*P[Li]}{P[v|Lj]*P[Lj]}} \qquad \text{(for all j).}$$

If there happen to be two or more alternatives with approximately equal probabilities of being the best match, the least-squares culling procedure can be extended as follows: whenever the first choice is about to be discarded (because it is the least consistent match), replace it with another one of the approximately equal choices. This extension increases the complexity of the least-squares culling routine, but it provides an automatic way of giving an operator a second chance whenever there is more than one possible explanation for its results.

The incorporation of the position information does not alter the ease with which the probabilities can be computed. Sequentially acquired information can still be included very nicely. Since the least-squares culling procedure can not be applied until some minimum number of features has been located, the position information can not contribute anything until that minimum number has been reached. The minimum number depends upon the number of parameters being adjusted, the number of equations contributed by each operator/feature pair, and the number of independence conditions. For example, if the least-squares method is performing a planar fit, there are three parameters, dX, dY, and $d\alpha$. Since each correlation feature and each point-on-a-line feature contributes two equations, any two of these features would be sufficient. Three or four would be better because the least-squares technique performs better when the parameters are over-constrained. Since this is true, it may be advantageous to increase the number of matches above the theoretical minimum before using this method to incorporate position information.

If there are several known alternatives for an operator, the program may make a mistake when it assigns an alternative to the results of an operator. That is, given the value and position information of a match for the operator, the program may decide that the operator located alternative `f1,1`, when in fact, it located alternative `f1,2`. Since the program does not know whether it is correct or not, one possibility is to use the probability associated with the best match as the expected number of good matches contributed by the operator. Thus, if the probability associated with the best alternative is .8, the operator contributes .8 of a good match toward the desired minimum.

Figure 3.5.6 outlines the general method suggested by this section. One operator after another is applied until the accumulated value information indicates that sufficient features have been located; then the least-squares method is applied. Additional features are added until the confidence reaches the desired limit. This algorithm could form the basis for an inspection system. It can be used to decide if a gasket is already on, to decide if a hole has been drilled, or to decide if the expected subassembly has been added.

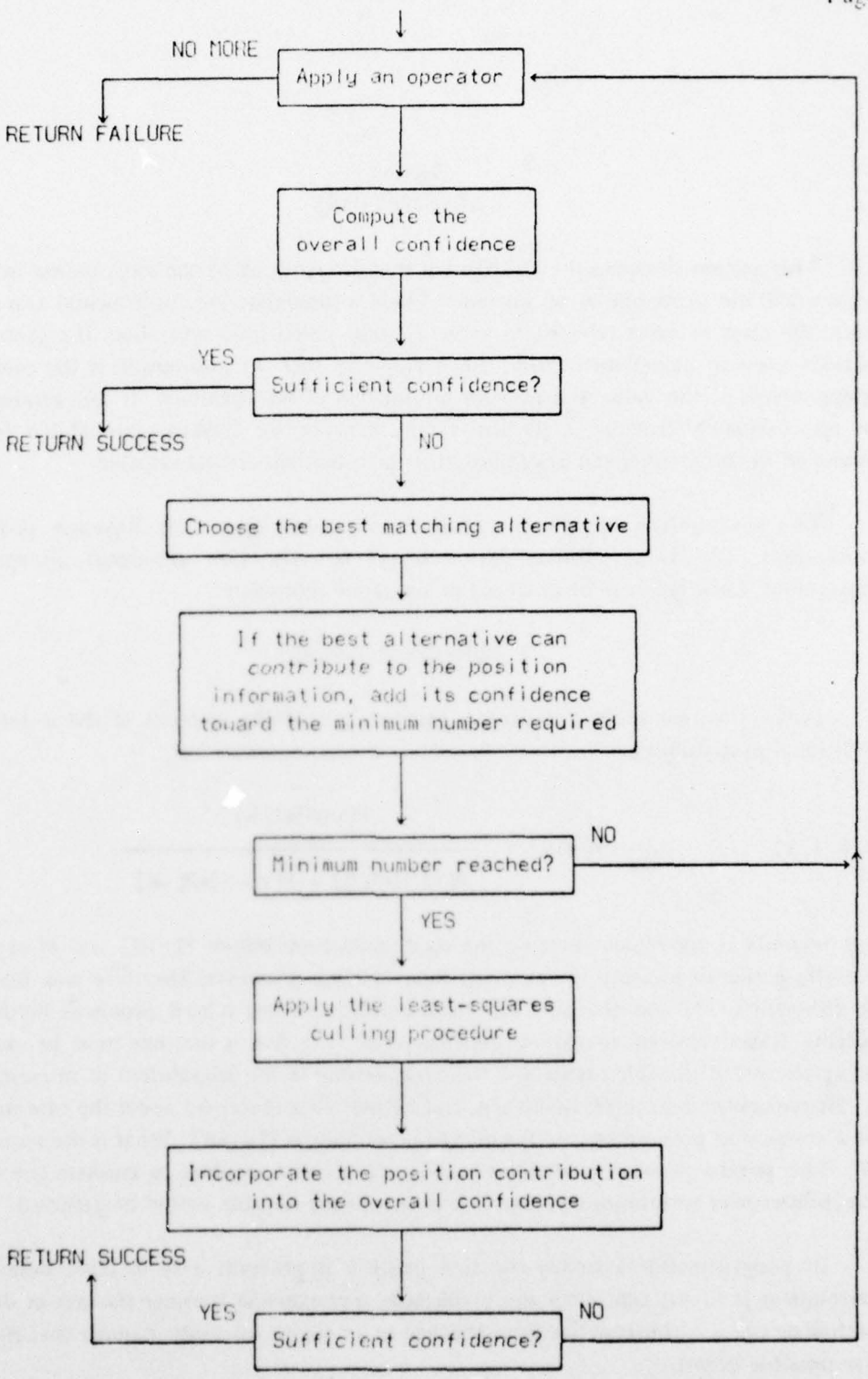

NO MORE
Apply an operator
RETURN FAILURE
Compute the overall confidence
YES
Sufficient confidence?
RETURN SUCCESS
NO
Choose the best matching alternative
If the best alternative can contribute to the position information, add its confidence toward the minimum number required
NO
Minimum number reached?
YES
Apply the least-squares culling procedure
Incorporate the position contribution into the overall confidence
RETURN SUCCESS
YES
NO
Sufficient confidence?

Figure 3.5.6

Section 6
ASSUMPTIONS

This section discusses the assumptions that are required by the combination rules used to compute the probabilities of interest. These assumptions are fundamental assumptions about the class of tasks referred to as verification vision tasks and about the probabilistic methods used to model such tasks. An example of such an assumption is the conditional independence of the value and position information of the operators. If this assumption is not approximately true for a particular task, none of the Bayesian probability formulas developed in this chapter can be applied; their preconditions are not satisfied.

The assumptions have been classified into three types: (1) Bayesian probability assumptions, (2) value distribution assumptions, and (3) conditional independence assumptions. Each type will be discussed in a separate subsection.

1. Bayesian Probabilities

Bayes' theorem states a desired *a posteriori* probability in terms of the *a priori* and conditional probabilities:

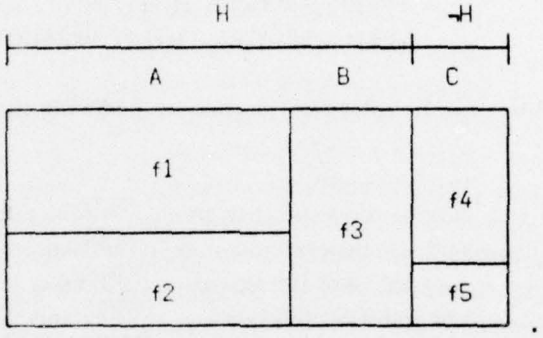
$$P[H|v] = \frac{P[v|H]*P[H]}{P[v|H]*P[H] + P[v|\neg H]*P[\neg H]}. \qquad (3.6.1.1)$$

This formula is convenient because the conditional probabilities P[v|H] and P[v|¬H] are generally easier to measure (or estimate) than P[H|v]. However, Shortliffe and Buchanan (see [Shortliffe 75] and [Nilsson 75]) have pointed out two related problems involved in applying Bayes' theorem to various decision tasks. The first is that one must be careful to specify the set of possible events and their relationship to the propositions of interest, H and ¬H. In particular, Shortliffe, Buchanan, and Nilsson were concerned about the case in which H is a compound proposition and the desired probability is P[v|¬H]. What is the meaning of ¬H? The second problem is that the amount of statistics required to estimate the desired probabilities may be prohibitive, even if it is clear which statistics should be gathered.

In programmable assembly the first problem is generally easy to solve because the environment is highly controlled and predictable. For example, consider the task of deciding whether or not a carburetor has been attached to an engine assembly. Assume that there are three possible events:

(3.6.1.2) $A \equiv$ ⟨carburetor of type X is attached⟩,
$B \equiv$ ⟨carburetor of type Y is attached⟩,
and $C \equiv$ ⟨no carburetor is attached⟩.

Consider an operator that behaves as follows: when a carburetor of type X is attached, it locates one of two features, f1 or f2; when a carburetor of type Y is attached, it locates feature f3; and when there is no carburetor attached, it locates one of two features, f4 or f5. The corresponding *a priori* probabilities can be diagramed as follows:

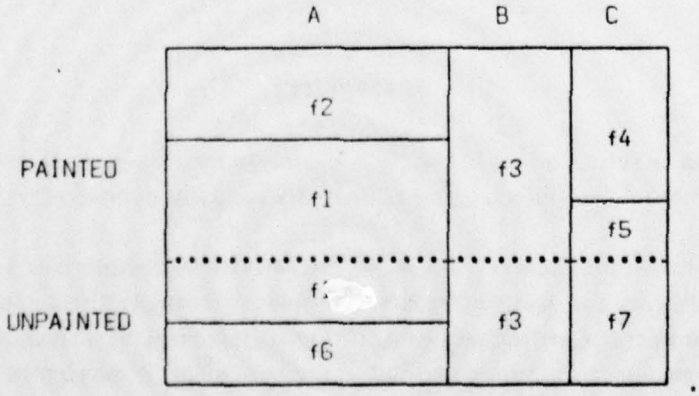


P[v|H] and P[v|¬H] can be expanded into

(3.6.1.3) $P[v|H] = P[v|f1]*P[f1] + P[v|f2]*P[f2] + P[v|f3]*P[f3]$

and $P[v|\neg H] = P[v|f4]*P[f4] + P[v|f5]*P[f5]$.

The same enumeration procedure can be applied when H is a compound proposition. For example, assume that the engine casing, mentioned in the previous example, may or may not be painted and that the *a priori* probabilities of the possible events are:



If the proposition of interest is

(3.6.1.4) $$H \equiv A \cap \langle PAINTED \rangle,$$

P[v|H] and P[v|¬H] can be expressed as follows:

(3.6.1.5) $$P[v|H] = P[v|f2]*P[f2] + P[v|f1,PAINTED]*P[f1,PAINTED]$$

and $$P[v|\neg H] = P[v|f3]*P[f3] + P[v|f4]*P[f4] + P[v|f5]*P[f5] + P[v|f6]*P[f6] + P[v|f7]*P[f7] + P[v|f1,UNPAINTED]*P[f1,UNPAINTED].$$

Thus, the first problem can be solved by carefully enumerating the possible events and their interactions.

Even though it is easy to solve the first problem within programmable assembly, it is difficult to avoid the second. Whenever there are several known alternatives, the training session has to gather statistics for all of the alternatives and all of the interactions between the alternatives. The amount of statistical information required and the number of trials needed to produce valid estimates for the statistics depend upon the number of alternatives and the expected distributions for the information. These quantities will be discussed in the chapter describing the the programming and training phases.

2. *Value Distributions*

Throughout the development of the formulas a normal distribution has been assumed for the value information of the operators. That is, the values associated with an alternative were assumed to have a normal distribution. This assumption, however, is *not* necessary to compute the likelihood ratios

(3.6.2.1) $$\frac{P[vi|On]}{P[vi|Off]}.$$

Any distribution is sufficient. It is even possible to use the histogram of values produced at training time as the distribution, as long as there is a sufficient number of trials.

A normal distribution was assumed in the derivations because it is a good model for some of the operators and it is convenient for displaying example distributions. However, if the values of an operator are not normally distributed, there may be a change of variable that can convert them into an approximately normal distribution. A later portion of this section will discuss a change of variable that converts correlation values into a distribution that is

approximately normal.

If values of an operator are known to follow some distribution other than a normal distribution, it is easy to incorporate the new distribution into the execution-time formulas. The only information needed in addition to the density function is a specification for the interval of *reasonable* values. What values of the operator should be classified as *unusual* and hence should be filtered out (see section 3.3)? For normal distributions it is easy to specify an interval in terms of the number of standard deviations away from the mean. Other distributions require some other specification for the interval of reasonable values.

Any distribution can be used for the value information of an operator. For example, if some *a priori* information implies that the distribution for a particular operator is a gamma distribution, a gamma distribution can be substituted into the appropriate formulas. If the training results imply that the distribution is not one of the standard distributions, the density function defined by the histogram can be used in the formulas.

Some of these other distributions have properties that make them good models for the operators used in VV. For example, the gamma distribution is asymmetrical; it can model a wider range of distributions than a normal distribution, and yet its form can be easily determined from the experimental mean and standard deviation of the training samples.

A Change of Variable for Correlation Values

One operator that is known to produce a non-normal distribution is cross-correlation (see [Hoel 71]). Consider the following formula for the correlation coefficient:

$$r = \frac{\sum_{i=1}^{N} (Xi-Mx)*(Yi-My)}{N * Sx * Sy}, \qquad (3.6.2.2)$$

where Xi and Yi are jointly normally distributed, Mx and My are the sample means of X and Y, respectively, and Sx and Sy are the sample standard deviations. It would be possible to use the actual distribution of r, but there is a convenient change of variable that converts r into a distribution that is approximately normal. The change of variable is

$$z = \frac{1}{2} * \log\left(\frac{1 + r}{1 - r}\right). \qquad (3.6.2.3)$$

The mean of the new distribution is

$$Mz = \frac{1}{2} * \log\left(\frac{1 + \alpha}{1 - \alpha}\right), \tag{3.6.2.4}$$

where α represents the theoretical value of the correlation coefficient. The standard deviation for the new distribution is

$$Sz = \frac{1}{\mathrm{sqrt}(N - 3)}, \tag{3.6.2.5}$$

where N is the number of samples used to compute r.

The correlation operator implemented by Hans P. Moravec at Stanford and used in this work behaves according to this theory [Moravec 76]. Consider figure 3.6.2.1. Figure 3.6.2.1.a is a histogram of fifty correlation coefficient values. The values are the results of applying the same correlation operator to fifty different pictures of a scene for one VV task. The interval size along the horizontal axis of the graph is one-half of the sample standard deviation. As predicted, the correlation values form a skewed distribution (with a theoretical upper limit of 1.0). The chi-square value is based upon the eleven intervals centered about the sample mean. Figure 3.6.2.1.b is the histogram of values produced by the change of variable in formula 3.6.2.3. Figure 3.6.2.1.c is the histogram that would be expected if the sample formed a perfect normal distribution.

The chi-square value drops significantly from 25.4 (with eight degrees of freedom) to 9.3 (with eight degrees of freedom) for the new distribution. The improvement is not always that dramatic, but the change of variable often transforms a skewed distribution into a distribution closer to normal. Consider figure 3.6.2.2, a scatter diagram of the pairs:

(3.6.2.6) (<chi-square of raw values>, <chi-square of changed values>).

Any point to the right of the diagonal line represents a case in which the change of variable made the distribution for the values of an operator look more like a normal distribution (according to the chi-square test). The change of variable only slightly degrades the chi-square value in the few cases that it makes the distribution worse. A point in the shaded area of figure 3.6.2.2 represents an operator whose distribution was improved significantly. Before the change of variable the chi-square test (at the 5% level) rejected the hypothesis that the sample could have come from a normal distribution. After the change of variable the chi-square test indicated that it was plausible for the sample to have come from a normal distribution.

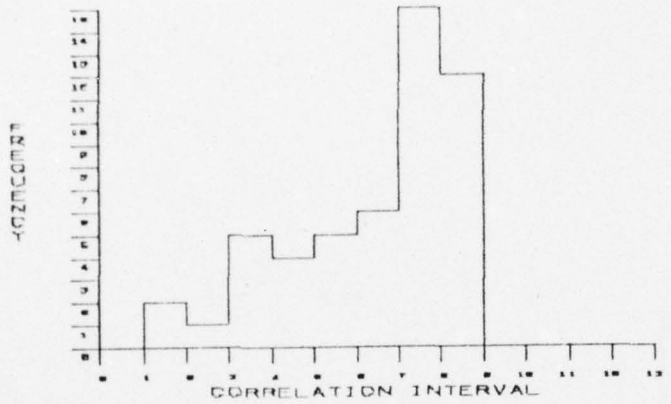


Figure 3.6.2.1.a

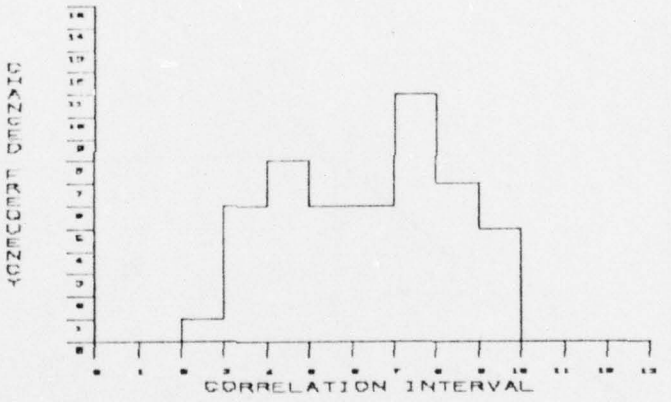


Figure 3.6.2.1.b

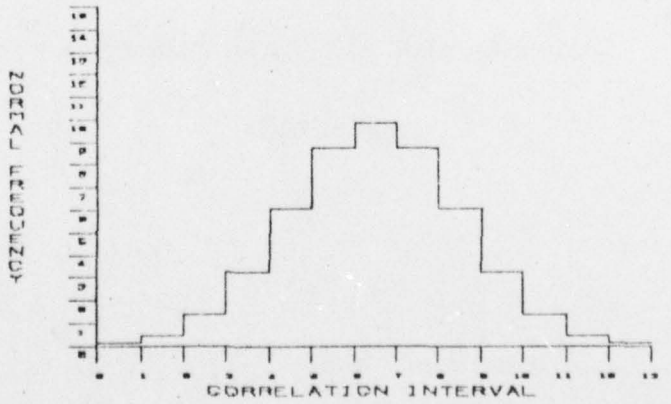


Figure 3.6.2.1.c

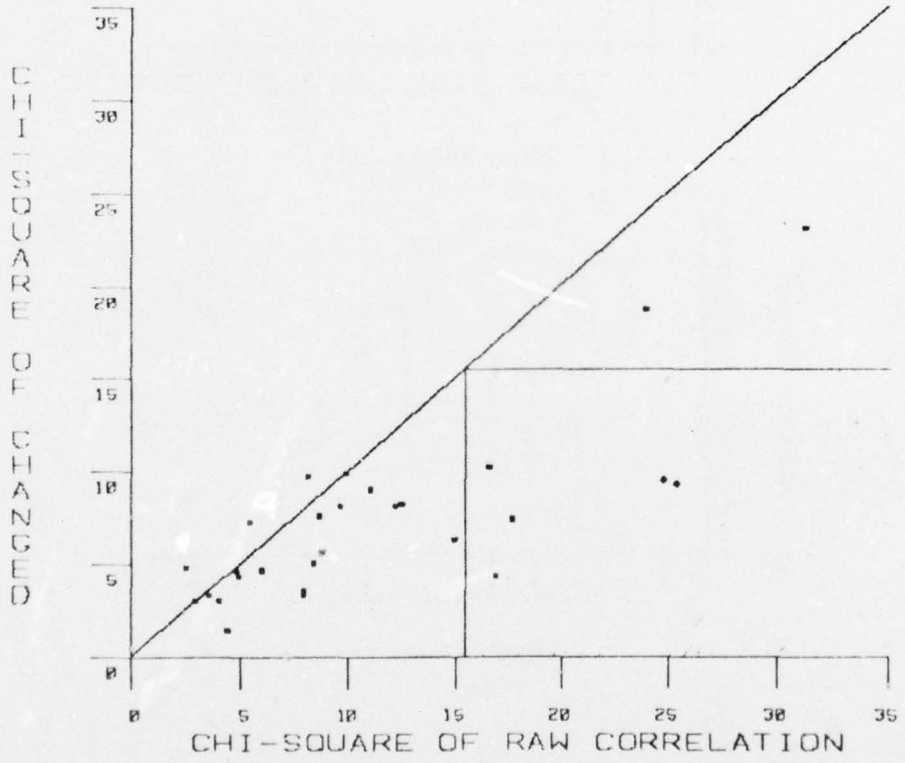

CHI-SQUARE OF CHANGED
35
30
25
20
15
10
5
0
0
5
10
15
20
25
30
35
CHI-SQUARE OF RAW CORRELATION

Figure 3.6.2.2

The question about which distribution to use to model the results of an operator is a hard one. The chi-square test used above is helpful, but it is mainly a method to reject proposed models.

3. Conditional Independence

The derivations of several of the formulas depend upon two important assumptions about the conditional independence of the values of the operators: (1) the value of an operator is conditionally independent of the values of the other operators and (2) the value of an operator is conditionally independent of the position of its match. Both of these assumptions are instrumental in simplifying the relevant formulas. For example, they make it possible to simplify formula 3.5.2:

$$(3.6.3.1)\quad P[On|v1,\ldots vN,p1,\ldots pN] = \frac{1}{1 + \frac{P[v1,\ldots vN,p1\ldots pN|Off]}{P[v1,\ldots vN,p1\ldots pN|On]} * \frac{P[Off]}{P[On]}}$$

into

$$(3.6.3.2)\quad P[On|v1,\ldots vN,p1,\ldots pN] = \frac{1}{1 + \frac{P[p1\ldots pN|Off]}{P[p1\ldots pN|On]} * \frac{P[Off]}{P[On]} * \prod_{i=1}^{N} \frac{P[vi|Off]}{P[vi|On]}}.$$

The assumptions significantly reduce the number of dependencies within the conditional probabilities and make them feasible to compute. The formulas are designed to be as flexible as possible and still be computable. If the assumptions are not true for an operator, formula 3.6.3.2 can not be used to estimate the desired probabilities.

Both of the assumptions depend upon the values of the operators and the values of the operators are based upon the appearances of the features. Unfortunately, there are several reasons why the appearance of a feature might change from one picture to the next:

(1) The feature itself may be different. For example, in assembly tasks all pump bases are not exactly the same, so all screw holes are not the exactly same. In photo-interpretation (abbreviated PI) features, such as roads, may be changed

(2) The position and orientation of objects in the picture may change. In assembly the pump bases may not seat exactly the same way in the vise.

(3) The lighting may be different. In PI the sun may be in a different location, causing different shadows and glares. In programmable assembly lights can be controlled more easily, but they still may vary slightly.

(4) The position and orientation of the camera may be different. In assembly the camera may be used for more than one task in such a way that it must be repositioned before each task. Repositioning is not exact. In PI the position of the plane or satellite may be different.

(5) The sensitivity of the camera may be different. Cameras have internal parameters such as target voltage that change over time.

(6) The camera noise level is variable.

In effect, the two conditional independence assumptions state that none of these variables change the expected distribution of values produced by an operator in VV. Some operators, of course, do depend upon one or more of these variables. This fact raises an important question:

> Given a specific VV task, is there a way to determine whether or not the assumptions hold for a particular combination of operators?

The remainder of this section develops some insight into this question.

The first assumption states that the value of an operator is conditionally independent of the values of the other operators, e.g.,

(3.6.3.3) $$P[v2|On,v1] = P[v2|On].$$

This formula says that given On, the probability of operator 2 producing the value v2 is the same whether or not the value of operator 1 is known. Formula 3.6.3.3 is equivalent to

$$\frac{P[v2,On,v1]}{P[On,v1]} = P[v2|On], \tag{3.6.3.4}$$

$$\frac{P[v1,v2|On]}{P[v1|On]} = P[v2|On], \tag{3.6.3.5}$$

and

$$P[v1,v2|On] = P[v1|On] * P[v2|On]. \tag{3.6.3.6}$$

Thus, it is possible to check the first assumption for two operators by (1) gathering the statistics necessary to form the distributions associated with the three conditional probabilities in (3.6.3.6) and (2) appling a test, such as the Chi-square test, to decide if the distribution formed by the convolution of the distributions for P[v1|On] and P[v2|On] is the same as the distribution for P[v1,v2|On]. The second assumption can be checked in a similar manner.

The first assumption is often true in VV because different operators depend upon different properties of the picture. The probability of producing a certain value for a correlation operator is often unaffected by the results of a previously applied edge operator. An obvious case in which the assumption is not true is when operator 1 and operator 2 are both correlation operators and they overlap. *Knowing the value of one operator certainly* alters the possible values for the second. Similarly, an edge and correlation test for the same corner will be inter-dependent. Fortunately, since most of the objects in programmable assemble have several interesting features, overlapping operators can be avoided.

The second assumption states that the appearance of a feature on an object does not change as the object moves around *within its tolerance volume*. Put another way, if an operator is applied to several different pictures, and it locates the same known alternative in each, the value returned by the operator is independent of the location of the alternative in *the picture*. This is usually *true in* VV because, by assumption, the changes are so small that the appearance of a feature is essentially constant.

However, there are two situations in which the second assumption might be false. The *first is when a small change in one of the variables in the transform causes a shadow to fall* on a feature. At some locations the feature is in a shadow and at others it is not. The value of almost any operator attempting to locate such a feature would depend upon whether the feature is in a shadow or not. Hence the value of the operator would depend upon the

position of the feature and the assumption would be false. The second situation arises when a small change in position causes a dramatic change in the appearance of a feature. For example, one picture may show a screw hole that is partially occluded on the left by a shaft and a second picture of the same hole may show the shaft occluding the hole on the right.

Both of these situations lead to operators that produce bivariate (or at least high variance) density functions. One peak is produced by the pictures showing the feature in the shadow and the other peak is produced by the pictures showing it in the light. Since the expected contribution for such operators is generally low, the automatic ranking scheme will place this type of operator near the bottom of the list of potential operators to be used in a task. If the VV system is interactive, a programmer can also discard any features of this type suggested by the system. If an operator is particularly important it is possible to break up the task into two or three subtasks, each of which satisfies the assumption. Taylor has used this technique within his constraint resolving system whenever the angular uncertainties are greater than a few degrees [Taylor 76].

CHAPTER 4

EXECUTION-TIME COMBINATION RULES FOR LOCATION

If the verification vision system is trying to locate, not inspect, an object, there are two important quantities: (1) an estimate for the location of the object and (2) the precision associated with that estimate. In the context of VV the *location of an object* refers to the position and orientation of the object's coordinate system in terms of some other coordinate system (e.g., the workstation coordinate system). Usually there is some point or feature on the object of particular interest, e.g., the center of a hole or the tip of a screw. Such a point will be referred to as a *point of interest*.

The previous chapter briefly mentioned that a least-squares technique can be used to combine a set of planned positions with a set of corresponding measured positions to produce an estimate for the transformation between them. Given this transformation and the planned location of the object, it is easy to compute the current estimate for the location of the object. The least-squares technique can also produce the standard deviations associated with the estimates for the individual parameters in the transform. These standard deviations can be combined to produce an estimate for the precision.

There are other metrics and fitting procedures beside least-squares techniques. The least-squares approach was chosen for the current implementation of the VV system because it provides the desired location and precision information and it is a well-known technique.

The application of the least-squares routine depends upon the correspondence between the matching points and the planning features. If the correspondence is correct, the estimate for the location of the object and the associated precision will be correct. If it is not correct, it is possible to determine a (seemingly) *structurally consistent* subset of the features, which leads to an incorrect estimate for the location of the object. This problem only arises when there are several known alternatives for the features or when the operators find surprises. To avoid making this mistake an overall estimate of the probability that the object is within the stated precision is needed. Such an estimate can be based upon the value and position information of the operators.

This chapter begins with a detailed explanation of the least-squares method and its application to the VV problem of producing location and precision estimates. The emphasis

is on the theoretical aspects of the least-squares method. The practical applications of this method are demonstrated in appendix V, which contains an annotated trace of a programmer interacting with the current implementation of the VV system. The programmer sets up a VV program to locate a screw hole and tests it on several examples. The trace demonstrates the computation of precision, the culling of bad matches, and the reduction of tolerance regions.

The second section of this chapter describes an example in which the results of the least-squares method are incorrect and then discusses two methods to estimate the confidence associated with a statement of precision.

Section 1
DETERMINING PRECISION

This section presents a general method for performing *nonlinear* generalized least-squares adjustments. A *major* portion of this discussion is a restatement of an internal paper at the Stanford Artificial Intelligence Project written by Donald B. Gennery entitled "Least-Squares Stereo-Camera Calibration" [Gennery 75]. The method uses partial derivatives to approximate the problem under the general linear hypothesis model of statistics, and then iterates to achieve the exact solution. For more detailed information see [Graybill 61].

The notational conventions are the following. Capital letters denote matrices. Vectors are represented by column matrices. A particular element of a matrix is represented by the corresponding lower-case letter followed by the appropriate indices. The transpose of a matrix A is denoted by A', and the inverse of A is denoted by A~. Multiplication (either scalar or matrix) is denoted by an asterisk.

Let the vector G denote a set of m unknown parameters for which values are desired. Let the vector U be a set of n scalar quantities ($n \geq m$) that are functions of G and can be measured *with some error*. Let F represent the vector of n functions that relate elements in U with G. Given an estimate for G, F(G) produces an estimate for U. Finally let the vector V represent the n residuals (i.e., the *unexplained errors*) that remain between U and an estimate produced by F(G). Thus

$$U = F(G) + V. \tag{4.1.1}$$

The goal is to minimize a function of V by modifying G.

In verification vision G is the set of parameters in the transform that maps the planned positions of the features into their matching positions (i.e., the planned positions into the measured positions). Typical elements in G are the displacement in X, dx, the displacement in Z, dz, and the unknown rotation about the Z-axis, dα. Different operator/feature pairs contribute different components to U and F. For example, when the transform is planar (so that the unknown parameters are dx,dy, and dα), a correlation operator/feature pair contributes two measured values to U: the X and Y components of the match (let them be referred to as Xm and Ym). The corresponding functions in F are:

(4.1.2)
$$Xe = (Xp-Xc)*COS(d\alpha) - (Yp-Yc)*SIN(d\alpha) + dx + Xc$$
$$Ye = (Xp-Xc)*SIN(d\alpha) + (Yp-Yc)*COS(d\alpha) + dy + Yc$$

where (Xc,Yc) is the center of rotation for dα, (Xp,Yp) is the planned position for the correlation patch, and (Xe,Ye) is the transformed position of (Xp,Yp). The transformed position of (Xp,Yp) is the estimate for (Xp,Yp)'s position in the current picture. The two residuals that would be associated with a correlation feature are

(4.1.3)
$$Xm - Xe$$
and
$$Ym - Ye.$$

These residuals are the components of V. The goal, of course, is to use the measured values to improve the estimates for the parameters.

The quadratic form

(4.1.4)
$$q = V'*W*V$$

is the criterion of optimization that is to be minimized. W denotes an n by n weight matrix. If W is the inverse of the covariance matrix of the errors in the observations, the result will be the maximum likelihood (in the F space) solution if the errors have a normal distribution. If W is a diagonal matrix, which indicates no correlation between errors in the different observations, the quadratic form reduces to a weighted sum of the squares of the elements of V. Thus the problem as stated here can be said to be a generalized least-squares adjustment.

The difficulty in obtaining a solution to the above problem lies in the fact that F in (4.1.1) is a nonlinear function, and thus in general there is no closed-form solution. One way of solving the problem is to use some type of steepest descent technique, which tries new values of G, recomputes q, and tries to drive q to a minimum. However, such methods tend to converge rather slowly. Also, numerical problems may occur if q has a very broad minimum, for round-off errors may give rise to spurious local minima. Instead of such an approach, the method described here approximates (4.1.1) by a linearization based on the

partial derivatives of F, solves the resulting linear problem, and iterates this process to obtain the solution to the nonlinear problem. The idea is similar to the Newton-Raphson method.

Let the n by m matrix P be composed of the partial derivatives of the functions in F, such that

$$p_{ij} = \frac{\partial f_i}{\partial g_j}. \tag{4.1.5}$$

Let G0 denote an approximation to G. Then equation (4.1.1) can be approximated as follows:

$$U = F(G0) + P(G0)*(G - G0) + V \tag{4.1.6}$$

where the functional dependence of P on G has been explicitly indicated. Define

$$\begin{aligned} E &= U - F(G0) \\ D &= G - G0. \end{aligned} \tag{4.1.7}$$

Then (4.1.6) can be rewritten as

$$E = P*D + V. \tag{4.1.8}$$

Thus the nonlinear equation (4.1.1) has been replaced by the linear equation (4.1.8), in which E represents the discrepancy between the observations and their computed values (using the current approximations of the parameters), and D represents the corrections needed to the parameters.

It is necessary to solve for D in (4.1.8) in order to minimize q in (4.1.4). This is a standard problem in linear statistical models (e.g., see [Graybill 71]). The solution for D is

$$D = (P'*W*P)\sim*(P'*W*E) \tag{4.1.9}$$

and the covariance matrix of errors in the solution for D is

$$S = (P'*W*P)\sim \tag{4.1.10}$$

assuming that W is the inverse of the covariance matrix of the observation errors.

Several other quantities of interest can be derived from the solution. The expected value of q is n-m. If the scale factor of the covariance matrix of observation errors is unknown, W can be adjusted by the ratio (n-m)/q and S by the ratio q/(n-m). Otherwise, q

can be used as a test on the adjustment; for, if the observation errors have the Gaussian distribution, q has the chi-square distribution with n-m degrees of freedom. S represents the covariance matrix of errors in the adjusted parameters. The square roots of the diagonal elements of S are the standard deviations of the adjusted parameters. The correlation matrix of the parameters can be obtained from S by dividing the i,j element by the product of the standard deviations of the ith and jth parameters, for all i and j.

Other results are the covariance matrix of the adjusted observations P*S*P' and the covariance matrix of the residuals W~ - (P*S*P'). It is often useful to compare the magnitude of the residuals to their standard deviations, i.e., the square roots of the diagonal elements of their covariance matrix. If a residual is greater than two (or three) standard deviations it indicates that the associated measured value is *inconsistent* with the other values used to compute the estimate for the transform. This test is the basis for the least-squares culling procedure mentioned in the previous chapter.

The covariance matrix about a point *not* in the solution is W~ + (P*S*P') where P is the set of partial derivatives at the point and W is the inverse of the covariance matrix that weights the measured values. In VV the standard deviation that can be computed from this covariance matrix can be used to determine the uncertainty associated with any other point on the object (e.g., a *point of interest*). It can also be used to determine the tolerance region about the next feature to be tried.

The solution of the nonlinear problem can now be described as follows. An initial approximation is used to compute the discrepancies Ei and the partial derivatives Pij. Then D is computed from (4.1.9) and is added to the current approximation for G to obtain a better approximation. This process repeats until there is no further appreciable change in G. Then the final values from the last iteration can be used to obtain S, Vi, q, and the other derived quantities described above. Of course, in order to converge to the absolute minimum of q rather than converge to some local minimum or divergence, it is necessary that the initial approximation be sufficiently close to the true solution. In most practical problems the initial approximation is not critical; in fact, often there is only one minimum.

Since on the last iteration the partial derivatives have been computed for the converged value of G, the solution gives the true generalized least-squares adjustment regardless of the nonlinearity. However, some other properties of the adjustment are only approximate in the nonlinear case. Among these are the use of S as the covariance matrix of the errors in the final value of G, and the properties that the solution for G is minimum-variance and unbiased. However, if the amount of nonlinearity over the range of the measurement errors is small, these results will be fairly accurate.

Figure 4.1.1 is a flowchart that outlines the basic steps involved in using a least-squares method to compute an estimate for the location of an object and a precision about that

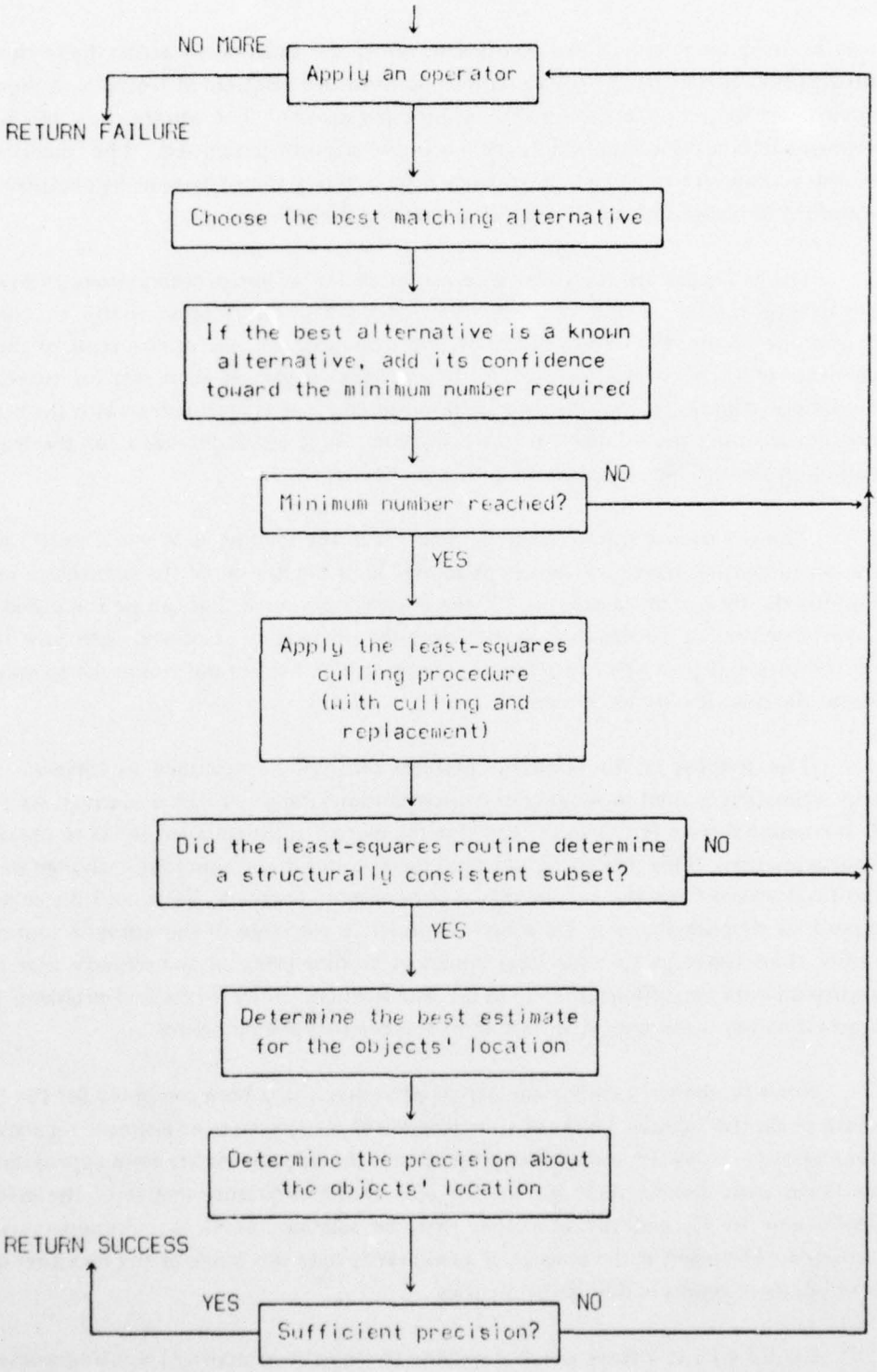

NO MORE
Apply an operator
RETURN FAILURE
Choose the best matching alternative
If the best alternative is a known alternative, add its confidence toward the minimum number required
NO
Minimum number reached?
YES
Apply the least-squares culling procedure (with culling and replacement)
Did the least-squares routine determine a structurally consistent subset?
NO
YES
Determine the best estimate for the objects' location
Determine the precision about the objects' location
RETURN SUCCESS
YES
NO
Sufficient precision?

Figure 4.1.1

estimate. The algorithm is a sequential algorithm that applies the least-squares routine as soon as a sufficient number of features has been found. The best values for the parameters are used to map the object's planned location into an estimate for its current location. The standard deviations associated with the best parameter values are combined to produce a region of uncertainty about the estimate. As stated, the algorithm is concerned with the location of the object. Given the estimate for the location of an object and a precision, it is easy to produce estimates and tolerance regions for any *points of interest* on the object.

Section 2
CONFIDENCE IN THE PRECISION

The algorithm shown in figure 4.1.1 can be used by itself to locate objects. However, to do so requires an assumption: if the least-squares culling routine determines a structurally consistent subset of the features, and if the desired precision has been reached, then a correct correspondence has been established between the positions produced by the operators and the known alternatives for the features. This assumption is generally reasonable when the number of known alternatives is small and the operators are reliable (i.e., they do not locate surprises very frequently). However, it is possible to locate a set of features that appears (to the least-squares culling routine) to be structurally consistent, when in fact, some of the results have been incorrectly assigned to alternatives. For example, consider figure 4.2.1. Figure 4.2.1.a shows a *point of interest* and sets of known alternatives for four operators. Operators three and four each have two known alternatives. Figure 4.2.1.b shows the *actual* positions of all of these points in a particular test picture. These positions are the positions at which the operators should find them, if they are reliable. Figure 4.2.1.c superimposes the best matching positions for the four operators on top of the actual positions. If the program decides that operator three has matched alternative 3.a and that operator four has matched alternative 4.a (both of which are wrong), the least-squares routine will probably decide that the features are structurally consistent and proceed to predict that the point of interest is located at the position marked in figure 4.2.1.d. This conclusion is wrong. The cause of the error was the incorrect assignment of alternatives to the results of the operators. The resulting assignment happens to appear to be structurally consistent and the program, having fooled itself, proceeds to draw an incorrect conclusion. This example is a simple example, but it points out a potential danger in unconditionally believing the results of the least-squares culling routine.

The program makes the wrong implication because (a) some of the known alternatives are incorrectly assigned to the results of the operators and (b) the pattern formed by the incorrect assignment happens to be structurally consistent. Thus there are two basic questions related to this type of mistake:

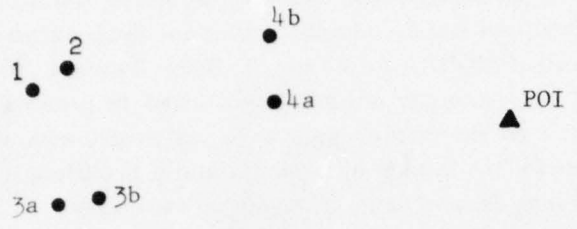


Figure 4.2.1.a

Figure 4.2.1.b

Figure 4.2.1.c

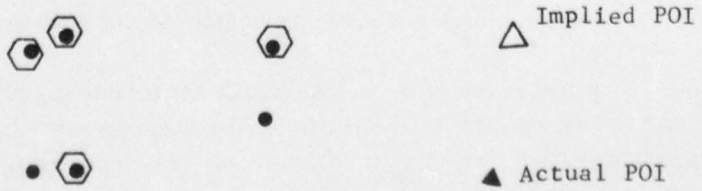


Figure 4.2.1.d

(1) Given the value and position information for several operators and an assignment of known alternatives to these results, what is the probability that the assignment is correct? That is, what is the probability that all of the individual assignments for the operators are correct?

(2) Given an assignment for a set of operators that is structurally consistent, are there other assignments for the same results that are also structurally consistent? That is, how unique is the pattern formed by the assignment?

These questions will be discussed in the remainder of this section.

It is possible to formulate a version of Bayes' theorem that expresses the probability that all of the individual assignments are correct, given the position and value information produced by the operators. Consider the case of two operators. Assume that operator 1 has M known alternatives and operator 2 has N known alternatives. Let f1,f2,...fM be the alternatives for operator 1 and g1,g2,...gN be the alternatives for operator 2. Then Bayes' theorem states:

$$(4.2.1)\quad P[fj,gk \mid v1,p1,v2,p2] = \frac{1}{1 + \dfrac{P[v1,v2,p1,p2 \mid \neg(fj,gk)]}{P[v1,v2,p1,p2 \mid fj,gk]} * \dfrac{P[\neg(fj,gk)]}{P[fj,gk]}}.$$

The probability P[fj,gk|v1,p1,v2,p2] represents the probability that the first operator has located alternative fj and that the second operator has located alternative gk, given the value and position information produced by the two operators. In other words, it is the probability that the assignment of fj to operator 1 and gk to operator 2 is correct.

Formula 4.2.1 can be rewritten in a more convenient form (see appendix I):

$$(4.2.2)\quad P[fj,gk \mid v1,p1,v2,p2] = \frac{1}{1 + \sum_{\neg(r=j \,\&\, s=k)} \sum_{\neg(w=j \,\&\, x=k)} \sum_{\neg(y=j \,\&\, z=k)} \frac{P[v1|fr]}{P[v1|fj]} * Q[r,s] * \frac{P[v2|gx]}{P[v2|gk]} * Q[w,x] * \frac{P[p1,p2,fy,gz]}{P[p1,p2,fj,gk]}},$$

where Q[r,s] represents P[fr,gs|¬(fj,gk)]. This version of the formula is convenient because the important probabilities form ratios that can be easily evaluated from the density functions.

This formula can be extended to include N operators, but its utility is limited by the number of terms that have to be computed. One possible way around this drawback is to consider only the position information produced by the operators. The rationale behind this simplification is that the local value information produced by an operator should be used to determine the known alternative that has been matched, if it can do so with the desired confidence. Otherwise, combining the value information from two or three operators is not expected to add any new constraints that might distinguish between the possible alternatives. The position information of an operator, however, is expected to contribute new constraints if it is combined with the position information from other operators. Thus, if the program is trying to determine the best assignment for two operators, it can choose the assignment [fj,gk] that has the highest value of:

(4.2.3) $$P[fj,gk \mid p1,p2].$$

Since

(4.2.4) $$P[fj,gk \mid p1,p2] = \frac{P[fj,gk,p1,p2]}{P[p1,p2]}$$

and the denominator in (4.2.4) is the same for all j's and k's, the program can simply choose the assignment that has the highest value of P[fj,gk,p1,p2]. Once the best assignment has been found, the probability that it is the correct assignment can be computed by the following formula:

(4.2.5) $$P[fj,gk \mid p1,p2] = \frac{1}{1 + \sum_{\neg(x=j \;\&\; y=k)} \frac{P[p1,p2,fx,gy]}{P[p1,p2,fj,gk]}}.$$

Even formula 4.2.5 involves a large number of terms for a reasonable number of operators. For example, ten operators, each with two alternatives, require 1024 probabilities to be computed in order to determine the best assignment. This fact implies that formula 4.2.5 should be used to assign alternatives to small subsets of the operators and then combine these assignments to form the overall assignment. The probability that the complete assignment is correct can be estimated by a function of the probabilities associated with the subsets. It is even possible to incorporate some position information for each subset. For

example, the centroid of each subset can be used as the position of the subset. Then the overall probability that the assignment is correct can be based upon the structure of these subsets.

The subsets can be set up in advance or they can be created dynamically. If the subsets are created dynamically more and more information can be used to make the assignments, if necessary. The idea is to only use the position information of other operators if the local information is not sufficient to choose a known alternative with the desired confidence. The assignments in section 3.2 were made by choosing the known alternative that has the highest value of

(4.2.6) $$P[f_j \mid v_1] \qquad (1 \le j \le M).$$

This formula can be extended to include the position of the match (see appendix II):

(4.2.7) $$P[f_j \mid v_1, p_1] \qquad (1 \le j \le M).$$

If the confidence is still not high enough in the assignment, the position information from another operator can be included:

(4.2.8) $$P[f_j, g_k \mid p_1, p_2] \qquad (1 \le j \le M).$$

This process can continue until an assignment can be made with the desired confidence or the system gives up because of the amount of computation.

It is important to choose subsets that are expected to produce unique patterns. For example, consider figure 4.2.2, which shows the known alternatives for four operators. If the program is trying to assign one of the two alternatives to the results of operator 1, and more information is needed to distinguish between these two alternatives, which of the other operators should be used? Operator 2 may not be helpful because it is probably difficult to distinguish between the assignment [1a,2b] and [1b,2a]. If the initial constraints allow for a small angular uncertainty and a small scale uncertainty, it may be difficult to distinguish between the assignments [1a,4] and [1b,4]. However, even if the scale is allowed to change fifty percent and the orientation in the plane is completely unconstrained, it is still possible to distinguish between the assignments [1a,3a] and [1b,3a], or between [1a,3b] and [1b,3b]. Therefore, the subset should be operator 1 and operator 3.

Since the known alternatives for the operators are known in advance and the task constraints are known in advance, it is possible to estimate the distinctness patterns and choose the best subsets. This choice can be made at planning time.

The main ideas in this section are (1) the probability that an assignment is correct is

important because it determines whether or not the results of the least-squares routine should be believed, (2) the combination of the position information produced by several operators adds constraints that help determine correct assignments, (3) the uniqueness of a pattern of matches determines the pattern's contribution toward the assignment decision, and (4) the uniqueness of a pattern can be estimated at planning time since the alternatives for the operators are known at planning time.

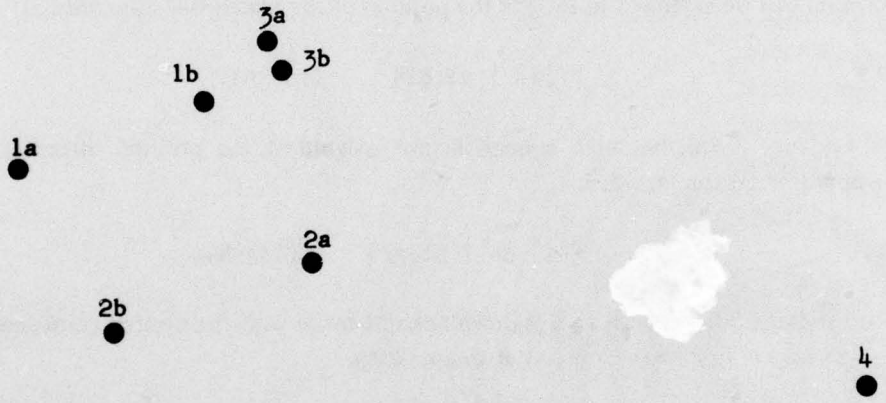


Figure 4.2.2

CHAPTER 5

PLANNING-TIME COMBINATION RULES

The goal of this chapter is to investigate ways to produce information that is useful to a *strategist*. In this context a strategist is a program (or possibly a person) that evaluates the various alternatives and develops a plan to achieve a particular goal. At one level a strategist might be trying to decide whether to use visual feedback or force feedback to check for a screw on the end of the screwdriver. At that decision point it needs information about the expected costs and reliabilities of the alternative methods (see [Taylor 76] and [Sproull 77] for descriptions of strategists and the information used to make such decisions). This chapter develops techniques to produce this type of cost and reliability estimates for verification vision.

The execution-time combination rules combine the results of sequentially applied operators and produce estimates for inspection confidences, precision, and precision confidences. These methods make it possible for the program to stop gathering information as soon as the desired confidence and precision have been reached. The underlying program structure is an *ordered* list of operators to be applied. The ordering criteria are important because some operators are more reliable than others, some operators contribute more than others, and some operators cost more to apply than others. This chapter investigates techniques to rank the operators according to their *expected* contributions and costs. It also presents techniques to estimate the *expected* number of features (and costs) required to achieve certain confidence and precision limits. Appendices IV and V demonstrate the application of some of these techniques.

Section 1
RANKING FEATURES BY VALUE

Consider the standard task of inspecting a scene to decide whether a screw is present or not. Section 3.1 developed a formula that reduces the value information from several operators into an overall confidence that the object is present. It also pointed out that the *contribution* of an operator is the value of the ratio:

$$\frac{P[vi|Off]}{P[vi|On]} \tag{5.1.1}$$

where vi is the value (or set of values) returned by the operator and On denotes the proposition that the object is present (see formula 3.1.12). For ranking purposes the logarithm of the inverse ratio,

$$\log\left(\frac{P[vi|On]}{P[vi|Off]}\right), \tag{5.1.2}$$

is more convenient. The greater the ratio, the better the contribution. The logarithm of the likelihood ratio is used because it is additive and there is a theorem (to be discussed in section 5.5) that uses it to estimate the number of operators required to reach a certain confidence.

At *planning time*, vi does not have a specific value. Operator i has only two density functions, one for its values when the screw is On and one for its values when the screw is Off. For ranking purposes one is interested in the expected value of the log-ratio, which depends upon the total density function for the values associated with operator i. The total density function is a weighted sum of the density functions for On and Off (as shown in figure 3.1.6). The weights are simply the corresponding *a priori* probabilities. Therefore,

(5.1.3) density(vi) = P[On]*On_density(vi) + P[Off]*Off_density(vi).

This is a valid density function since

(5.1.4)

$$\int_{-\infty}^{+\infty} \text{density}(X)dX = \int_{-\infty}^{+\infty} P[On]*On\_density(X)dX + \int_{-\infty}^{+\infty} P[Off]*Off\_density(X)dX$$

$$= P[On]*\int_{-\infty}^{+\infty} On\_density(X)dX + P[Off]*\int_{-\infty}^{+\infty} Off\_density(X)dX$$

$$= P[On] + P[Off] = 1.$$

The expected value can then be computed as follows:

$$\text{(5.1.5)} \qquad \text{expected\_log-ratio} = \int_{-\infty}^{+\infty} \text{log-ratio}(X) * \text{density}(X)\ dX.$$

MACSYMA (see [Mathlab Group 74]) was used to expand this integral symbolically, assuming that the density functions are normal. The derivation is given in appendix III. The result is a readily evaluated expression of the two means (M1 & M2), the two standard deviations (SD1 & SD2), and the *a priori* probability of On (i.e., P):

$$\text{(5.1.6)} \quad \text{expected\_log-ratio} = \log(SD2) - \log(SD1) + 1/2 - P + P*\frac{SD1^2 + (M2 - M1)^2}{2 * SD2^2} - (1-P)*\frac{SD2^2 + (M2 - M1)^2}{2 * SD1^2}.$$

Later sections will also need estimates for the expected log-ratio, given either On or Off. The expected log-ratio, given On, can be computed as follows:

$$\text{(5.1.7)} \qquad \text{ELR\_given\_On} = \int_{-\infty}^{+\infty} \text{log-ratio}(X) * On\_density(X)\ dX.$$

The integral can be expanded to produce

$$(5.1.8)\quad ELR\_given\_On = \log(SD2) - \log(SD1) + \frac{SD1^2 + (M2 - M1)^2}{2 * SD2^2} - \frac{1}{2}.$$

Similarly, the expected log-ratio, given Off, can be expressed as

$$(5.1.9)\quad ELR\_given\_Off = \log(SD2) - \log(SD1) + \frac{1}{2} - \frac{SD2^2 + (M2 - M1)^2}{2 * SD1^2}.$$

Since the expected log-ratio for an operator represents the operator's average contribution, operators that have large expected log-ratios should be applied first in order to minimize the number of operators used to reach a certain confidence limit. Thus, a simple operator-ranking scheme is to compute the expected log-ratio for each of the operators and then rank them according to their expected value (largest first).

Section 2
KNOWN ALTERNATIVES AND SURPRISES

The method used in the last section can also be used to compute the expected contributions for operators that have several *known alternatives* and/or are subject to *surprises*. However, it is quite difficult to expand symbolically the integrals that express the expected value. A numerical technique is used instead.

Formula 3.3.9 expresses the probability that the object is present, given the values of several operators, each of which may have several known alternatives and surprises. That formula is

(5.2.1)

$$P[On|v1,v2,\ldots vK] = \frac{1}{1 + \frac{P[Off]}{P[On]} * \prod_{j=1}^{K} \left(\frac{P[On]}{P[Off]} * \frac{\sum_{0 \le i \le Nj} P[vj|gj,i]*P[gj,i]}{\sum_{0 \le i \le Mj} P[vj|fj,i]*P[fj,i]} \right)},$$

where fj,1; fj,2; ... fj,Nj are the Nj known alternatives for jth operator when On is true, gj,1; gj,2; ... gj,Mj are the Mj known alternatives for jth operator when On is false, fj,0 is the surprise for jth operator when On is true, and gj,0 is the surprise for jth operator when On is false. The *contribution* of the jth operator toward the overall probability is:

(5.2.2)
$$\left(\frac{P[On]}{P[Off]} * \frac{\sum_{0 \le i \le Nj} P[vj|gj,i]*P[gj,i]}{\sum_{0 \le i \le Mj} P[vj|fj,i]*P[fj,i]} \right).$$

For ranking purposes the logarithm of the inverse of this ratio is used:

(5.2.3)
$$\text{log-ratio}(vj) = \log\left(\frac{P[Off]}{P[On]} * \frac{\sum_{0 \le i \le Mj} P[vj|fj,i]*P[fj,i]}{\sum_{0 \le i \le Nj} P[vj|gj,i]*P[gj,i]} \right).$$

The expected value can again be computed by

(5.2.4)
$$\text{expected\_log-ratio} = \int_{-\infty}^{+\infty} \text{log-ratio}(X) * \text{density}(X)\, dX,$$

where the density depends upon all of the known alternatives and surprises. Since

(5.2.5)
$$P[On] = P[fj,0] + P[fj,1] + \ldots P[fj,Nj]$$
$$\text{and } P[Off] = P[gj,0] + P[gj,1] + \ldots P[gj,Mj],$$

the density for operator j is

(5.2.6)

$$\text{density}(X) = \sum_{i=0}^{Mj}(P[fj,i]*\text{density}(fj,i)) + \sum_{i=0}^{Nj}(P[gj,i]*\text{density}(gj,i)).$$

Thus, if ELR denotes the expected log-ratio for the jth operator, then

(5.2.7)

$$ELR = \int_{-\infty}^{+\infty} \log\left(\frac{P[Off]}{P[On]} * \frac{\sum_{0\le i\le Mj} P[vJ|fj,i]*P[fj,i]}{\sum_{0\le i\le Nj} P[vJ|gj,i]*P[gj,i]} \right) * \text{density}(X)dX$$

or

(5.2.8) ELR =

$$\log(P[Off])-\log(P[On]) + \int_{-\infty}^{+\infty} \log\left(\frac{\sum_{0\le i\le Mj} P[vj|fj,i]*P[fj,i]}{\sum_{0\le i\le Nj} P[vj|gj,i]*P[gj,i]} \right)*\text{density}(X)dX.$$

The logarithms of the sums could be expanded in a Taylor series in order to integrate this expression symbolically, but it is simpler to use a numerical integration technique to approximate the value for a specific value of the operator. High-precision values are not needed because they are only used to rank the operators and predict the expected number of operators required to achieve a certain confidence in On.

It is not necessary to integrate the function from minus infinity to plus infinity. As discussed in section 3.3 each alternative defines an interval of reasonable values: plus or minus three standard deviations about its expected value. It is only necessary to integrate over the range that is the union of all of the intervals for the individual alternatives.

The result of this section is a set of formulas that compute the expected contribution of an operator, even if it may involve several known alternatives and surprises. These expected contributions will be used in later sections to compute other important quantities.

Section 3
COST INFORMATION

Since different operators cost different amounts to apply, a slightly more sophisticated ranking scheme can rank the operators according to a cost-adjusted version of their expected contribution, i.e., the expected value of

$$\frac{\langle \text{log-ratio} \rangle}{\langle \text{cost} \rangle}. \tag{5.3.1}$$

If it is difficult to compute the expected value of that ratio, it can be approximated by

$$\frac{\langle \text{expected log-ratio} \rangle}{\langle \text{expected cost} \rangle}. \tag{5.3.2}$$

The cost of applying an operator could involve such factors as training time, computation time, and memory space, but in this discussion, for simplicity, the expected cost of an operator is defined to be the expected computation time required to locate a *match*.

Computation time is a function of several variables: (1) the initialization time, (2) the number of times the operator is applied, and (3) the computation time for each application. If an operator is applied over a complete region (e.g., the tolerance region about a known alternative), it is relatively easy to predict the expected cost. However, if an operator is sequentially applied in a region (using some search strategy) until a *reasonably good* match is found, one has to predict the number of separate applications to be used to find such a match. This prediction is a little more difficult. It is based upon the type of feature, the expected distributions of the feature and its alternatives, and the local characteristics of the operator (e.g., the size of the region covered by one application). Each feature-operator-strategy triple needs a separate mechanism to predict the average number of applications required to find a match.

An operator-ranking scheme that incorporates cost estimates is: compute the benefit-cost ratios (as in formula 5.3.1) for each of the operators and order them according to the largest first.

Section 4
LEAST-SQUARES CULLING

As mentioned in section 3.5 the least-squares culling routine requires a minimum number of matches. Let M represent this minimum number. Let N be the number of operators that must be applied in order to find M matches. Since an operator may or may not locate a known alternative (i.e., a match), N is greater than or equal to M. This section develops a method to predict N, given M and an ordered list of operators. The following sections derive methods to estimate the *expected number of operators* required to achieve a certain goal. It should be pointed out that it is possible to estimate these numbers by simply applying the operators to several training pictures and averaging the number of operators needed to reach the desired goal. Often this direct way is the best way to proceed. However, sometimes it is useful to be able to produce an independent estimate of the expected number of operators. Techniques to produce these independent estimates will be discussed in the following sections.

In order to predict the average number of operators required to locate M matches it is necessary to compute each operator's expected contribution toward M. Consider figure 5.4.1. Figure 5.4.1.a shows the possible matches associated with a typical operator: three known alternatives and a surprise (f1, f2, f3, and S). Assume that the *a priori* probabilities for these possibilities are:

(5.4.1)
$$P[f1] = .5,$$
$$P[f2] = .2,$$
$$P[f3] = .1,$$
$$\text{and} \quad P[S] = .2.$$

Figure 5.4.1.b shows the densities associated with the various possibilities, but they are scaled by their *a priori* probabilities of occurrence. Figure 5.4.1.c shows the weighted density function for the operator. That is,

(5.4.2)
$$\text{density}(X) = P[S]*\text{density}(S) + \sum_{j=1}^{3}(P[fj]*\text{density}(fj)).$$

Given a specific value for the operator, the best alternative is the alternative with the highest probability of being the correct match, i.e.,

(5.4.3)
$$\text{MAX}(P[f1|v],\ P[f2|v],\ P[f3|v],\ P[S|v]).$$

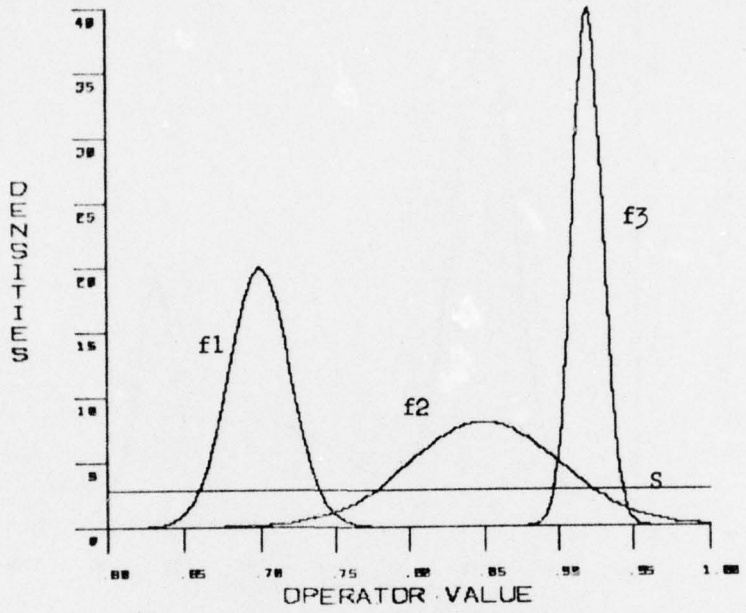


Figure 5.4.1.a

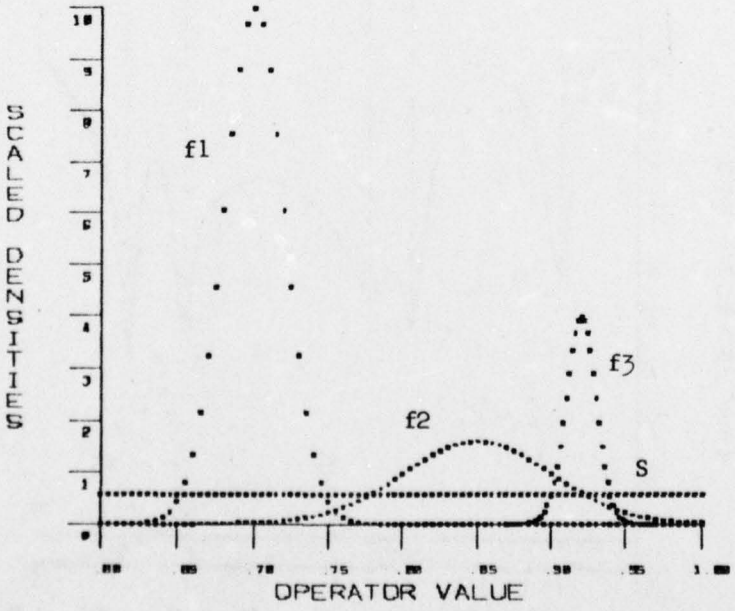


Figure 5.4.1.b

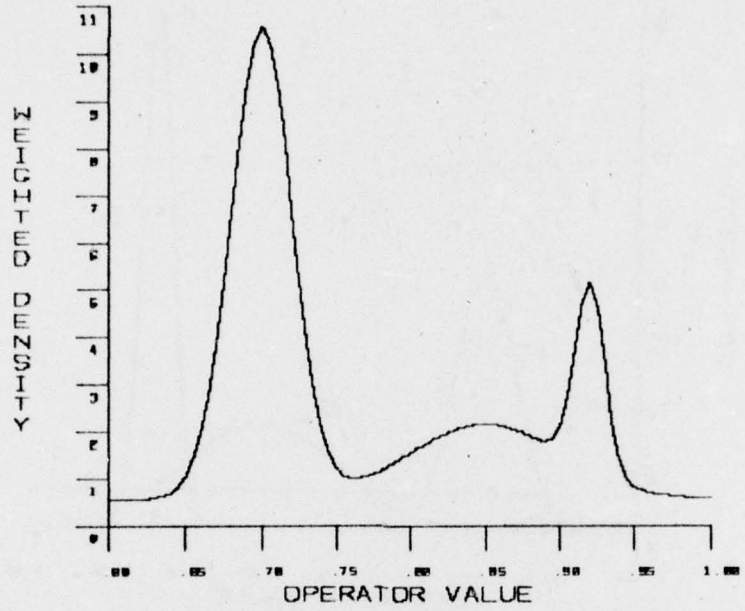


Figure 5.4.1.c

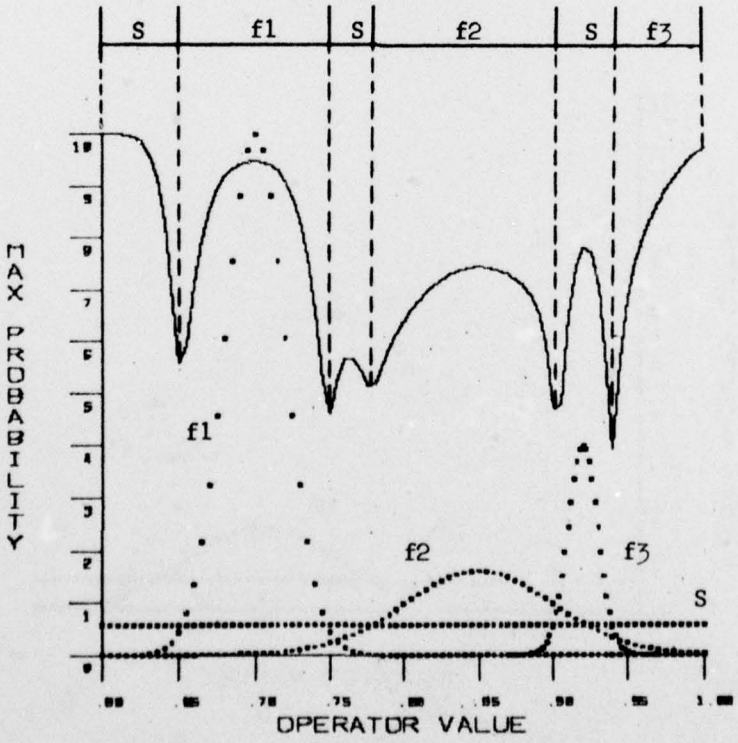


Figure 5.4.1.d

The algorithm shown in figure 3.5.2 uses the probability associated with the best match as the operator's contribution toward the goal of M matches, except when the operator's value is *unusual* or it suggests that a *surprise* is the best match. In the case of an unusual value or surprise, no contribution is credited to the operator. Figure 5.4.1.d superimposes the graph of the operator's contribution (scaled by a factor of 10) on top of the scaled densities shown in figure 5.4.1.b. Figure 5.4.1.d also labels each interval with the name of the possibility that would be returned as the best match. Notice that there are three intervals that imply that the surprise is the best match.

The expected contribution of an operator toward M (abbreviated EC) can be computed in the standard way:

$$EC = \int_{-\infty}^{+\infty} \langle \text{contribution at } X \rangle * \text{density}(X)\, dX \qquad (5.4.4)$$

where

$$\langle \text{contribution at } X \rangle = \begin{cases} \emptyset & \text{(if unusual or surprise)} \\ \text{MAX}(P[f1|X], \ldots P[fn|X]) & \text{(otherwise).} \end{cases} \qquad (5.4.5)$$

Again a numerical integration technique is the easiest way to compute the value of EC.

Formula 5.4.4 is important because it computes the expected contribution of an operator. Given an ordered list of operators and their expected contributions it is possible to estimate the number of operators that have to be applied in order to locate M matches. The expected number of operators is the minimum N such that

$$\sum_{j=1}^{N} \langle \text{operator } j\text{'s expected contribution} \rangle \geq M. \qquad (5.4.6)$$

This value of N is the planning stage's prediction of the number of operators to be needed at execution time. At execution time the actual contributions of the operators can be used to decide when a sufficient number of matches have been found. For example, if the theoretical minimum number of matches required by the least-squares routine is two, then M is two. If the planning stage predicts that each of the operator/feature pairs will contribute .5 toward the required number of reliable matches, N would be four. At execution time, when the operators are applied to a test picture, if each operator happens to locate a best match with a probability of .75, then the first three operators are sufficient. If the user wants to be

particularly conservative about including the structural consistency information, he can increase the minimum number, M, from two to three. This increase forces the program to apply more operators before incorporating the position information, which means that there is a smaller chance that the least-squares routine will produce an incorrect precision estimate (as discussed in section 4.2).

Section 5
INSPECTION

In an inspection task each operator contributes a certain amount toward increasing (or decreasing) the overall confidence that the proposition On is true. Sections 5.1 and 5.2 developed techniques to compute the *expected* contribution of an operator. Given the expected log-ratio (i.e., the expected contribution) of each operator, what is the expected number of operators required to generate a certain confidence in On? The answer to this question is based upon a theorem in sequential pattern recognition [Andrews 72]:

THEOREM: Let e(On) be the error rate allowed for saying that On is true when it really is false and let e(Off) be the error rate allowed for incorrectly saying that Off is true when it really is false. Let

$$A = \frac{1 - e(Off)}{e(On)} \quad \text{and} \quad B = \frac{e(Off)}{1 - e(On)}.$$

Then, given that On is true, the expected number of operators to be used to make a decision is given by

$$expected\_\#(On) = \frac{(1-e(Off))*\log(A) + e(Off)*\log(B)}{\langle \text{average log-ratio, given On} \rangle}.$$

And given that Off is true, the expected number of operators to be used to make a decision is given by

$$expected\_\#(\neg On) = \frac{e(On)*\log(A) + (1-e(On))*\log(B)}{\langle \text{average log-ratio, given Off} \rangle}.$$

And finally, the expected number of operators to achieve the specified error rates is

$$expected\_\# = P(On)*expected\_\#(On) + P(Off)*expected\_\#(Off).$$

The theorem is based upon the assumption that there are an infinite number of operators whose average log-ratios are known. However, there are only a finite number of operators (usually on the order of ten) for any specific VV task. The theorem can still be used to produce an approximate number of operators expected by assuming that there are an infinite number of operators with the contribution of the best operator. If that were the case, how many operators would be needed? If the answer is one or less, then the best operator will probably be sufficient, on the average. If the answer is more than one, consider the average of the first two operators and compute the number needed if there were an infinite number of operators with that expected ratio. If the answer is less than or equal to two, the best two operators will be enough on the average, etc. Figure 5.5.1 lists eight operators and their expected log ratios. Using those operators and a goal of $e(On) = e(\neg On) = .05$, the expected number of operators would be one. The expected number of operators to achieve $e(On) = e(\neg On) = .005$ would be three.

1. 2.9
2. 2.1
3. 2.0
4. 1.7
5. 1.6
6. 1.4
7. 1.2
8. 1.2

Expected Log-likelihood Ratios

Figure 5.5.1

This theorem is powerful because it provides a way to predict the number of features, on the average, that will be necessary to achieve a specific confidence. The theorem applies to all operators whether or not they have several known alternatives and/or surprises. The only effect of alternatives and surprises is to reduce the expected contribution of the operator.

Section 6
PRECISION

Chapter 4 developed a method to locate objects (in the domain of VV). The method divided a location task into three subtasks:

(1) locate enough features to be able to apply the least-squares culling routine (this set of features is referred to as the *kernel*),

(2) locate enough additional features to produce the desired precision about the point of interest,

and (3) locate enough additional features to develop the required amount of confidence in the statement of precision.

In order to predict the total number of operators needed in a location task, one needs estimates for each of the subtasks. Section 5.4 developed a method to predict the expected number of operators required in subtask (1). This section and the next section will develop methods to predict the expected number of operators required by subtasks (2) and (3), respectively.

There is a precision associated with each operator/feature pair. For example, given an edge operator and a specific line to be found, the edge operator will locate a point on the line within some precision. Given a different line (maybe a fuzzy line), the precision of the same edge operator will probably be different. In fact, the precision of most operators also depends upon the *type* and *amount* of change between the planning picture and the test picture (e.g., the amount of rotation or the change in the overall light level). In order to predict the number of operators needed to locate a point of interest within a predeclared precision, it is necessary to model the precision of each operator. A statistical model is convenient because it provides variance information for each piece of position information. Given the variances about the position information, the weight matrix (i.e., W) can be constructed. The least-squares routine uses the weight matrix to determine the variances about the resulting parameter values and the points of interest.

The modelling of the position information is simplified in VV because there are no large unknown changes between the planning picture and the test picture. The variation in the appearance of a feature is limited. The main factors that affect the precision of an operator are: (1) the inherent operator characteristics (e.g., its maximum resolution), (2) the local feature characteristics (e.g., fuzziness), and (3) small rotations (e.g., 15 degrees). Often the

inherent characteristics of an operator are the dominant factors involved in determining the precision of the operator. If that is true, an *a priori* estimate can be used to model the precision. If that is not true or if a better estimate is desired, it is possible to apply the operator (in conjunction with several other operators with known precisions) to several trial pictures and produce an estimate for the precision of the operator/feature pair.

Given a statistical model for each operator/feature pair, approximately how many operators will be needed by the least-squares routine to produce the desired precision? A property of the least-squares routine makes it possible to answer this question in a straightforward way. If the least-squares routine is applied to a set of N operator/feature results, it produces essentially the same precision no matter what the actual position values are, as long as they conform to the stated variances. Therefore, if the routine is applied to the position information produced by N operator/feature pairs when they are applied to two different trial pictures, it will return approximately the same precision about the point of interest. The precisions are approximately equal because they mainly depend upon the relative structure of the features and the expected variances, both of which remain the same from trial to trial. This property is the basis for a procedure to predict the number of features needed to reach a certain precision: given a trial picture, locate a kernel set of matches, and apply the least-squares technique; if the resulting precision is sufficient, stop and return the number of operators used as the expected number operators to be needed; if the precision is not sufficient, locate another match, apply the least-squares routine, and repeat the precision check.

This procedure estimates the number of operators required to produce a certain precision. This estimate is an important piece of information for a strategist. If the least-squares routine requires a minimum of two matches before it can be applied, and if the expected number of matches required to reach the desired precision is six, a strategist has to decide when to apply the least-squares routine. One strategy is to locate six matches and then apply the routine. Another strategy is to locate two matches, apply the routine, use the results to reduce the searching required to find the third match, find the third match, apply the routine, etc. The second strategy may be more efficient in terms of the number of operators used, but less efficient in terms of the amount of computation time. It may be able to stop after four matches, but each application of the least-squares routine requires a certain amount of computation time. It is not clear which strategy is better. Sproull has investigated this type of problem and has implemented a strategist based upon decision theory that can decide which strategy is better [Sproull 77]. This chapter mainly develops techniques to estimate the quantities that are of interest to a strategist like his.

Section 7
CONFIDENCE IN THE PRECISION

As mentioned in section 4.2 it is often reasonable to assume that the location estimate for an object and the precision about that estimate produced by the least-squares routine is correct. Under that assumption, the expected number of operators required for a location task is the same as the expected number of operators needed to reach the desired precision. If the assumption is not true, it is possible to use a method similar to the one used in section 5.5 to estimate the expected number of operators required to reach a certain confidence in an inspection task.

Formula 4.2.7 indicates what the contribution of an operator is toward the overall confidence in a precision. Given this symbolic expression for the contribution, it is possible to employ a numerical integration routine to compute the *expected* contribution from an operator. The sequential pattern recognition theorem referred to in section 5.5 can be applied again. Given the expected contributions for the individual operators, the theorem produces the expected number of operators needed to reach a certain confidence in a precision.

Given this technique, the general prediction scheme for location tasks can be stated as follows: determine the expected number of operators required to achieve the desired precision, determine the expected number of operators required to reach the desired confidence in that precision and return that number as the expected number of operators required to accomplish the task. For example, if the expected number of operators required to achieve the precision is four, and if the expected number required to develop the confidence is six (i.e. an additional two operators are needed), then predict that six operators will be needed for the task.

Section 8
EXPECTED COST

Given an ordered list of operators and the expected number of operators required to achieve a certain goal (i.e., N), it is easy to produce an estimate for the expected cost associated with achieving the goal: sum the expected costs for the first N operators, i.e.,

$$(5.8.1) \qquad \sum_{j=1}^{N} \langle \text{expected cost of operator } j \rangle .$$

This expression is just a rough estimate for the expected cost because it assumes that the expected cost is the sum of the expected costs for the expected number of operators, which is not generally true. A better estimate is:

$$(5.8.2) \qquad \sum_{j=0}^{\infty} (P[Aj] * Cj),$$

where Aj means that the goal is achieved after applying operators one through j and Cj denotes the expected cost of applying the first j operators.

CHAPTER 6

PROGRAMMING TIME AND TRAINING TIME

There are four stages in the development of a VV program: programming, training, planning, and execution (see figure 6.1). There are also four types of information that are passed from one stage to the next:

(1) A statement of the task, i.e. the feature of interest, a set of constraints on the objects in the scene, and the goal.

(2) A camera calibration, i.e. the transformation between the camera's screen coordinate system and the workstation coordinate system.

(3) A list of operator/feature pairs. Each pair contains a feature that is part of one of the objects in the scene and an operator that can locate the image of the feature in a picture of the scene.

(4) A set of statistics for the operators, i.e. statistics that describe the reliability of the operators and the distribution of values produced by the operators.

The execution-time and planning-time chapters described the combination rules, assuming that all of this information already existed. This chapter describes how this information is produced. Each type of information will be discussed in a separate section. The discussions are based upon several example VV tasks. Annotated traces for two of these tasks can be found in appendices IV and V. The traces show the capabilities of the current implementation of the system.

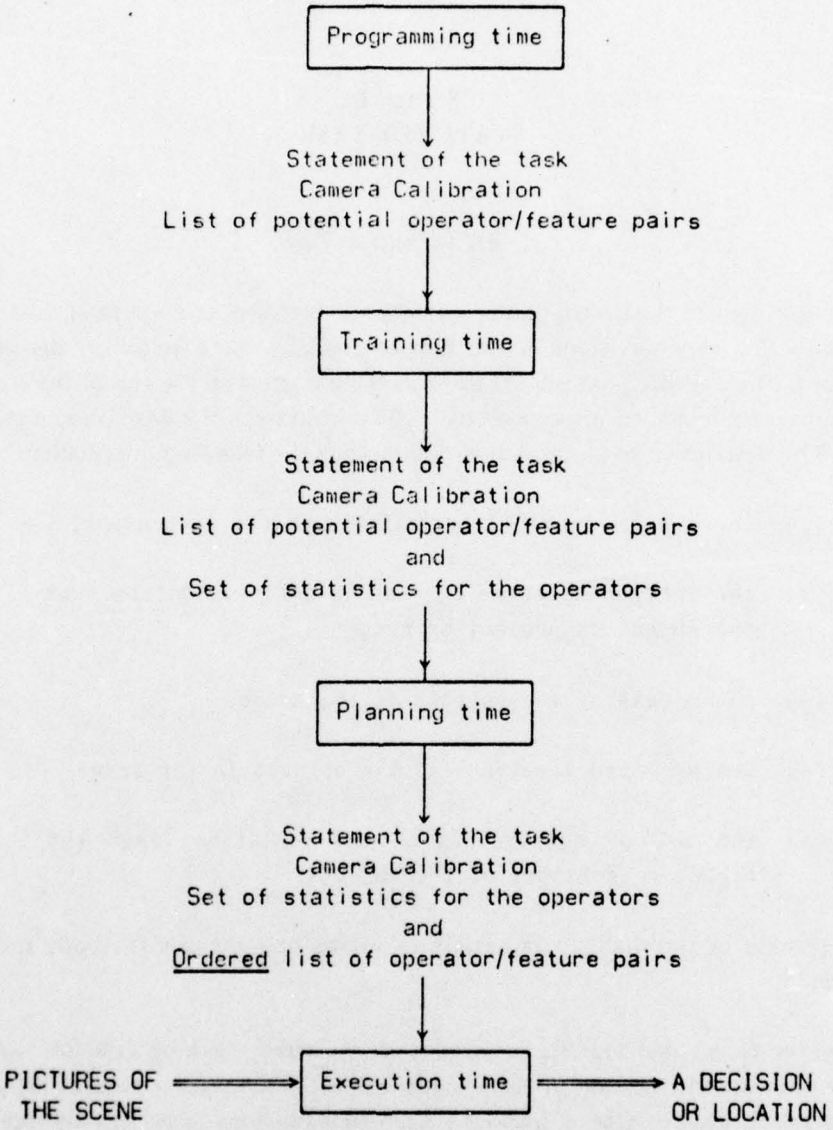

Programming time
Statement of the task
Camera Calibration
List of potential operator/feature pairs
Training time
Statement of the task
Camera Calibration
List of potential operator/feature pairs
and
Set of statistics for the operators
Planning time
Statement of the task
Camera Calibration
Set of statistics for the operators
and
Ordered list of operator/feature pairs
PICTURES OF THE SCENE
Execution time
A DECISION OR LOCATION

Figure 6.1

Section 1
STATE THE TASK

1. An Inspection Task

The first step in the description of a task is the statement of the *type* of task: inspection or location. For example, consider the task of checking for a screw on the end of the screwdriver. The ultimate goal is a *yes* or *no*: Is there a screw on the end of the screwdriver? This goal makes the task an inspection task. The remainder of the description depends upon this fact. The description of an inspection task includes the following information:

(1) the *a priori* probability that the object is present,

(2) the thresholds to be reached in order to decide that the object is present or not,

(3) the models of the objects in the scene,

(4) the expected locations of the objects in the scene,

and (5) the set of constraints on the deviations from the expected locations of the objects.

The first portion of the dialog in appendix IV shows how the user currently specifies this information.

In order to set the *a priori* probability in the screw-checking task, the user has to decide from his experience what percentage of the time the screwdriver successfully picks up a screw from the dispenser. Does it pick up a screw nine times out of ten or only five times out of ten? The lower the *a priori* probability, the more information the program has to gather to decide that the screw is present.

The second part of an inspection task description is a statement of the desired thresholds. In order to decide that the screw is present, the program has to raise the probability that the screw is present above a certain threshold. To decide that the screw is not present the system has to lower the probability that the screw is present below a certain threshold. The first threshold will be referred to as the *yes threshold* and the second will be referred to as the *no threshold*. The user has to determine the expected consequences of a

wrong decision and set the thresholds accordingly, taking into account the fact that more stringent thresholds are harder to achieve. For example, if the arm jams the screw dispenser if it tries to pick up a screw when it already has one, the *no threshold* should be set to a low value, e.g. .0005. This value would force the VV system to be quite sure that a screw is not on the end before trying to pick up another one.

Sometimes these thresholds will be referred to as the *confidences* to be reached by the system. In these terms the system increases its confidence in a *no* decision as it lowers the probability that the object is present.

The third part of a description is a set of object models. An object model is used to state the set of constraints that limit the possible locations of the object and to describe the structure of the features that the system can use to locate the object.

The current object models are simply a three-dimensional reference coordinate system and a structured set of features defined in terms of that coordinate system. Features are added as they are needed to describe constraints or as they are discovered during the programming process. For example, in the screw-checking task, the screw is initially represented as a coordinate system whose origin is located at the tip of the screw. As features of the screw, such as the corner formed by the head of the screw and side of the shaft, are investigated they are added to the model.

The fourth part of a task description is a list of the expected locations for all of the objects of interest in the scene. For the screw-checking task this simply consists of an expected position and an expected orientation of the screw:

```
(6.1.1)  The expected position of the screw (i.e., the tip of the
         screw) is (60 cm, 45 cm, 5 cm).

         The expected orientation of the screw is ROT(X, 180*DEG),
         i.e., vertical, pointing down.
```

The expression, ROT(X, 180*DEG), is an AL expression for an orientation (see [Finkel 74]). It means that the screw coordinate system is formed by rotating the workstation coordinate system 180 degrees about its X-axis. Since the Z-axis of the workstation points up, the Z-axis of the screw points down.

The fifth and final part of a task description is a set of constraints that state the location restrictions associated with the objects in the scene. The following statement is a typical set of constraints for the screw-checking task:

(6.1.2) Let α represent the angle between the z-axis of the screw and the z-axis of the workstation. Then the maximum deviations from the expected positions are:

$$-1.2 \text{ cm} \leq dx \leq 1.2 \text{ cm}$$
$$-1.2 \text{ cm} \leq dy \leq 1.2 \text{ cm}$$
$$-.8 \text{ cm} \leq dz \leq .8 \text{ cm}$$
$$-10 \text{ deg} \leq d\alpha \leq 10 \text{ deg}.$$

Given the location of a feature in the screw coordinate system, these constraints are sufficient to determine the three-dimensional volume that represents the possible positions for the feature. Given this volume and a camera calibration, it is possible to determine the tolerance region of the feature in the screen coordinate system.

The constraints mentioned above are all stated directly in terms of the location of the screw even though they are the result of several other uncertainties, such as the accuracy of the arm. The user is responsible for combining the various inaccuracies and producing the final set of constraints associated with the location of the screw. It would be significantly better if the user could state the basic inaccuracies and have the system compute the resulting constraints on the screw. For example, the screw is on the end of the screwdriver, which is held in the hand of the arm. In this arrangement there are three places for uncertainties: the arm (i.e., the placement of the hand), the linkage between the hand and the screwdriver, and the linkage between the end of the screwdriver and the screw. Typical expected locations and constraints on these three sources of uncertainty are:

(6.1.3) The expected position and orientation of the hand in the workstation's coordinate system is (60 cm, 45 cm, 15 cm) and ROT(X, 180*DEG).

The expected position and orientation of the screwdriver in the hand's coordinate system is (0 cm, 0 cm, 8 cm) and ROT(X, 0*DEG).

The expected position and orientation of the screw in the screwdriver's coordinate system is (0 cm, 0 cm, 2 cm) and ROT(X, 0*DEG).

Let $\alpha 1$ represent the angle between the z-axis of the hand and the z-axis of the workstation. Then the maximum deviations about the location of the hand due to the inaccuracies of the arm are:

$$-.6 \text{ cm} \leq dx \leq .6 \text{ cm}$$
$$-.6 \text{ cm} \leq dy \leq .6 \text{ cm}$$
$$-.4 \text{ cm} \leq dz \leq .4 \text{ cm}$$
$$-4 \text{ deg} \leq d\alpha 1 \leq 4 \text{ deg}.$$

Let $\alpha 2$ represent the angle between the z-axis of the screwdriver and the z-axis of the hand. Then the maximum deviations about the location of the screwdriver due to the inaccuracies of the grasping operation are:

$$-.1 \text{ cm} \leq dx \leq .1 \text{ cm}$$
$$-.1 \text{ cm} \leq dy \leq .1 \text{ cm}$$
$$-.1 \text{ cm} \leq dz \leq .1 \text{ cm}$$
$$-3 \text{ deg} \leq d\alpha 2 \leq 3 \text{ deg}.$$

Let $\alpha 3$ represent the angle between the z-axis of the screw and the z-axis of the screwdriver. Then the maximum deviations about the location of the screw due to the inaccuracies of the attachment of the screw to the screwdriver are:

$$-.1 \text{ cm} \leq dx \leq .1 \text{ cm}$$
$$-.1 \text{ cm} \leq dy \leq .1 \text{ cm}$$
$$-.1 \text{ cm} \leq dz \leq .1 \text{ cm}$$
$$-3 \text{ deg} \leq d\alpha 3 \leq 3 \text{ deg}.$$

A constraint resolving system can reduce this series of expected locations and constraints to an expected position for the screw and a single set of constraints on the screw, such as the set stated in (6.1.2). Taylor's constraint resolving system provides this type of capability for position deviations and *small* angle deviations, where small angles are defined to be five degrees or less [Taylor 76].

2. *A Location Task*

Consider the task of locating the screw dispenser, which is sitting upright on the table. It is a location task. The goal of the task is to locate the *pick-up* point on the screw dispenser so the arm can insert the tip of the screwdriver and pick up a screw. Since it is a location task, the statement of the task consists of

(1) an object model of the screw dispenser,

(2) an expected location for the screw dispenser,

(3) the known constraints on the location of the screw dispenser,

and (4) the desired constraints on the pick-up point.

The initial model for the screw dispenser contains the origin of the coordinate system and the location of the pick-up point. A typical expected location for the screw dispenser is (55 cm, 50 cm, 0 cm) and ROT(Z, 180*DEG) in the workstation coordinate system.

Since the dispenser is known to be sitting upright on the table, there are only three remaining degrees of freedom: X, Y, and α, where α is a rotation about the z-axis of the workstation. Typical initial constraints on the dispenser are:

(6.1.3)
$$-1.0 \text{ cm} \leq dx \leq 1.0 \text{ cm}$$
$$-1.0 \text{ cm} \leq dy \leq 1.0 \text{ cm}$$
$$-5 \text{ deg} \leq d\alpha \leq 5 \text{ deg}.$$

Typical desired constraints on the location of the pick-up point are:

(6.1.4)
$$-.1 \text{ cm} \leq dx \leq .1 \text{ cm}$$
$$-.1 \text{ cm} \leq dy \leq .1 \text{ cm}.$$

Thus, the goal of this task is to reduce the uncertainties associated with the screw dispenser so that the location of the pick-up point is known within the desired tolerances.

All of the constraints mentioned above assume that the camera is positioned once, calibrated, and left in a fixed position for all of the assemblies. If that is not the case, the constraint computations need to include the uncertainties associated with the repositioning of the camera.

Section 2
POSITION AND CALIBRATE THE CAMERA

1. Set up for a Location Task

There are two main considerations that enter into the placement of the camera:

(1) Can several features on the objects of interest be seen?

and (2) Is the resolution of the picture high enough for the program to produce the desired precision?

Both requirements have to be met in order for the placement to be acceptable.

Intuitively the first requirement seems easy to meet: place the objects in their expected locations and then manually move the camera around until a sufficient number of features are in view. However, even for this requirement there are several complicating factors. For example, there have to be a sufficient number of visible features at all possible locations for the objects, not just their expected locations. If the uncertainty about the location of an object includes an unknown rotation of plus or minus ten degrees, the VV system needs to be able to see enough features at all angles within that twenty-degree range.

The resolution of the picture is important because (a) it may not be possible to achieve the desired precision if the resolution is too low and (b) the amount of searching may be excessive if the resolution is too high. Thus, if the resolution is properly adjusted for a task, the program can locate the feature of interest within the desired precision without requiring an unnecessary amount of searching.

The VV system can help the user find a good location for a camera if it is given several pieces of information in addition to a camera calibration:

(1) the expected locations for the objects in the scene,

(2) the known constraints on the objects,

(3) a list of potential features,

(4) the feature of interest,

(5) the desired precision about the feature of interest,

(6) the types of operator/feature pairs to be used,

and (7) the expected resolution of the operators.

The first three pieces of information can be used to determine the three-dimensional volume that should be visible to the camera (see the trace in appendix IV). The user roughly points out several potential features, indicates which objects they are a part of, and the system uses the set of constraints to sweep out the total volume that they might appear in. That volume will be referred to as the *tolerance volume for the task*. It is essentially the union of the tolerance volumes associated with the individual features. For example, consider figure

Figure 6.2.1.a

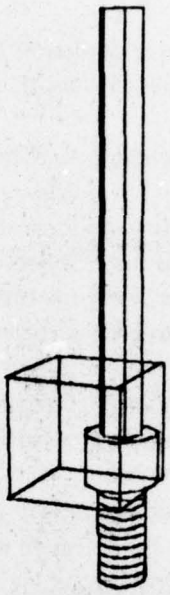


Figure 6.2.1.b

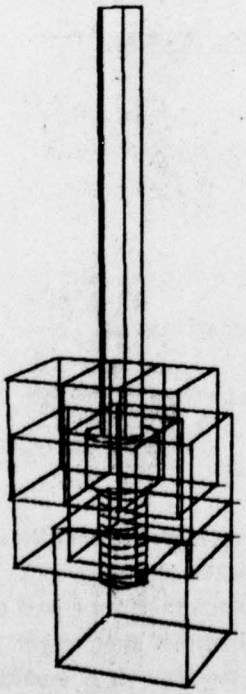


Figure 6.2.1.c

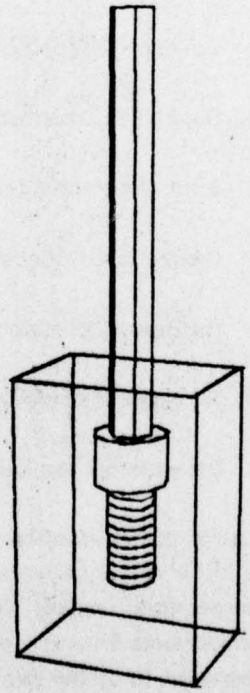


Figure 6.2.1.d

6.2.1.a, which shows a model of a screw on the end of a model of a screwdriver. Figure 6.2.1.b shows one feature and its tolerance volume. Figure 6.2.1.c shows several features and their tolerance volumes. Figure 6.2.1.d shows the task tolerance volume.

Given a camera calibration and the tolerance volume of the task, it is easy to check if the volume is included in a picture: project the volume onto the screen of the camera and see if that region is contained in the picture. (The projection of the tolerance volume for a task is the *tolerance region for the task.*)

Thus, if the tolerance region of a task is contained in the picture, the potential features will be visible unless they are obscured by other objects in the scene. Currently the user is responsible for deciding whether or not objects in the scene will obscure the potential features. Eventually the program will be able to simulate the movement of the objects in a scene and decide automatically whether or not the features are visible throughout the full range of uncertainties for the objects.

Items (1), (2), and (4) can be combined to produce the known precision about the feature of interest. If the initial constraints happen to imply that the initial precision is greater than the desired precision, there is no need to apply any operators; the feature is already known within the desired precision. In general, however, an increase in precision is desired and a sequence of operators has to be applied to gather position information.

The expected resolution of each operator is important because it determines the expected contribution of the operator toward the overall precision. Given the two-dimensional resolution of an operator in the screen coordinate system, it is possible to determine the corresponding resolution in the workstation coordinate system. For example, if a correlation operator can reliably locate a corner within one pixel and if the height of the corner is known, it is possible to convert the one-pixel uncertainty into workstation X and Y uncertainties, such as

(6.2.1)
$$-.07\text{cm} \le Cx \le +.07\text{cm}$$
$$-.12\text{cm} \le Cy \le +.12\text{cm},$$

where Cx is the operator's best estimate for the X coordinate of the corner and Cy is the operator's best estimate for the Y coordinate of the corner.

A set of these uncertainties can be used directly by the least-squares fitting routine to determine the expected precision to be produced by a reasonable number of operators, such as five or ten. This expected precision can then be compared with the desired precision. If it is less than the desired precision the resolution of the picture has to be increased. If it is much greater than the desired precision, the resolution of the picture can be reduced.

Figure 6.2.2 shows an iterative method for finding a camera location that satisfies both the visible feature requirement and the resolution requirement. The basic idea is to place the camera at some reasonable position, adjust the focal length so that the precision requirement can be met, and then locally adjust the position so that enough features are in view.

An improvement in this algorithm can be made by incorporating the following interactive technique:

(1) Intentionally include more of the workstation than is probably necessary in the first picture.

(2) Use a split screen display; show the first picture, which is the wide-angle view of the scene, on the left side and show a picture taken at another camera location, which is hopefully better than the first, on the right (see figure 6.2.3.a).

(3) Have the system superimpose the tolerance region of the task on top of the first picture. Since the first picture was designed to cover more of the scene than is necessary, the tolerance region of the task should fit within the bounds of the picture (see figure 6.2.3.b).

(4) After determining the desired change in magnification, scale the tolerance region of the task to the desired size, and overlay it on the right side (see figure 6.2.3.c).

(5) Then the user can adjust the camera, take a picture, display it, and see if the camera is correctly positioned (see figure 6.2.3.d). The camera is correctly positioned if the region outlined in the left picture is blown up at least as large as the region outlined on the right and the region outlined on the left is completely contained in the picture on the right.

This technique simplifies the earlier procedure because the user can easily see the effect of changing the position and focal length of the camera. He can position the camera by adjusting the portion of the picture outlined on the left until it fills up the outline on the right.

This technique is interactive, not automatic. Currently there is no substitute for experience when it comes to deciding which camera positions are reasonable and which are not. Eventually high-level strategy programs will be able to suggest reasonable camera locations from models of the task. To do that the strategy program will have to consider several factors:

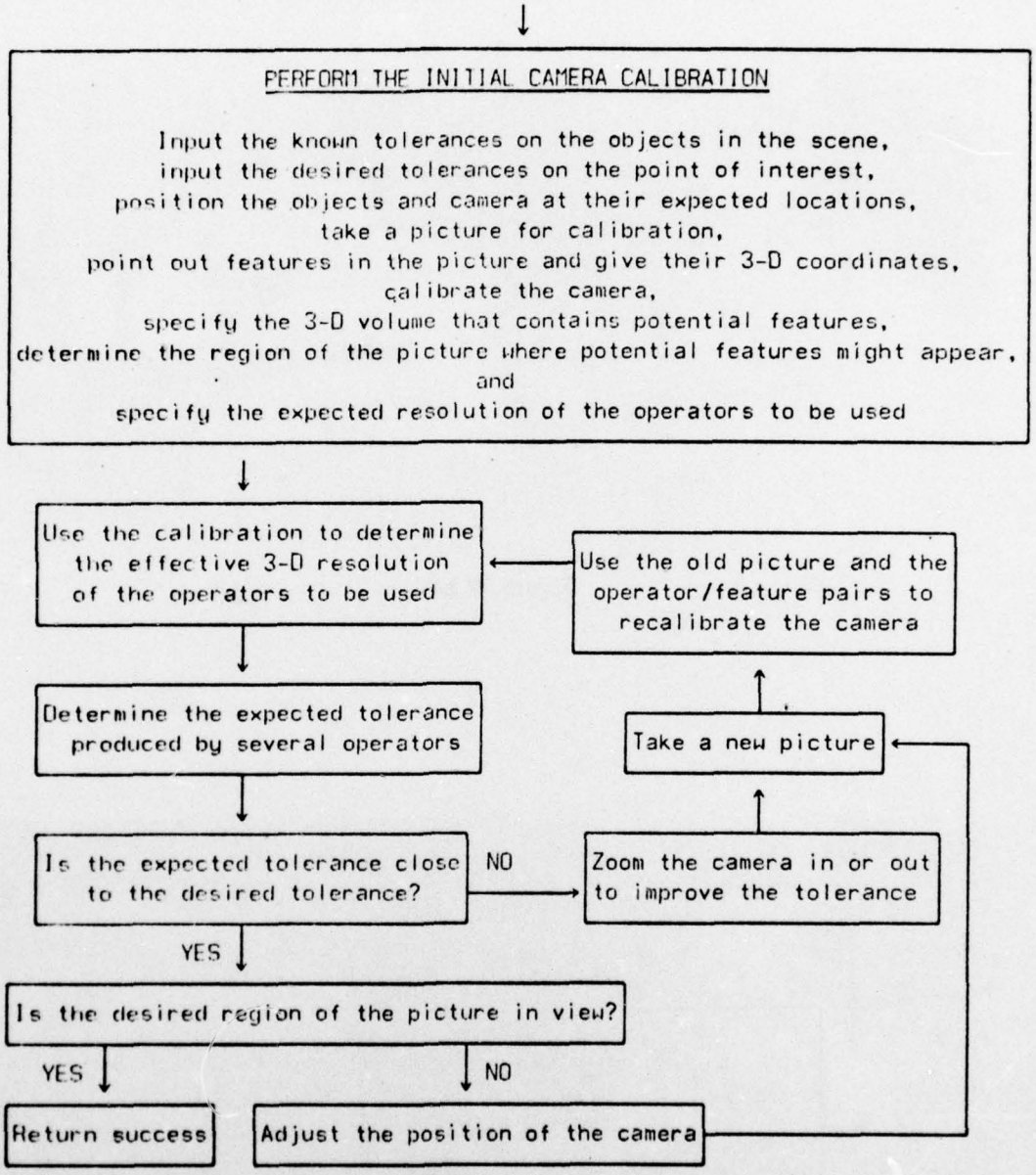

PERFORM THE INITIAL CAMERA CALIBRATION
Input the known tolerances on the objects in the scene,
input the desired tolerances on the point of interest,
position the objects and camera at their expected locations,
take a picture for calibration,
point out features in the picture and give their 3-D coordinates,
calibrate the camera,
specify the 3-D volume that contains potential features,
determine the region of the picture where potential features might appear,
and
specify the expected resolution of the operators to be used
Use the calibration to determine the effective 3-D resolution of the operators to be used
Use the old picture and the operator/feature pairs to recalibrate the camera
Determine the expected tolerance produced by several operators
Take a new picture
Is the expected tolerance close to the desired tolerance?
NO
Zoom the camera in or out to improve the tolerance
YES
Is the desired region of the picture in view?
YES
NO
Return success
Adjust the position of the camera

Figure 6.2.2

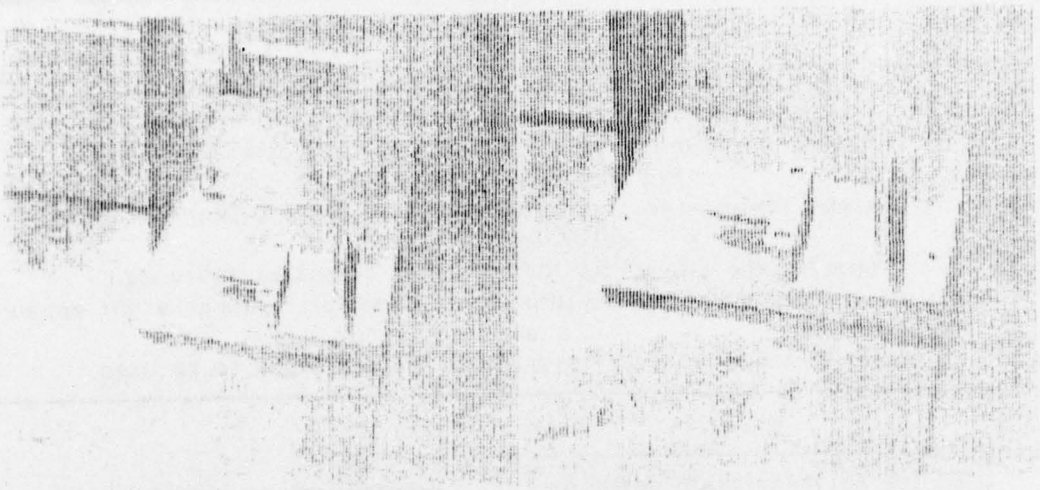


Figure 6.2.3.a

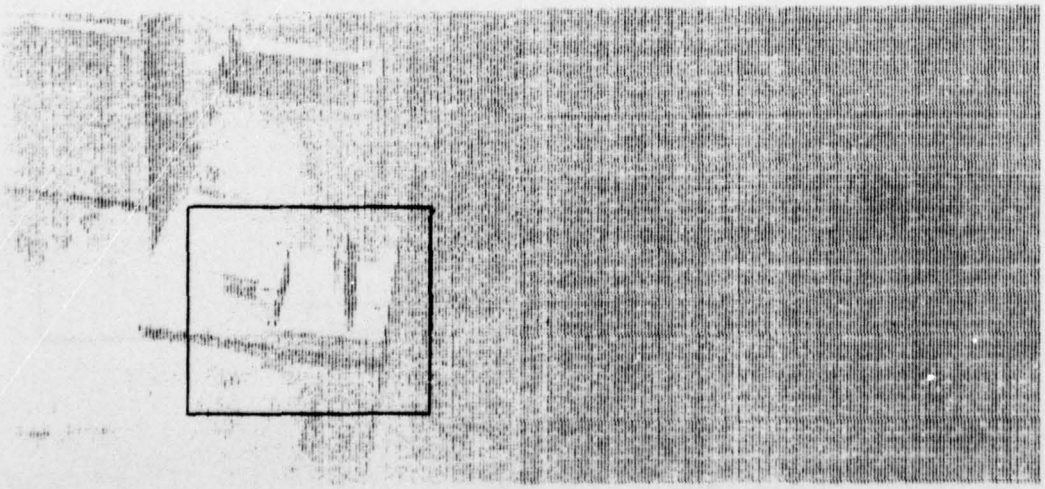


Figure 6.2.3.b

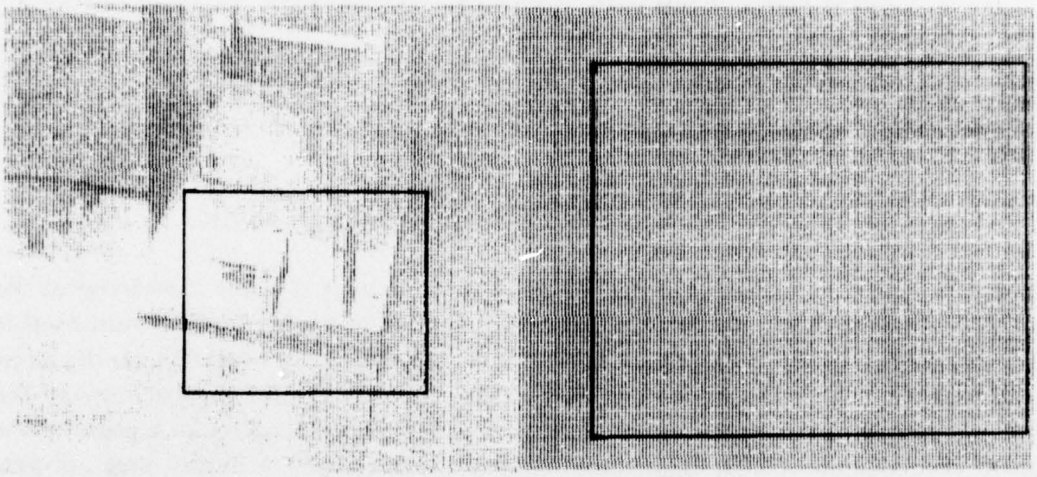


Figure 6.2.3.c

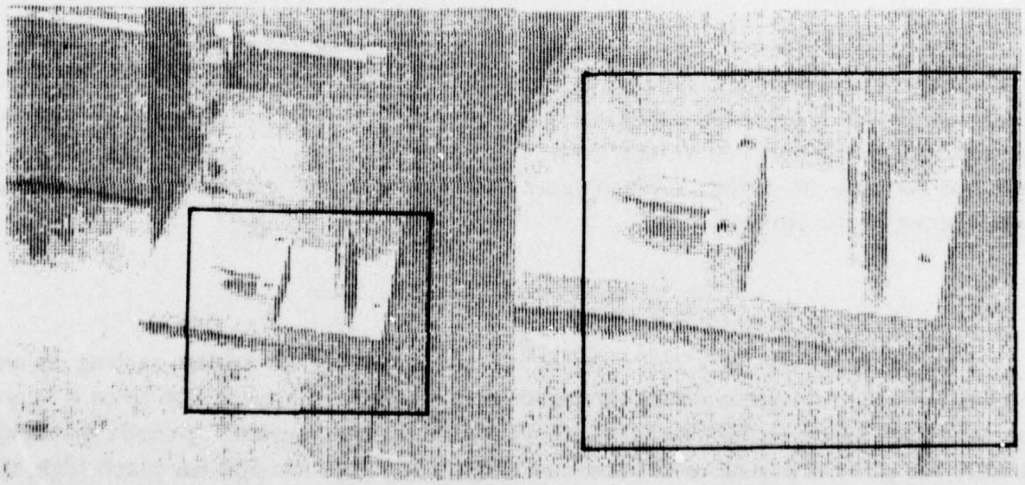


Figure 6.2.3.d

(a) the tolerance volume of the task,

(b) the resolution of the picture,

(c) the other objects in the scene,

(d) the path of the arm,

(e) the lighting,

and (f) the other vision tasks for which the camera might be used.

For example, if VV is to be used to check for a screw on the end of the screwdriver as the screwdriver leaves the screw dispenser, the camera must be in a safe place and must be able to see the screw. To be in a safe place it must be out of the way of the arm. To see the screw it must be above the table and not behind the dispenser. In general, the placement of the camera is similar to the object avoidance problem. The program has to locate a place for the camera such that the line of sight from the camera to the object of interest does not pass through any other objects in the scene.

If the camera is not movable, it has to be out of the way of the arm for all of the motions in the task, not just the one or two proceeding the vision subtask. For example, if a camera is to be used to locate a hole, it has to be out of the working area of the arm when the arm places the part in the vise, aligns the top, inserts the screws, and finally transfers the part to its packing box.

The strategy program should also consider the complete sequence of events when it tries to select a position for the camera. If a camera can be used for more than one task, try to position it so that it meets the requirements for all of the tasks. For example, if a camera can be used to locate a hole and visually servo a screw into the hole, try to position it so that it does not have to be moved between tasks. In general, try to minimize the amount of repositioning of the cameras.

2. Set up for an Inspection Task

For an inspection task the resolution of the picture is not critical as long as the operators produce sufficiently different results when the object is present than when it is not. For example, in the screw-checking task if one correlation operator can reliably decide whether the screw is present or not when the image of the screw is only a few pixels high, the resolution of the picture can be quite low (see figure 6.2.4.a). However, if one operator is not reliable enough or distinguishing enough, the resolution has to be increased so that several

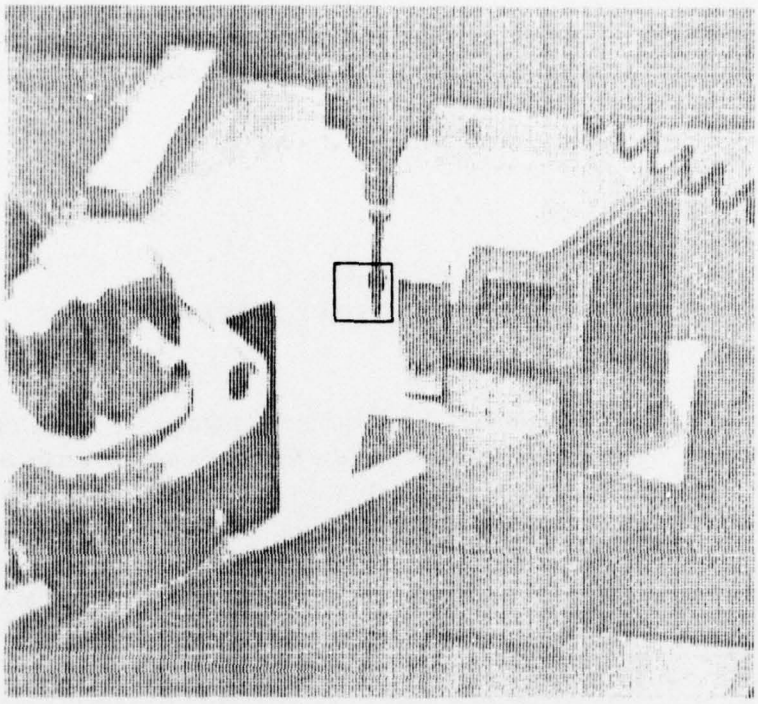


Figure 6.2.4.a

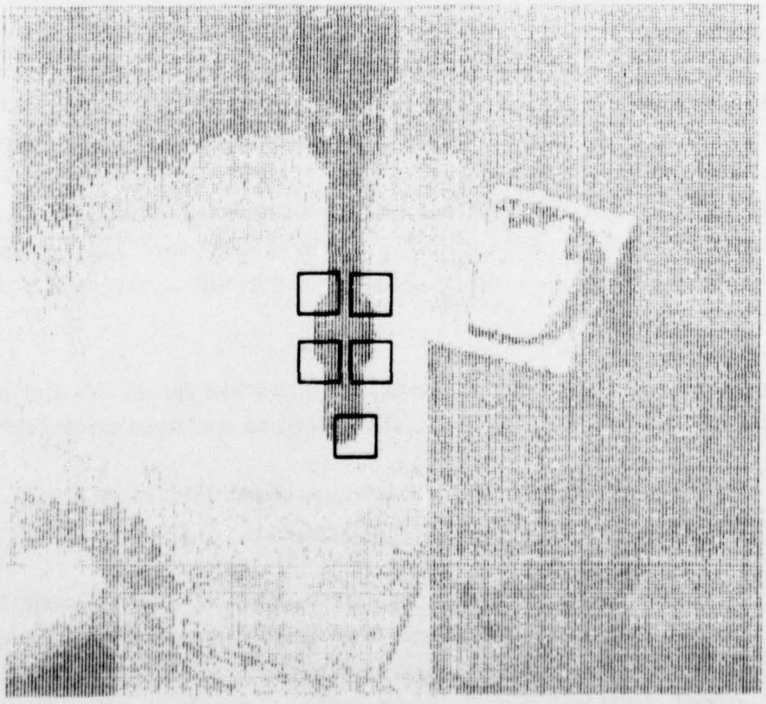


Figure 6.2.4.b

features are visible. Figure 6.2.4.b shows the screw enlarged so that the corners can be distinguished. An even higher resolution would be required to detect the texture of the screw threads.

Section 3
CHOOSE POTENTIAL OPERATOR/FEATURE PAIRS

How does one choose potentially useful operator/feature pairs? Often the appearances of several features are quite similar. How can one find operators that locate unique features or operators that only have a few known alternatives? As mentioned in chapter 2, a VV system can provide three levels of assistance to help a user answer these questions: (1) a convenient environment in which to experiment with different operators, (2) a list of suggestions derived from typical pictures of the task, or (3) a list of suggestions derived from a model of the task. These three levels of assistance are discussed below.

1. A Convenient Environment

One of the basic features of the current VV system is a user interface that facilitates experimentation. Such an interface is important because the current analytic tools are not complete enough to predict the result of applying an operator to a picture. The VV system can not automatically investigate all aspects of an operator. A user has to help the program decide which operator/feature pairs have the most potential. To help the system, the user has to experiment with different operators and develop an intuition about them. Thus, empirical techniques are used to bridge the gaps that remain in analytic models. An operator is applied to several training pictures in order to predict how it will behave when applied to an unknown picture.

Figure 6.3.1.1 is a flowchart for one type of interactive system. A user of that system might produce the following protocol as he selects potential operator/feature pairs:

(1) Position the screw dispenser at its expected location and take a picture to be used as the planning picture.

(2) Describe a corner that has an internal angle of approximately 90 degrees and a contrast across the edges of approximately 12 grey levels. (The digitization of an analog picture converts the intensity at each pixel into one of a finite number of grey levels. Pictures are

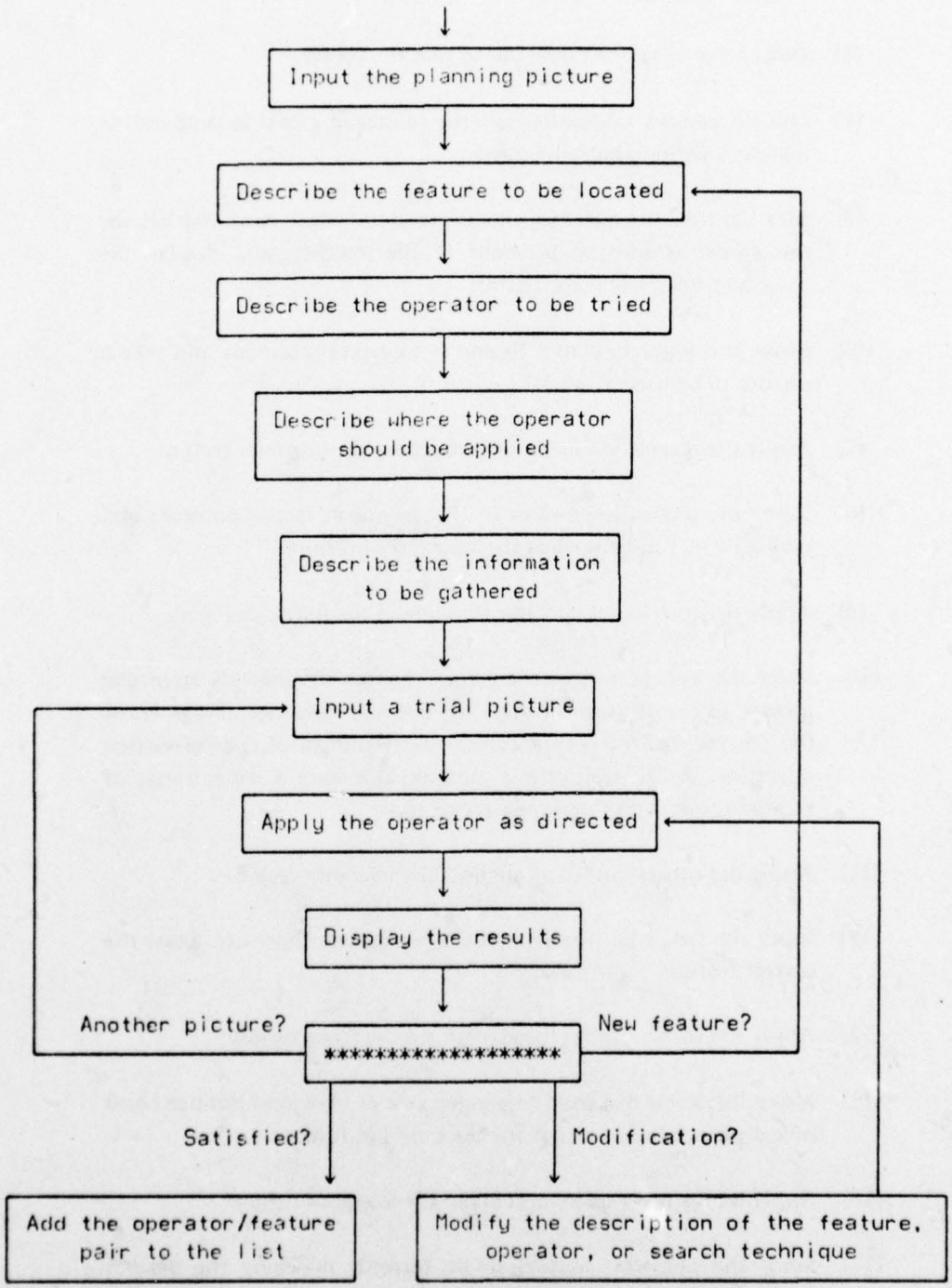

Input the planning picture
Describe the feature to be located
Describe the operator to be tried
Describe where the operator should be applied
Describe the information to be gathered
Input a trial picture
Apply the operator as directed
Display the results
Another picture?
New feature?
****
Satisfied?
Modification?
Add the operator/feature pair to the list
Modify the description of the feature, operator, or search technique

Figure 6.3.1.1

usually digitized to 16, 32, 64, or 256 grey levels.).

(3) Describe a correlation operator to find the corner.

(4) Use the camera calibration and the constraint model to produce the tolerance region about the corner.

(5) Indicate that the operator should locate the three best matches for the corner, show the positions of the matches, and display the associated correlation coefficients.

(6) Move the screw dispenser to one of its typical positions and take a picture to be used as a trial picture.

(7) Apply the correlation operator throughout the tolerance region.

(8) Move the screw dispenser to another one of its typical positions and take a picture to be used as the second trial picture.

(9) Apply the correlation operator throughout the tolerance region.

(10) Since the appearance of the corner changes significantly from one picture to the next, the correlation operator does not always locate the correct corner. Try a corner-finder instead of the correlation operator. A corner-finder is more reliable over a wider range of rotations, but it is more expensive to apply.

(11) Apply the corner-finder throughout the tolerance region.

(12) Input the first trial picture to see if the corner-finder can locate the correct feature.

(13) Apply the corner-finder throughout the tolerance region.

(14) Move the screw dispenser to another one of its typical positions and take a picture to be used as the third trial picture.

(15) Apply the corner-finder throughout the tolerance region.

(16) Since the operator appears to be correctly matching the desired corner, add the corner-finder and corner to the list of potential operator/feature pairs. The pair may be discarded later during the

planning stage, but so far it looks like a potential source of information.

This protocol demonstrates the overall process involved in choosing potential operator/feature pairs. Traces of this process within the current version of the VV system can be found in appendices IV and V.

Several of the terms and operations used in the above protocol need to be explained further. The remainder of this section describes these concepts in more detail.

Planning picture

The planning picture is a picture of the objects in their expected locations. It is the reference picture in the sense that it is used to define the planning locations for all of the features. For example, the planning location for the 90-degree corner on the screw dispenser can be defined by pointing out its position in the planning picture. Since the model of the screw dispenser contains the height of the 90-degree corner, the camera calibration can be used to compute the corresponding point in the workstation coordinate system. At execution time when the corner-finder determines the matching position for the corner in the picture, the same calibration and height can be used to compute the current location for the corner.

Since relative location changes can be more accurately computed than absolute locations, the execution-time program directly computes the relative change from the position of a feature in a planning picture, its position in the current picture, and its planning location in the workstation. This relative change can be used to adjust the destination of the arm.

Describing a feature

There are two basic mechanisms that are necessary to describe features: (1) a cursor with which to point out features and (2) a model within which to store the description of the features. For example, consider the 90-degree corner that is part of the screw dispenser shown in figure 6.3.1.2. In order to describe that corner the user might do the following:

(a) Declare that he is about to describe a corner.

(b) Use the cursor to point out the vertex of the corner.

(c) State the height of the vertex.

(d) Indicate a point on one of the edges that forms the corner.

(e) Indicate a point on the second edge that forms the corner.

This sequence of actions describes the location of the feature in the planning picture and in the workstation coordinate system. It does *not* describe the appearance of the feature. The appearance is described in conjunction with the description of the operator used to find the feature. A different description of the appearance is made if a correlation operator is to be used than if a corner-finder is to be used.

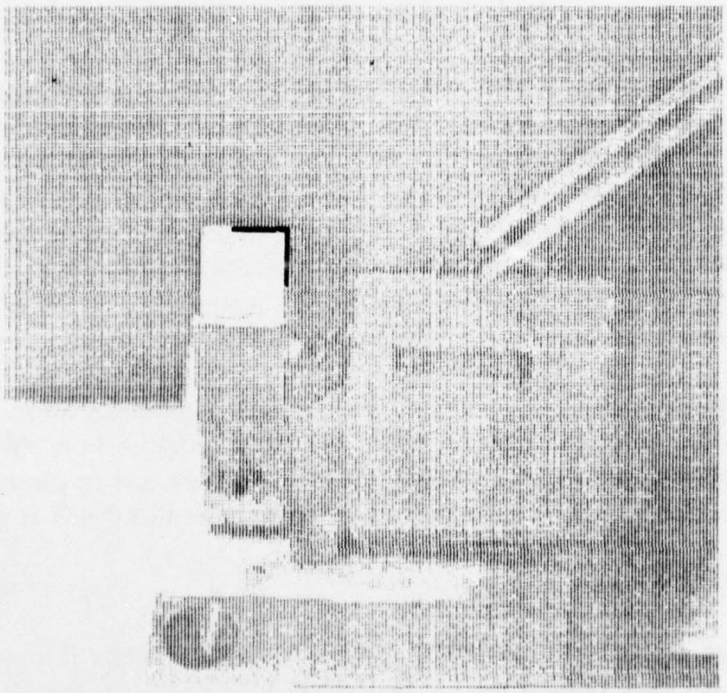


Figure 6.3.1.2

Different types of features require different types of descriptors. For example, a texture has to be described in terms of frequencies, sizes, and shapes; a region has to be described in terms of boundary segments or areas; and an edge has to be described in terms of length, orientation, and distinctness. As new features are added to the VV system, the appropriate descriptors have to be added to the interactive system.

Describing an operator

Operators are generally parameterized; they have different sizes, different thresholds, and different versions. The user has to set the values for these parameters in order to begin

experimenting with an operator. For example, in order to apply a corner-finder, such as Aharon Gill's [Gill 72], the user may have to specify an estimate for the contrast across the edges that form the corner. Often estimates for this type of quantity can be obtained by applying a less complicated operator, such as an intensity extraction operator, to one of the training pictures.

In general, the parameters for an operator go through three stages as the VV program is constructed:

(1) they are initially estimated, possibly with the help of other operators,

(2) they are changed several times as the user experiments with different variations,

and (3) they are finally fixed after the training session.

As operators are better understood and modelling techniques improve, this process of successive approximation will be shortened.

Describing where an operator should be applied

Given a feature and an operator to locate the feature, the calibration can be used to determine the tolerance region about the feature, but where should the operator be applied within the tolerance region? That is, what search strategy should be used to locate the feature? As mentioned in chapter 2 the strategy depends upon several factors: the type of feature, the size of the operator, the size of the tolerance region, and the distribution of occurrences within the tolerance region. Each type of feature needs its own method to determine the best search technique for each situation.

Figure 6.3.1.3 shows an example of a search strategy to be used to find a point of a line segment. The diagonal line is the expected location of line segment in question. The rectangle is the tolerance region about the center of the segment. The numbered dots indicate the position and order of the edge operator applications. The first part of the search concentrates on the region close to the planned position of the center of the segment because the distribution of occurrences indicates that the probability of finding a match is highest there. The pattern guarantees at least one point on the line segment.

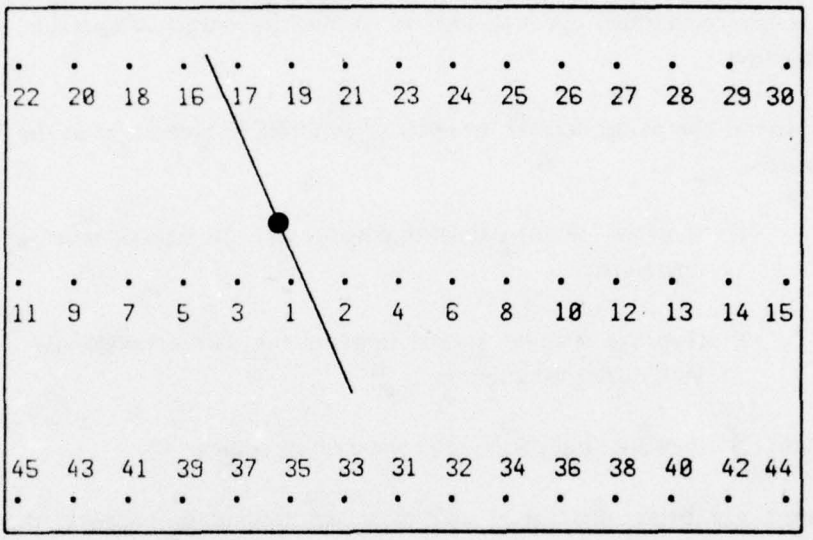


Figure 6.3.1.3

Checking for potential decoys

The reliability and expected contribution of an operator depend upon the number of potential confusions for the operator. For example, if there are three corners that all appear similar to the 90-degree corner, the corner-finder may sometimes locate one them and think that it has found the expected 90-degree corner.

To check for decoys the user can ask the operator to return the best three or four matches for a feature. If the values of the operator indicate that it is difficult to distinguish between the matches, the operator may be discarded. If the matches are similar, but the operator seems to be able to distinguish the desired match from the others, the decoys can be added to the model of the feature as *known alternatives*. The training session will gather statistics on the known alternatives and decide whether or not they can be reliably distinguished from the desired match.

2. *Suggestions from Typical Pictures*

Given a convenient environment for experimentation, the user still has to generate his own candidates for potential operator/feature pairs. This section describes an automatic

method to generate such candidates. The idea is to apply an operator to a training picture and locate visually distinct features. For example, a procedure to suggest line segments might be: apply an edge operator to a picture, link up sets of edge segments that form lines, and suggest as candidates those lines that have a certain minimum length, contrast, and uniqueness. Given a list of suggestions, the user can discard any of them because they are not on the objects of interest or because they can not be reliably found.

This type of automatic suggestion system is most important when it is difficult for the user to be part of the feature selection process. For example, if a vehicle is navigating across the martian surface, it would be difficult for a user to point out features of interest. It would be possible, of course, but it would be better if the vision system could make its own suggestions. In programmable assembly the user is currently part of the programming process, but an automatic suggestion system is still useful if it makes reasonable suggestions. The better the suggestions, the less work the user has to perform. If a user asks for suggestions and they are all unreasonable, he can easily revert back to pointing out features himself.

Line segment suggestions

The following list of steps is a more detailed description of a line segment suggestion procedure:

(1) Automatically set thresholds for the edge operator.

(2) Automatically apply the operator to the whole picture.

(3) Automatically group the resulting edges according to position, angle, and contrast.

(4) Automatically fit lines through the groups.

(5) Automatically analyze the resulting line segments and suggest those lines that are a certain length.

The fact that the suggestions are produced automatically is important because it reduces the user's work to the subtask of selecting good line segments from the list of suggestions.

The technique referred to in the first step that automatically sets the thresholds for the edge operator was developed by Binford [Binford 75]. The existence of this technique is important because it avoids the educated guessing that has characterized the use of edge operators in the past. Binford's technique makes it possible for a consistent method to be used to set the thresholds at programming time, training time, and execution time.

The fact that the same operator is applied at programming time to make the suggestion and at execution time to find the segment insures that the suggestions are realistic. The suggestions are not hypothetical; they are features that have been found by a standard visual operator. This concreteness avoids features that are not distinct, but it also introduces suggestions that are based upon strictly visual phenomena, such as shadows and glares.

The idea of linking edge points together, mentioned in steps three and four, is not new; line followers have been used within blocks world vision for some time. Recently, however, there has been renewed interest in this problem (see [Marr 75b], [Nevatia 75], and [Binford 76b]). The new approaches apply edge operators to a picture and then collect the edge segments that form larger structures, such as lines or curves.

The line segment suggestion system can be extended to suggest more complex features such as corners and curves, but the heuristics required to decide which points are related are much more complex. A partial step in that direction is to let the user point out two or three points on a curve and have the system automatically apply the edge operator to follow and characterize the curve that appears in the picture. A limited system of this type has been implemented for the programmable assembly environment (see appendix VI). Smooth curves are particularly important within programmable assembly because the parts are often formed by boring and milling operations, which produce cylindrical holes and other surfaces that appear as smooth curves in a picture.

Correlation suggestions

Some operators can not be directly used to locate suggestions like the edge operator. They require more information than a few parameter values before they can be applied. A correlation operator is a prime example. It is defined in terms of an array of intensity values, which are normally taken directly from a planning picture. For such operators a companion operator, called an *interest operator*, can be defined that locates portions of the picture that are likely candidates for that type of operator. Once the interest operator has located a likely portion of the picture, the operator can be defined. In the case of a correlation operator, the intensity values in the picture are used to define the correlation operator.

Quam and Moravec have both designed and implemented interest operators that locate promising positions for correlation operators [Quam 74] and [Moravec 76]. The basic idea is that the most effective correlation operators are based upon portions of a picture that contain high variance information along two different directions. For example, consider the ninty-degree angle shown in figure 6.3.2.1.a. It has high variance information in the vertical and horizontal directions because of the sharp changes in intensities in those directions. When that correlation operator tries to find a match in another picture, such as figure 6.3.2.1.b, which contains a similar corner, the two-dimensional information will clearly distinguish the best match. The correlation operator shown in figure 6.3.2.1.c, on the other

hand, only has high variance information along one direction. When it is applied to another picture, such as figure 6.3.2.1.d, it will not be able to locate a unique match because all of the points along the line match equally well. Thus, good correlation operators should be based upon regions that have high variance information in two directions. The closer the directions are to being orthogonal, the better the operator.

Gennery has developed a correlation operator that directly measures the precision of each match, which depends upon the two-dimensional information contained in the operator and the trial picture [Moravec 76]. His operator returns an error ellipse with each match. The ellipse indicates the two-dimensional precision associated with the match. For an operator like the one shown in figure 6.3.2.1.c the error ellipses are generally elongated because the precision is poor along the edge, but good perpendicular to the edge.

Both Quam's and Moravec's interest operators check local areas of a picture for high variance information in two directions. Moravec's operator accumulates the sum of the squares of the differences along four directions: vertical, horizontal, and the two diagonals. If the local area has no directional information or only one-directional information, one of the four sums will be close to zero. If the local area has two-dimensional information, all of the sums will be greater than zero. The measure of interest for a local region is the minimum of the four sums. In this way if the area has two-dimensional information the measure will be significantly greater than zero. Otherwise, it will be close to zero. To distinguish *interesting* areas from *uninteresting* areas the program compares the interest values of the areas with a threshold. The program sorts the potential suggestions according to their interest value and presents them to the user as an ordered list of suggestions.

Figure 6.3.2.2 shows four typical planning pictures to which Moravec's interest operator has been applied. Figure 6.3.2.3 shows the same four pictures and the first fifteen suggestions made by interest operator. Most of the suggestions are reasonable in the sense that the areas contain two-dimensional information. However, some of the suggestions are on objects that not of interest. For example, the twelfth suggestion in figure 6.3.2.3.a is on the screw dispenser, which remains the same whether the screw is present or not. Hence, its values will not contribute much to the final decision. If an operator of this type happens to be given to the training stage, the system will accumulate values of the operator and form two distributions, one for the case with the screw present and one for the case with the screw missing. The two distributions will be essentially the same. Since they are essentially the same, the planning stage will give the operator/feature pair a poor rating and the operator will not be used.

A few of the operators shown in figure 6.3.2.3 are based upon visual features that change as the objects in the scene change locations. For example, the twelfth operator in figure 6.3.2.3.b is centered upon an angle formed by the side of the pencil sharpener and part of the fixture. Since the location of the fixture is initially only known within plus or minus ten degrees, it may rotate ten degrees during an actual assembly. Since there is a large depth

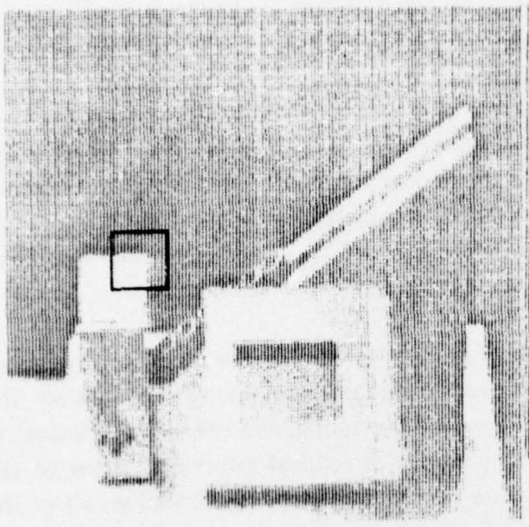


Figure 6.3.2.1.a

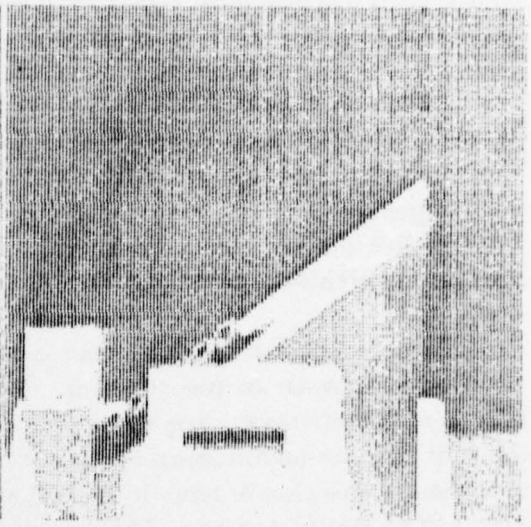


Figure 6.3.2.1.b

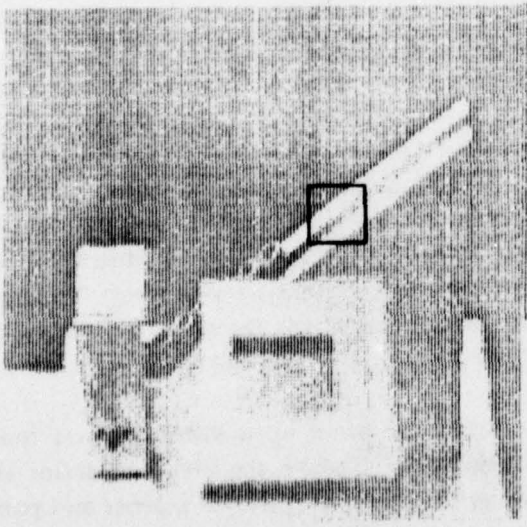


Figure 6.3.2.1.c

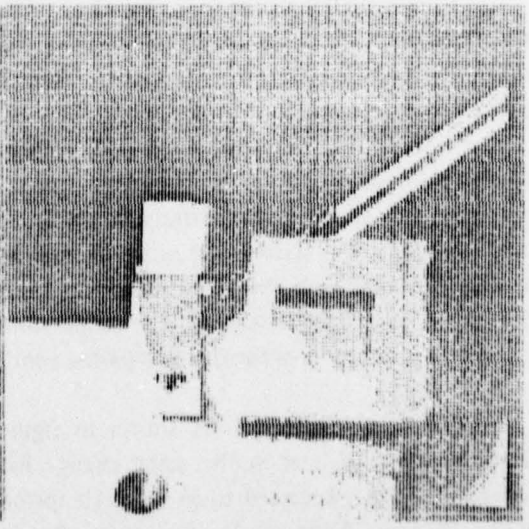


Figure 6.3.2.1.d

discontinuity between the side of the pencil sharpener and the fixture, the rotation will cause the visual corner to slide along the side. Thus, even if the correlation operator locates the same visual corner, it will not be locating the same point on the pencil sharpener.

Operator number seven in figure 6.3.2.3.b is another example of an operator that may not locate the same point on the pencil sharpener even though it finds a good match. In this case the operator is based upon one of the teeth in the main gear. If the subassembly rotates ten degrees away from its planning position, the operator will find a matching tooth, but it will be the wrong one. The user or the current system should discard suggestions of this type because their position information is not reliable. A more powerful system could use the fact that all of the teeth form a circle. If a VV program finds one tooth, it has located one point on that circle.

The training system can automatically discard features that produce unreliable position information if it uses the least-squares fitting routine to check the structural consistency of a set of matches:

(a) After all of the matches have been made, use the least-squares routine to determine the best fit for all of the matches.

(b) Use the best fit to determine the residual error for each feature.

(c) Gather statistics on the residual errors of the features.

and (d) Discard features that consistently have large residual errors.

This procedure works (and is demonstrated in appendix V), but it would be more esthetically pleasing if the program could analyze a model of the scene and determine the character of each visual feature. The next subsection will discuss this type of analysis.

One of the problems with Moravec's current interest operator is that some of its suggestions are not centered about the portion of the picture that contains the two-dimensional information. For example, operator number six in figure 6.3.2.3.a is not centered on the corner that provides the two-dimensional information. This problem arises because the interest operator is actually applied to a reduced version of the picture instead of the picture itself. This approach is nice because it avoids some of the high frequency noise that otherwise might confuse the interest operator, but it also makes it difficult to locate suggestions accurately. This slight problem will soon be solved by applying the interest operator to the reduced picture and then refining the position of the suggestion by applying the operator to the original picture.

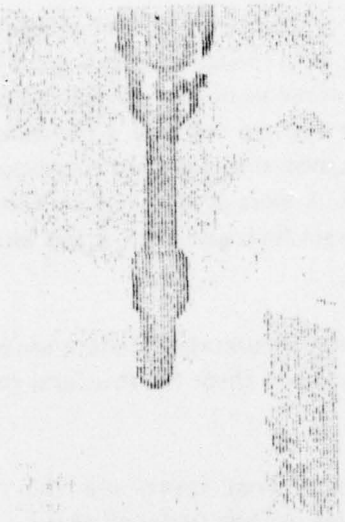


Figure 6.3.2.2.a

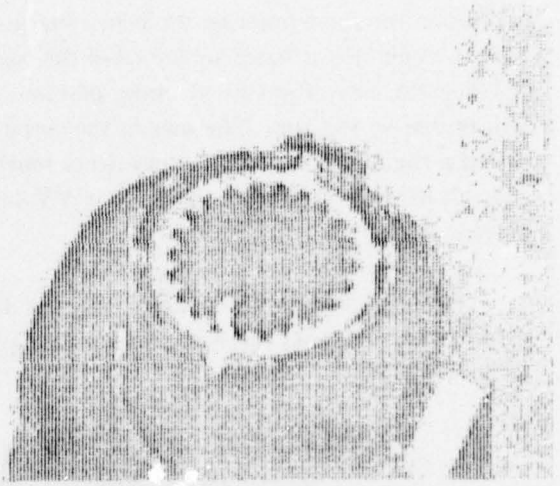


Figure 6.3.2.2.b

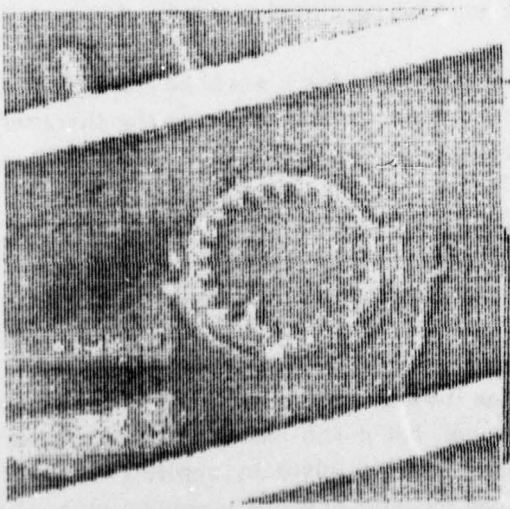


Figure 6.3.2.2.c

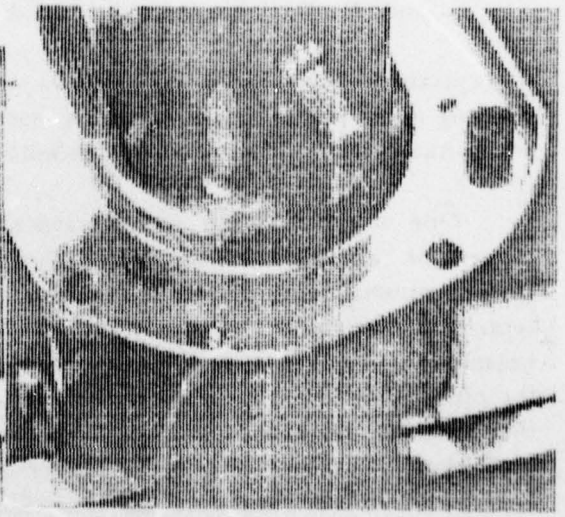


Figure 6.3.2.2.d

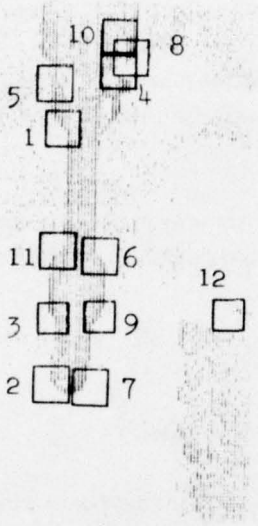


Figure 6.3.2.3.a

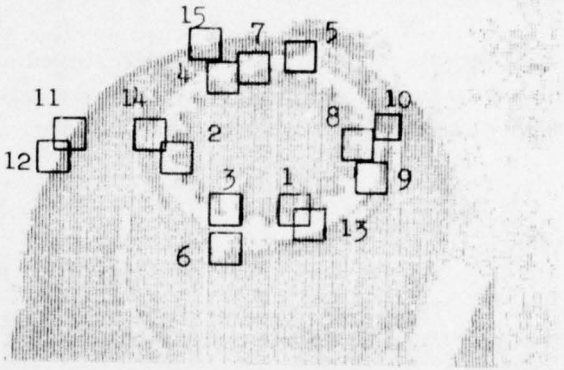


Figure 6.3.2.3.b

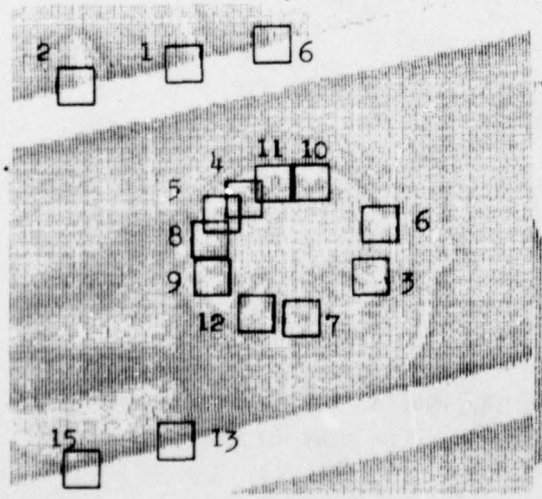


Figure 6.3.2.3.c

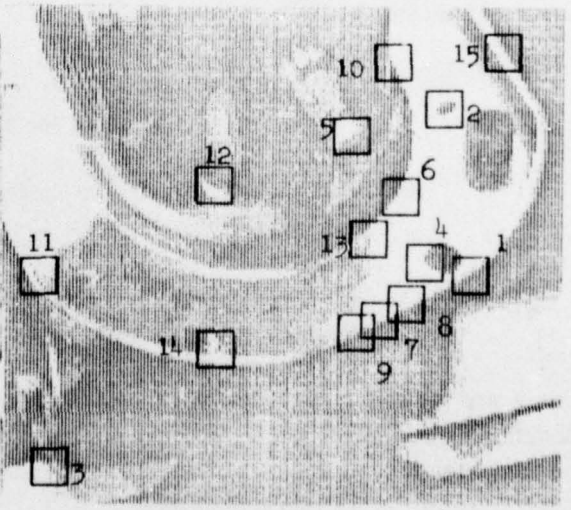


Figure 6.3.2.3.d

The interest operator also suggests a few regions that do not have two-dimensional information, e.g., see operator number fifteen in figure 6.3.2.3.c and operator six in figure 6.3.2.3.d. Since the variances are only computed at forty-five-degree intervals, edges at approximately twenty-two degrees appear to have some variance along all four directions. If these variances are large enough, the area may be erroneously suggested as a good place for a correlation operator. A partial solution to this problem is to increase the number of directions in which the variance information is computed.

In programmable assembly, since suggestions are only made once for each task, one can tolerate relatively slow methods that suggest potential operator/feature pairs. The better the suggestions, the less work required of the user.

3. Suggestions from a Model

A model-based suggestion procedure uses a three-dimensional model of the scene to produce a synthetic picture of the scene and then analyzes the synthetic picture to suggest potential operator/feature pairs. The advantage of such a procedure is that it can associate visual features with the parts of the objects that produce them. Given this correspondence, the system can potentially use all of the properties of the objects to decide which visual features are the best candidates for a VV task.

In this discussion a *synthetic* picture is a picture that is generated from an analytic model of the scene. A *real* picture is a picture taken by a camera. An analytic model of a scene may include:

(1) object models and camera models that determine the positions of the features of the objects in the picture,

(2) light models and surface models that determine the grey level for each pixel,

and (3) attachment models and constraint models that determine the tolerance region for each feature.

There are several levels of complexity for each of these models. For example, the surface models may be planar patches or splined quadratic patches. The better the model, the better the suggestions.

In programmable assembly, since many of the parts are manufactured from mechanical

drawings, it is relatively easy to produce the basic object models. Hopefully as object modelling is improved, a three-dimensional description language will be developed that can be used to control all of the processing associated with a part: manufacturing, inspection, and assembly. Such a language would help standardize parts and unify the process of assembly.

A first step toward an automatic suggestion system has been implemented. It is used to suggest promising curve segments. It is based upon Miyamoto's modelling and graphics system [Miyamoto 75], which in turn is based upon Binford's spine-cross-section model (see [Agin 72], [Nevatia 73], and [Miyamoto 75]). Each part within a spine-cross-section model consists of a spine and a sequence of smoothly varying cross-sections along that spine. For example, a shaft can be described as a straight line that has circular cross-sections (see figure 6.3.3.1.a). A rectangular box can be described as a straight line that has rectangular cross-sections (see figure 6.3.3.1.b). Smooth curves are approximated by a sequence of line segments. The user specifies the curve and the number of segments to use to approximate the curve and the system produces the correct sequence of line segments. This representation has the nice property that it distinguishes between lines that are actually part of the object and lines that are used to approximate a curve. Since this distinction is made, the system can be smart about which lines it displays. For example, the hidden-line view of the shaft shown in figure 6.3.3.1.a does not show all of the visible approximating lines, only those that outline the part (see figure 6.3.3.1.c). Figure 6.3.3.1.d shows the standard hidden-line view that includes all of the visible lines.

The steps in the curve segment suggestion procedure are:

(1) Interactively construct models for the objects in the scene and place them at the correct relative locations with respect to the camera.

(2) Use a hidden-line scheme to delete lines that are not visible and produce a two-dimensional line drawing of the scene.

(3) Interactively point out promising curve segments that appear in the line drawing.

(4) Automatically fit a smooth curve through the broken-line segments that approximate the curve segment.

(5) Interactively estimate the contrast across the curve.

(6) Automatically use the smooth curve to locate and characterize a similar curve in a real picture of the scene, if one exists.

(7) If the characterization of the curve indicates that it has sufficient

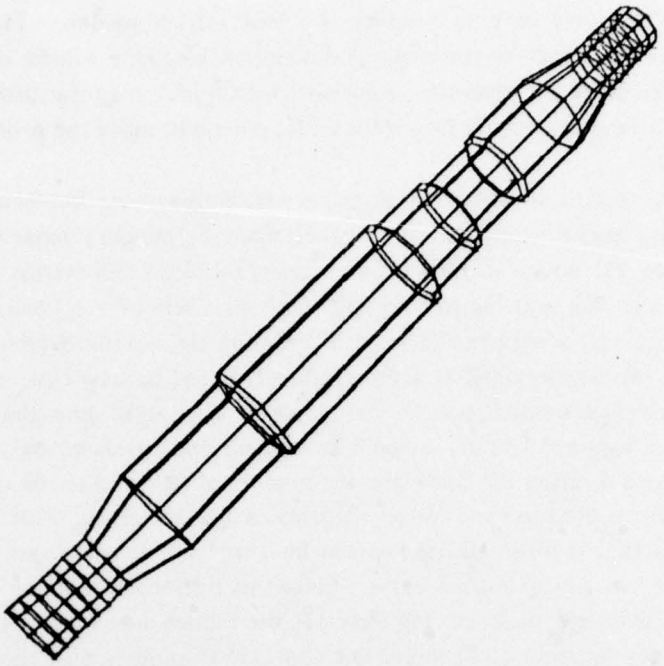


Figure 6.3.3.1.a

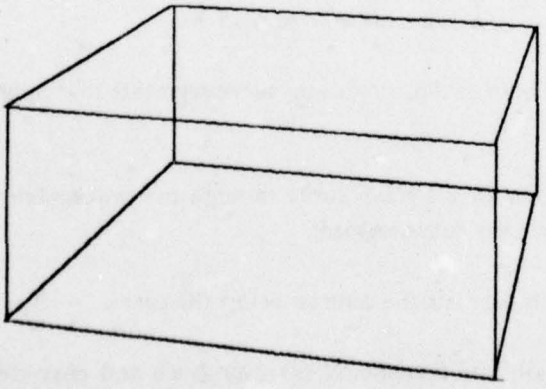


Figure 6.3.3.1.b

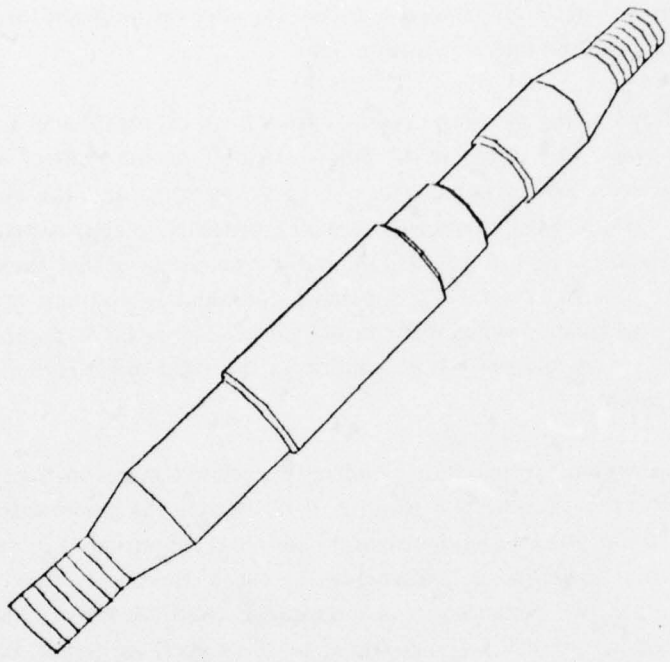


Figure 6.3.3.1.c

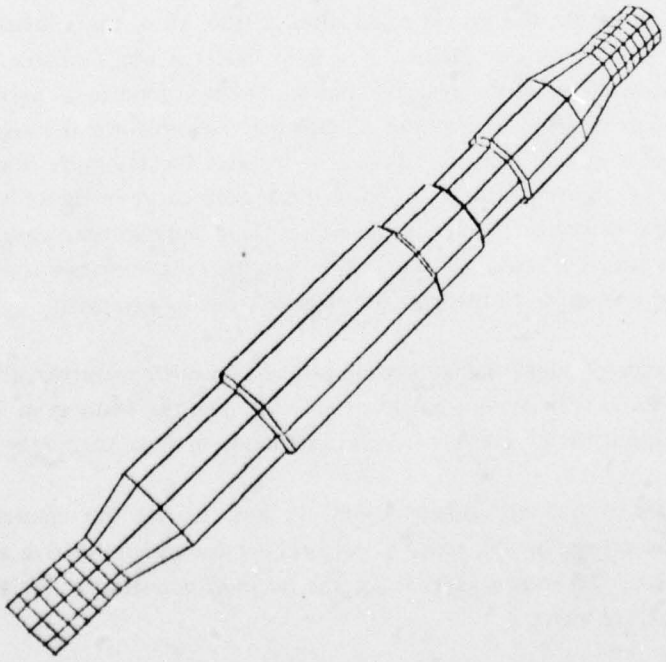


Figure 6.3.3.1.d

contrast, length, and smoothness, add the edge operator and curve to the list of potential operator/feature pairs.

Figure 6.3.3.2 shows a line drawing produced from a typical model and a real picture of the corresponding scene. The object is the points assembly on the shaft of a chainsaw engine. The user interactively points out one of the curve segments and the system automatically characterizes it. Figure 6.3.3.3.a shows one of the broken-line approximations overlayed on top of the real picture. Figure 6.3.3.3.b shows the smooth curve that the system fit through the broken-line segments. Figure 6.3.3.3.c shows the initial applications of the edge operator that are designed to locate points on the actual curve. Figure 6.3.3.3.d shows the final curve produced by the curve follower. It extends from the right-angle corner at the top to the shadow at the bottom.

Since the program can maintain pointers that relate features in the synthetic picture to the object models that produce the features, it is theoretically possible for the program to avoid several of the problems mentioned in the previous subsection. For example, the system could determine that the right-angle corner at the end of the smooth curve is formed by two objects that have a depth discontinuity between them: the shaft is in front of the side of the points assembly casing. Such a corner would not be a good suggestion because its position and appearance will change as the locations of the objects in the scene change.

The system could also decide whether or not a visual feature is formed by a shadow. If a light model is available, the system could directly label all of the synthetic visual features that are formed by shadows or glares. If a light model is not available, the system could check the line drawing for object features that might correspond to a specific visual feature. If there are no object features that might explain the visual feature, the visual feature would be classified as a transient feature. Transient features include such things as dirt spots, shadows, and glares. For example, one end of the smooth curve in figure 6.3.3.3.d is lost is a shadow. When the characterization routine notices the change in contrast and distinctness as it is following the curve, it could ask the modelling system if the change is due to a feature of the object or some transient. In this case the change is due to a transient.

This same type of suggestion procedure can suggest other features, such as corners, line segments, and regions. The system has to be able to locate the features in a line drawing of the scene and decide if the grey levels are distinct enough to make them easy to find.

The ultimate suggestion system would be able to use the constraint information associated with the objects in the scene to perform a more comprehensive analysis. It could determine the number of known alternatives, the range of appearances for a feature, and the existence of degenerate views.

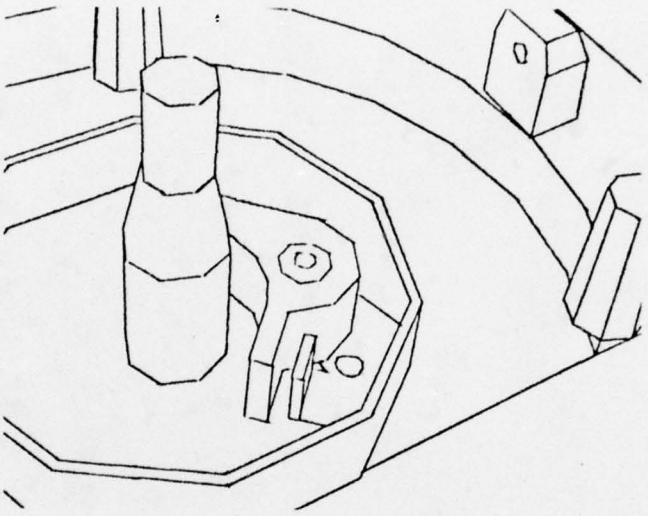


Figure 6.3.3.2.a

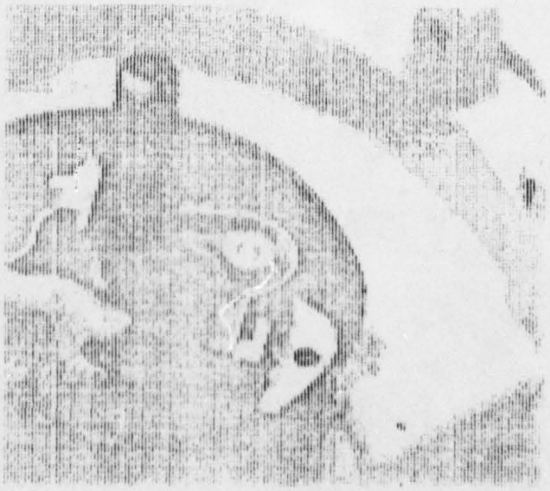


Figure 6.3.3.2.b

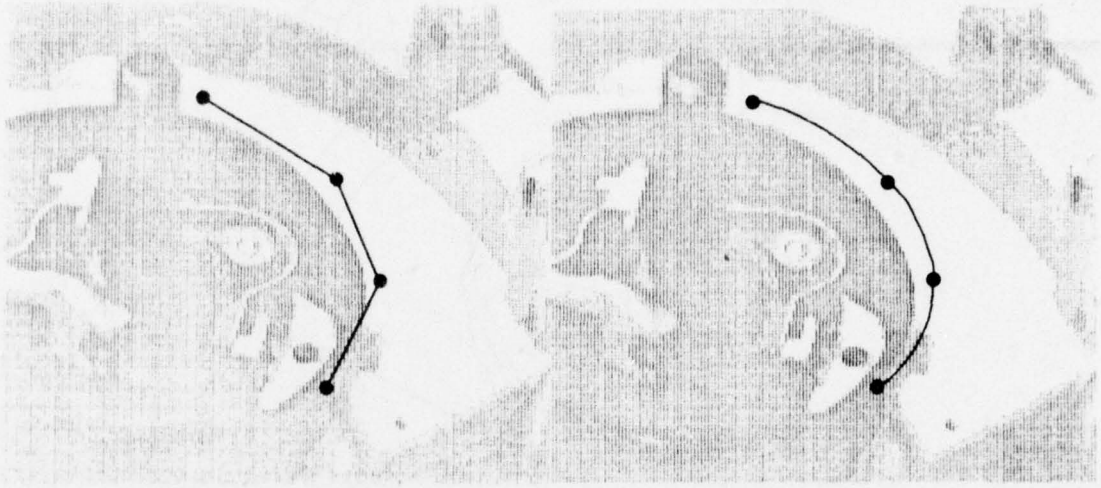


Figure 6.3.3.3.a

Figure 6.3.3.3.b

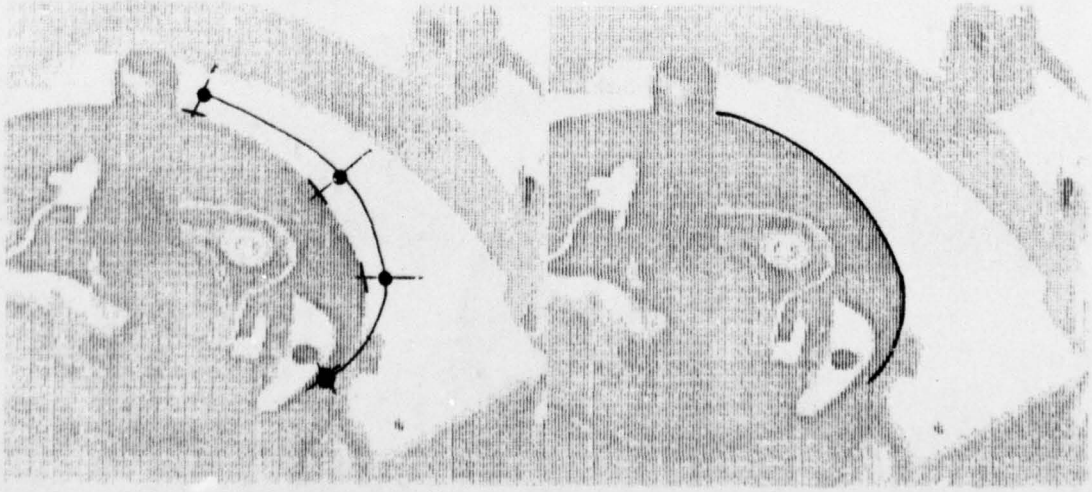


Figure 6.3.3.3.c

Figure 6.3.3.3.d

Section 4
GATHER STATISTICS

1. The Training Session

Given a list of operator/feature pairs, the purpose of the training phase is to gather statistics on the reliability of the operators and determine the distributions of values produced by the operators. This information is passed to the planning phase where it is used to rank the potential operator/feature pairs and to the execution phase where it is used to determine the likelihood ratios that correspond to the values produced by the operators.

The training phase is generally considered to be a one-shot learning process that determines the *a priori* information associated with the operators. In programmable assembly, however, since the same task is repeated many times, additional statistics can be gathered during the execution phase. The system can dynamically maintain models for all of the operators. Dynamic models can adjust for gradual changes, such as gradual lighting changes.

The reliability statistics are based upon the relative number of times the operator locates the correct match. Each time the operator is applied there are three possible outcomes:

(1) the operator locates the correct match,

(2) the operator locates one of the known alternatives,

or (3) the operator locates a surprise.

If there are two known alternatives in addition to the correct match, there are really four possible outcomes. The user supervises the training session and indicates the outcome for each application of each operator (see appendix IV).

The system uses the percentage of times that the operator matches a possibility as an estimate for the *a priori* probability of that possibility. For example, if an operator matches a surprise three times out of thirty applications, the system's estimate for that operator's *a priori* probability of a surprise is 0.10.

The training phase also gathers statistics on the values produced by each operator. If an operator has four possible outcomes, the system gathers statistics to approximate the four

distributions. If the task is an inspection task, the system gathers one set of statistics for the case in which the object is present and another set of statistics for the case in which the object is missing.

The type of statistics gathered depends upon the expected form of the distribution. If the distribution is expected to be a normal distribution, the sum of the values and the sum of the squares of the values are sufficient to determine the distribution uniquely. As mentioned in section 3.6.2 there is a change of variable that converts correlation values into values that form an approximately normal curve. Thus, for correlation operators the changed values are used instead of the correlation coefficients. If an operator produces an unusual distribution and if a scaled version of the histogram is to be used as the distribution of the operator, the system stores the values themselves.

Figure 6.4.1.1 is a flowchart for a training session based upon these ideas. It uses a split screen to display the planning picture on the left and the current training picture on the right. In this way the planning positions and appearances for the features can be overlayed on top of the left picture and the matching positions can be overlayed on top of the right picture. The user can easily see which features are being looked for, where they are located, and whether or not the operator finds the correct match.

This particular training method displays an expected feature and all of its known alternatives on the left and the matching feature on the right (see figure 6.4.1.2). One operator at a time is displayed and the user decides which alternative has been matched (e.g., number 2 in figure 6.4.1.2). A separate display for each operator is necessary if the operators have several known alternatives. However, if only a few of the features have known alternatives it is more efficient to show all of the expected features and all of the matching positions at once (see figure 6.4.1.3). The user can still decide the outcome of each operator and indicate it to the system.

If the operators almost always locate the correct match, the process can be shortened further so that the user only specifies the results for the operators that miss their expected match (see the trace of the training phase in appendix IV). This approach makes it is easy to gather the training information: the user positions the objects at typical positions, takes a picture, applies the operators to the picture, and indicates which ones locate something other than their expected feature.

Given a system that can easily gather the appropriate statistical information, the user still has two concerns: (1) the number of training pictures to use and (2) the arrangement of the objects to use to take the pictures. The set of training pictures should include enough pictures to determine the value distributions of the operators and a sufficient range of pictures to provide the system with a representative set of situations.

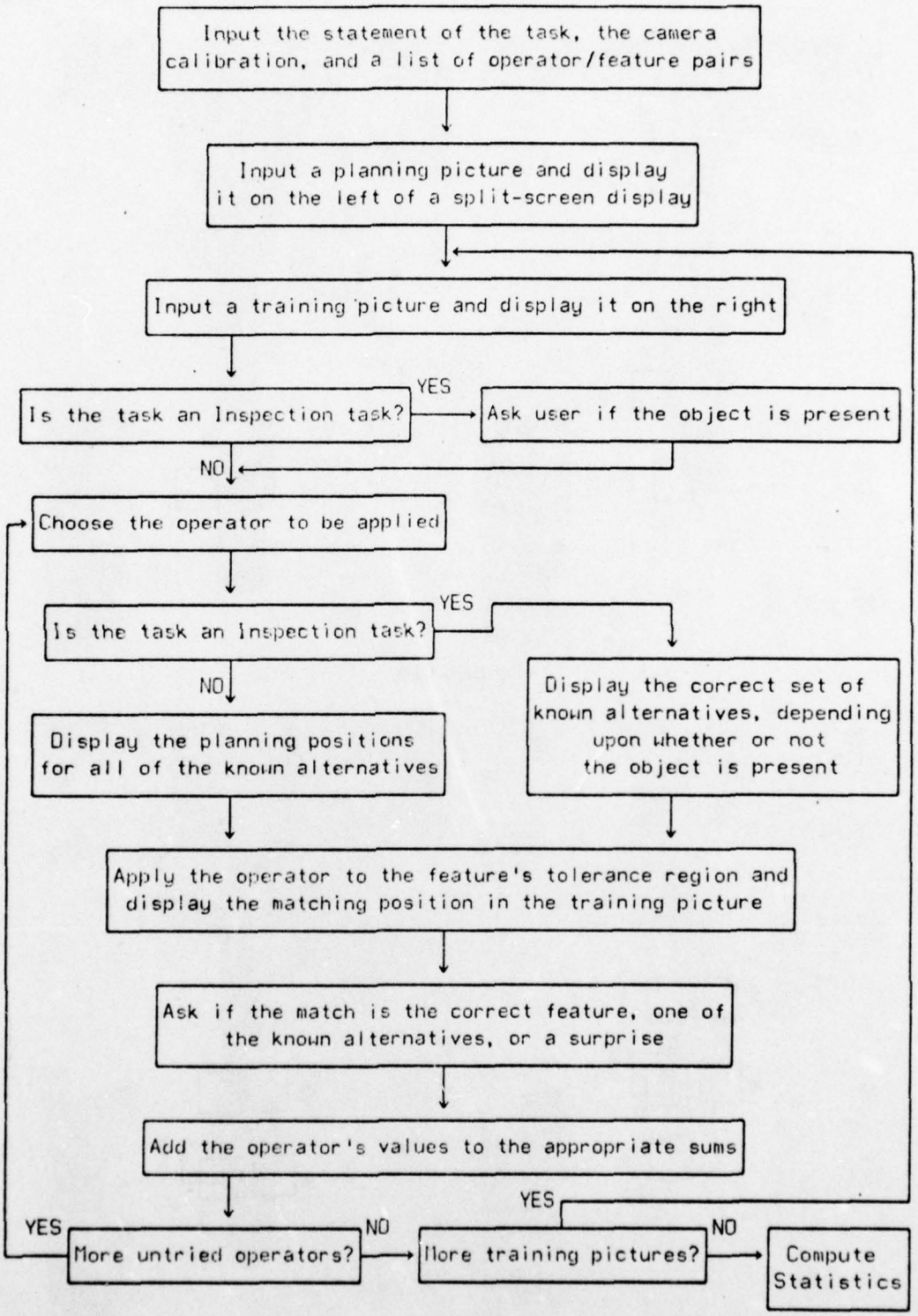

Input the statement of the task, the camera calibration, and a list of operator/feature pairs
Input a planning picture and display it on the left of a split-screen display
Input a training picture and display it on the right
Is the task an Inspection task?
YES
Ask user if the object is present
NO
Choose the operator to be applied
Is the task an Inspection task?
YES
NO
Display the planning positions for all of the known alternatives
Display the correct set of known alternatives, depending upon whether or not the object is present
Apply the operator to the feature's tolerance region and display the matching position in the training picture
Ask if the match is the correct feature, one of the known alternatives, or a surprise
Add the operator's values to the appropriate sums
YES
NO
More untried operators?
YES
NO
More training pictures?
Compute Statistics

Figure 6.4.1.1

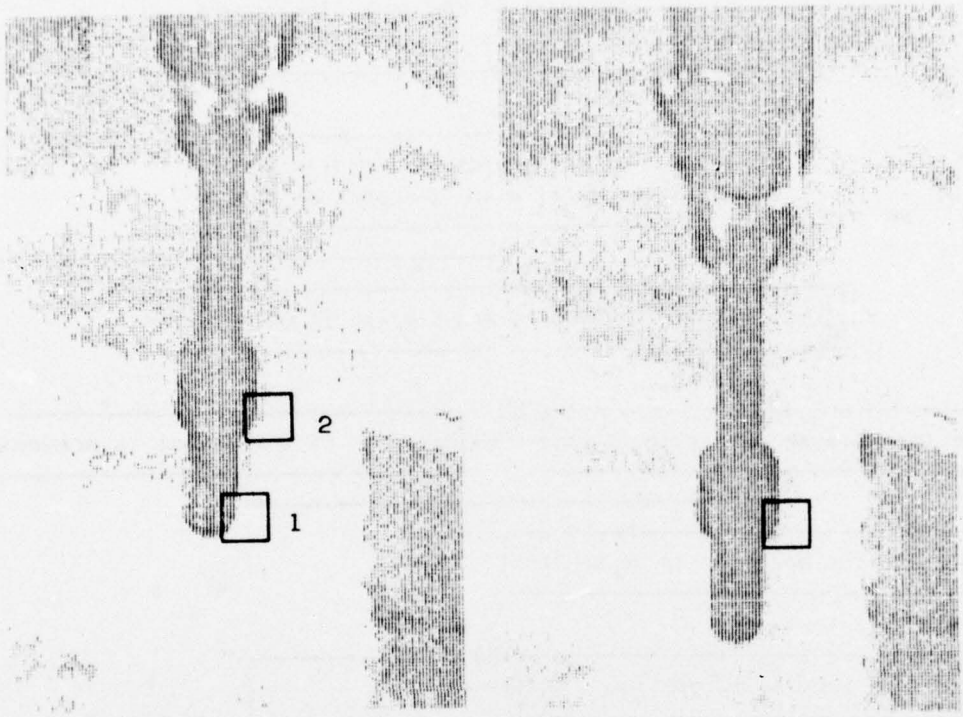


Figure 6.4.1.2

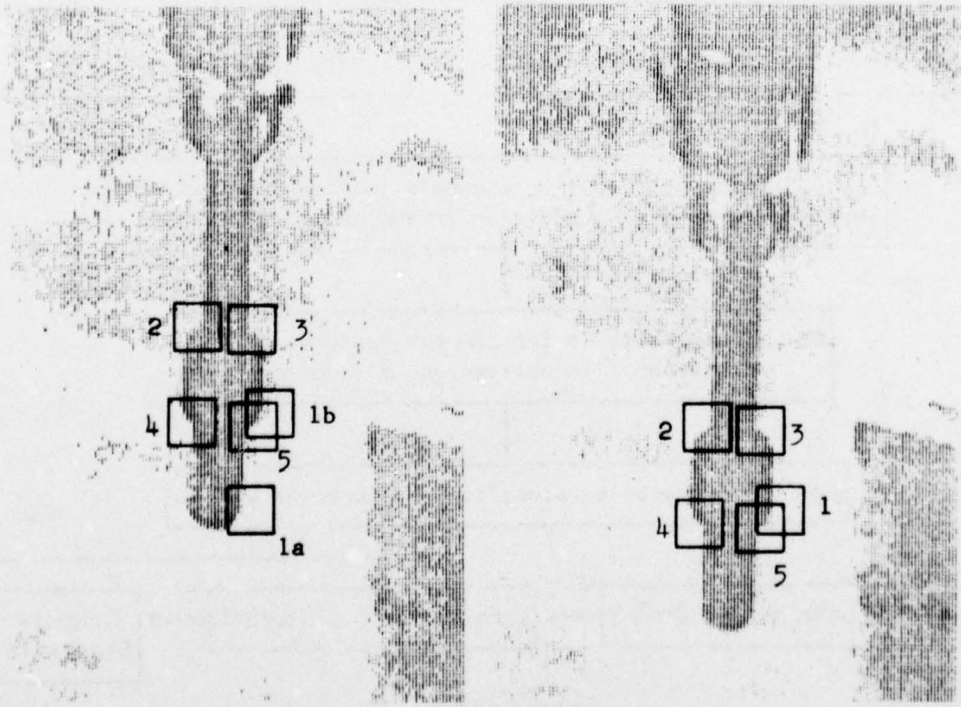


Figure 6.4.1.3

The number of training pictures depends upon the type of the distribution. Twenty to fifty are normally required for normal distributions (see section 6.4.1.2). Scaled histograms often require over a hundred if the values are widely distributed.

It is more difficult to specify which pictures should be used. The overall distribution of pictures should roughly correspond to the expected distribution of object arrangements. The more representative the training pictures are, the better the statistics will be. The set of training pictures should include some of the unusual and special situations in addition to several of the standard arrangements. For example, the set should include pictures of the objects when they are at their extreme locations, such as at their maximum rotations and translations. These pictures insure that the operators can locate their matches when the features are distorted the most. The set should also include any degenerate cases that involve one or more of the features. For example, if a corner is one of the expected features and if an edge on another object happens to appear close to the corner when the objects are in one particular arrangement, that arrangement should be included in the training sequence. In order to present the correct relative importance of the pictures, all of them should be weighted by their *a priori* probability of occurrence.

Given a sufficient number of training pictures that reflect the expected distribution of scenes, the training system, under the supervision of the user, produces behavior models of the operators. The planning system analyzes these models to construct a plan to achieve the goal and the execution system uses these models to make decisions based upon the behavior of the operators when applied to the pictures of unknown situations.

2. *Number of Trials for Normally Distributed Values*

After deciding which distribution to use to model the results of an operator, one still has to decide how many training pictures to use in order to produce a good approximation for the distribution. If the chosen distribution is normal, one needs enough samples to approximate the mean and standard deviation, since a normal distribution is completely determined by these two parameters. How many samples are needed? There are two theorems that help answer this question (see [Hoel 71]):

THEOREM: If X is normally distributed with variance V and

$$Vs = \frac{\sum_{i=1}^{N} (Xi - Ms)^2}{N}$$

is the sample variance based upon a random sample of size N and Ms is the sample mean, then

$$\frac{N*Vs}{V}$$

has a chi-square distribution with (N-1) degrees of freedom.

THEOREM: If X is normally distributed with mean M and variance V and a random sample of size N is taken, then the sample mean Ms will be normally distributed with mean M and variance V/N.

Let CS(n,p) represent the value such that a chi-square distribution with n degrees of freedom has p percent of the population to the right of that value. One application of the first theorem states that there is a ninety-five percent chance that the sample variance and actual variance are related as follows:

$$\text{(6.4.2.1)} \qquad CS((N-1),.975) \le \frac{N*Vs}{V} \le CS((N-1),.025).$$

Let S and Ss represent the standard deviation and the sample standard deviation of the distribution. Since Vs = Ss*Ss and V = S*S, formula 6.4.2.1 can be converted into the following statement concerning the actual and sample standard deviations:

$$\text{(6.4.2.2)} \qquad \frac{sqrt(N) * Ss}{sqrt(CS((N-1),.025))} \le S \le \frac{sqrt(N) * Ss}{sqrt(CS((N-1),.975))}.$$

The second theorem can be used to produce a ninety-five percent confidence interval about the mean. That is,

$$\text{(6.4.2.3)} \qquad |M - Ms| \le \frac{2*S}{sqrt(N)}$$

or, substituting the larger value from (6.4.2.2) into (6.4.2.3) produces

$$\text{(6.4.2.4)} \qquad |M - Ms| \le \frac{2*Ss}{sqrt(CS((N-1),.975))}.$$

For example, if Ms = 1.3 and Ss = .2, the ninty-five percent confidence intervals based upon a sample size of 15 are

(6.4.2.5)
$$|M - Ms| \leq .16950$$
$$\text{and } .15092 \leq S \leq .32824.$$

For a sample size of 30 the intervals are

(6.4.2.6)
$$|M - Ms| \leq .10022$$
$$\text{and } .16151 \leq S \leq .27446.$$

One interesting possibility is to use the planning-time formulas to predict the effect of gathering more samples from an operator's distribution. Two important questions can be answered in this way:

(1) Given a sample mean and a sample standard deviation, plus confidence intervals about them, what is a reasonable, but *conservative* distribution (or set of distributions) that can be used to model the operator?

(2) Given an additional set of N samples from a distribution, what is the probable change in the operator's expected contribution?

In this situation a *conservative* distribution is a distribution that understates the contribution of the operator. The use of such a distribution may require more operators to be applied than theoretically necessary, but there is a smaller chance of making an incorrect decision. For example, assume that a potential operator in an inspection task has the following characteristics:

(6.4.2.7)

(sample size of 15)	
On	Off
Ms = 1.3	Ms = 1.95
Ss = .2	Ss = .22

Assume that the probability of On is .9. Then the expected log-ratio for the operator is 3.41. To pick a more conservative distribution for the operator consider the sixty percent confidence intervals about the means and standard deviations:

(6.4.2.8)

(sample size of 15)	
On	Off
$1.234 \leq M \leq 1.365$	$1.878 \leq M \leq 2.022$
$.182 \leq S \leq .252$	$.200 \leq S \leq .277$

If one assumes that the the most conservative set of distributions is produced at the extremes of these intervals, there are sixteen possible combinations for the pair of distributions to be used to model the operator. Figure 6.4.2.1 shows the expected contribution of the operator for each of the sixteen possibilities. The most conservative set is the set that has the lowest expected contribution, i.e.,

(6.4.2.9)

(sample size of 15)	
On	Off
$M = 1.365$	$M = 1.878$
$S = .182$	$S = .277$

(the expected log-ratio is 1.25).

Expected Log-likelihood Ratios

```
min M1, min S1, min M2, min S2:  4.04
min M1, min S1, min M2, max S2:  1.90
min M1, min S1, max M2, min S2:  6.03
min M1, min S1, max M2, max S2:  2.79

min M1, max S1, min M2, min S2:  4.38
min M1, max S1, min M2, max S2:  2.11
min M1, max S1, max M2, min S2:  6.52
min M1, max S1, max M2, max S2:  3.15

max M1, min S1, min M2, min S2:  2.57
max M1, min S1, min M2, max S2:  1.25 *
max M1, min S1, max M2, min S2:  4.20
max M1, min S1, max M2, max S2:  1.98

max M1, max S1, min M2, min S2:  2.81
max M1, max S1, min M2, max S2:  1.35
max M1, max S1, max M2, min S2:  4.56
max M1, max S1, max M2, max S2:  2.20
```

*** the minimum expected log-likelihood ratio, the most conservative set of distributions**

Figure 6.4.2.1

What is the expected gain from gathering another fifteen samples from the operator's distributions? The intervals are:

(6.4.2.10)

(sample size of 30)			
	On		Off
1.258≤	M ≤1.342	1.904≤	M ≤1.996
.185≤	S ≤.231	.203≤	S ≤.254

and the most conservative distribution (within the sixty percent intervals) is

(6.4.2.11)

(sample size of 30)	
On	Off
M = 1.342	M = 1.904
S =.185	S = .254

(the expected log-ratio is 1.80).

The potential gain is significant in terms of the increase in the expected log-ratio for the conservative set of distributions. More samples would increase the expected log-ratio even further. *The upper limit on this log-ratio would be reached when the conservative set of* distributions was the same as the sample set. At that point the expected log-ratio for both of them would be 3.41. The number of samples actually used in a VV task depends upon how conservative the programmer is, how important execution time is, and how much time can be devoted to training the system. Sample sizes on the order of twenty to fifty have worked well.

In programmable assembly since each VV task is performed repeatedly, it is possible to gather additional samples during production runs. This is important because a larger set of samples can help to refine the model for an operator in two ways. First, more samples can improve the distributions being used to model the operator, and second, if one of the global variables (e.g., lighting or camera sensitivity) changes slowly over time, continuous sampling can maintain an up-to-date model for the operator.

CHAPTER 7

CONCLUSIONS AND EXTENSIONS

There are two main conclusions of this thesis:

(1) A large class of visual feedback tasks can be formulated and accomplished within one framework.

(2) The framework can be implemented in such a way that it is relatively easy for a programmer who is not an expert in vision research to construct programs that perform visual feedback tasks.

The justifications for both of these statements were essentially proofs by construction. A class of visual feedback tasks, referred to as verification vision tasks, was characterized and an interactive system that greatly simplifies the programming of such tasks was implemented.

Section 1
FRAMEWORK FOR VV

A VV task is a task in which the scene is highly predictable; there are no big surprises. The largest single step involved in establishing a framework for VV is the development of a set of combination rules that use this predictability to accomplish the task in as efficient way as possible. Since several types of visual operators have been used extensively in the past, their behaviors are reasonably well understood. It is also relatively clear that the quantities of interest for inspection and location tasks are confidences and precisions. But it is not clear how to combine the results of different operators in order to estimate the quantities of interest.

Two sets of combination rules were derived in the thesis: one for inspection tasks and

one for location tasks. Each of these sets will be discussed in a separate subsection below.

1. Inspection

The inspection rules had to deal with four types of complications:

(1) Different operators return different distributions of values that describe their best local match. One type of operator computes correlation values; another computes the distinctness of an edge.

(2) An operator may not locate a unique feature. It may locate any one of two or three possible features.

(3) An operator may be so unpredictable that it occasionally finds a completely unexpected feature as its best match.

(4) An operator may compute several values that describe the matching feature.

The straightforward extensions of the rules to cover these possibilities involved increasingly complex formulas. A set of assumptions were made in order to make the formulas computable. The most basic assumption was that training examples were available from which the behavior of the operators could be approximated. Fortunately, training examples are available in several types of visual feedback tasks, e.g., programmable assembly, satellite monitoring of crops, and chest x-rays.

The second type of assumption used within the derivations of the inspection rules is the conditional independence of some of the information produced by the operators. These assumptions were critical because they reduced the interdependencies sufficiently so that the rules were computable. The two conditional independence assumptions were:

(1) The value of an operator is conditionally independent of the values of all the other operators.

(2) The value of an operator is conditionally independent of its matching position.

Within programmable assembly these assumptions were found to be approximately true for the set of tasks that were programmed. The first assumption is often true because different operators, especially different types of operators, depend upon different properties of the pictures. The second is generally true because the size of the location uncertainties is so small within VV that the appearances of the features do not change significantly from one position

to the next. It is also important to note that there are techniques that can be used to check the validity of these assumptions for a specific set of operators.

2. *Location*

The combination rules for location tasks systematize a collection of special-purpose rules that use the location of one or more features to help locate other features. Given the location of one hole in a subassembly, the system can predict the location of a second hole. In fact, the system can produce a range of possible locations for the second hole. This range of locations can be interpreted as a precision about that hole. Precisions are important for two reasons: (1) they specify information that is of direct interest to the programmer and (2) they can be used to cull matches that are inconsistent. Culling is important, of course, because the operators are not completely reliable.

A least-squares technique was used to combine the position information for a large number of matches. There are other possible metrics, but least-squares is a well known technique and can be easily extended to include different types of position information, such as a point on a circle.

The location rules can be extended in at least two ways: (1) they could incorporate a wider range of position information and (2) they could adjust a larger set of parameters. The current routines only provide for point-to-point matches. Other possibilities include point-on-a-line matches and point-on-a-circle matches. The parameters could include the six parameters that describe a relative change in the location of the object. The parameters could also represent a relative change in the location of the camera.

The location rules also provide a way to check the consistency of a small number of matches. The consistency is based upon the likelihood that the object would be at the location implied by the matches. If it is impossible or very unlikely that the object is at the implied location, one or more of the matches must be incorrect.

Section 2
EASE OF PROGRAMMING

The difficulty in using the VV system depends upon two factors: (1) the number of steps that are not completely automated and (2) the quality of the human engineering for those steps. The semantic structures necessary to automate some of the steps, such as ranking

operator/feature pairs, have been presented in this thesis. These techniques need to be improved and others have to be added in order to complete the automation of all four stages within VV.

In the programming stage the current system helps the user

(1) choose potential operator/feature pairs by applying an interest operator to the picture,

(2) determine the correct position of the camera by computing the task tolerance volume and by predicting the expected precision produced by the operators,

and (3) recalibrate the camera each time a new location is tried.

The user has to state the task, filter out suggested operator/feature pairs that the system does not know enough to avoid, suggest additional operator/feature pairs, measure the three-dimensional position of the features on the objects, and manually adjust the location of the camera.

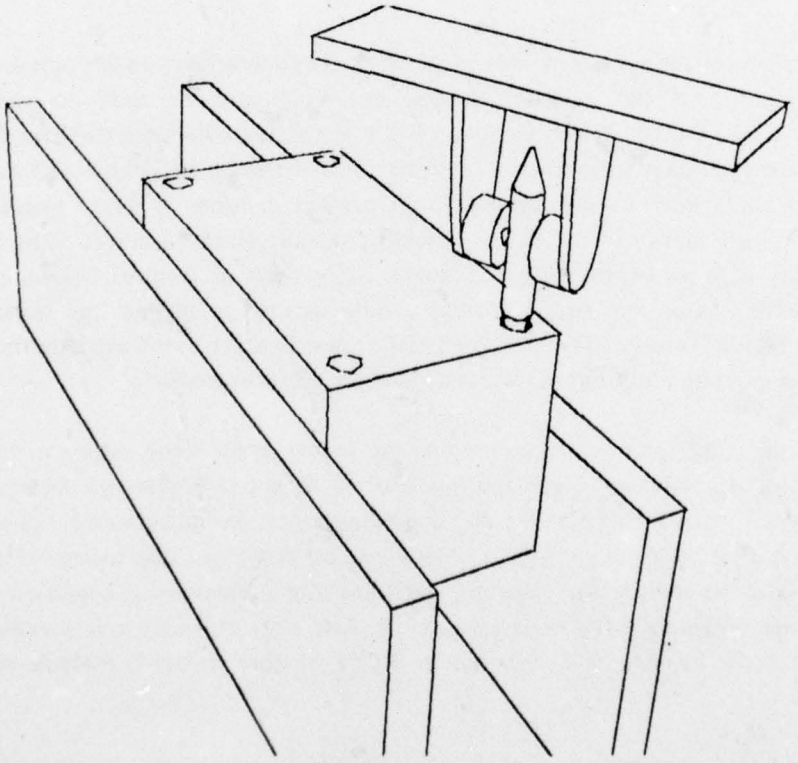
The system can be easily extended to adjust the location of the camera automatically. A pair of cameras and the associated stereo techniques can be used to measure the three-dimensional positions of the features. But it is considerably more difficult to improve the operator/feature pair suggestion subsystem. Two types of suggestion techniques were discussed: (1) apply interest operators to typical pictures in order to locate visually distinct features and (2) use models of the objects to predict visually distinct features. The analysis of pictures needs to be extended so that it locates other types of features besides correlation features. Curve linking and region growing should be used to suggest line features, curve features, and region features. The small step made toward model-based suggestions should be completed and extended to produce several different types of suggestions.

At training time the user has to position the objects in the scene, take a picture, decide how typical the arrangement is, and indicate whether or not each operator locates a known alternative or a surprise. Some of the object positioning can be automated by programming the mechanical arms to sequence through a set of typical locations. The structural consistency tests can be used to convert the reliability decisions into a monitoring operation. That is, after all of the operators have been applied, the best structural match is formed and the system shows it to the user, who only has to decide whether or not the whole structure is correct.

The current planning stage ranks the operator/feature pairs for inspection tasks according to their expected contribution (i.e., their expected log-likelihood ratio) and the

operator/feature pairs for location tasks according to their relative structural consistency. These methods are useful, but they do not take several important factors into account. For example, they do not consider the possibility of expanding the local context about a match. That is, if the results of an operator do not uniquely determine a match, incorporate the position information of other operators to distinguish between the alternatives. A better strategy subsystem is needed that develops more complete plans based upon better estimates for the expected costs, more flexible VV program structures, and better models of the trade-offs between local and global checking.

At programming time the user only monitors the progress of the task. However, the system can still be improved so that it provides more complete diagnostic information if something fails and manual intervention is required. For example, it would be useful if the system could tell the user what it expects the current state of the world is and why an error occurred.



BIBLIOGRAPHY

Agin, Gerald J. [1972], Representation and Description of Curved Objects, Stanford Artificial Intelligence Project Memo No. 173, October 1972.

Agin, Gerald J. and Binford, Thomas O. [1973], Computer Description of Curved Objects, Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, August 1973, pp. 629-640,

Agin, Gerald J., and Duda, Richard O. [1975], SRI Vision Research for Advanced Industrial Automation, Second USA-Japan Computer Conference, pp. 113 - 117.

Ambler,A.P. and Popplestone,R.J. [1973], Inferring the Positions of Bodies from Specified Spatial Relationships, Dept. of Machine Intelligence, University of Edinburgh, Edinburgh, Scotland.

Andrews,H.C. [1972], Introduction to Mathematical Techniques in Pattern Recognition, John Wiley and Sons, Inc., 1972.

Bajcsy,R. [1972], Computer Identification of Textured Visual Scenes, Stanford Artificial Intelligence Project Memo No. 180, October 1972.

Bajcsy,R. [1973a], Computer Description of Textured Surfaces, Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, August 1973, pp. 572-579.

Bajcsy,R. and Tavakoli,M. [1973b], A Computer Recognition of Bridges, Islands, Rivers and Lakes from Satellite Pictures, Conference Proceedings Machine Processing of Remotely Sensed Data, Purdue University, October 1973, pp. 54-68.

Barnea, Daniel I. and Silverman, Harvey F. [1972], A Class of Algorithms for Fast Digital Image Registration, IEEE Transactions on Computers, Vol. C-21, No. 2, February 1972, pp. 179-186.

Barrow,H.G. and Popplestone,R.J. [1971], Relational Description in Picture Processing, Machine Intelligence 6, 1971, pp. 377-396.

Baumgart, Bruce G. [1972], Winged Edge Polyhedron Representation, Stanford Artificial Intelligence Project Memo No. 179, October 1972.

Baumgart, Bruce G. [1973], Image Contouring and Comparing, Stanford Artificial Intelligence Project Memo No. 199, October 1973.

Baumgart, Bruce G. [1974a], GEOMED - A Geometric Editor, Stanford Artificial Intelligence Project Memo No. 232, May 1974.

Baumgart, Bruce G. [1974b], Geometric Modeling for Computer Vision, Stanford Artificial Intelligence Project Memo No. 249, October 1974.

Binford,T.O., Paul,R., Feldman,J.A., Finkel,R., Bolles,R.C., Taylor,R.H., Shimano,B.E., Pingle,K.K., and Gafford,T.A. [1974], Computer Integrated Assembly Systems, Progress Report covering March 1974 to September 1974, prepared for the National Science Foundation, Stanford Artificial Intelligence Project, September 1974.

Binford,T.O., Grossman,D.D., Miyamoto,E., Finkel,R., Shimano,B.E., Taylor,R.H., Bolles,R.C., Roderick,M.D., Mujtaba,M.S., and Gafford,T.A. [1975a], Exploratory Study of Computer Integrated Assembly Systems, Progress Report covering September 1974 to November 1975 prepared for the National Science Foundation. Stanford Artificial Intelligence Project, November 1975.

Binford, Thomas O. [1975b], Optimizing the Hueckel Operator, an internal memorandum at the Stanford Artificial Intelligence Project, December 1975.

Binford,T.O., Grossman,D.D., Liu,C.R., Bolles,R.C., Finkel,R., Mujtaba,M.S., Roderick,M.D., Shimano,B.E., Taylor,R.H., Goldman,R.H., Jarvis,J.P., Scheinman,V., and Gafford,T.A. [1976a], Exploratory Study of Computer Integrated Assembly Systems, Progress Report covering December 1975 to July 1976 prepared for the National Science Foundation, Stanford Artificial Intelligence Project Memo No. 285, July 1976.

Binford,T.O. [1976b], Vision Language Writeup, Stanford Artificial Intelligence Project internal memo, August 1976.

Bolles, Robert C. and Paul, Richard [1973], The Use of Sensory Feedback in a Programmable Assembly System, Stanford Artificial Intelligence Project Memo No. 220, October 1973.

Bolles, Robert C. [1975], Verification Vision within a Programmable Assembly System: An Introductory Discussion, Stanford Artificial Intelligence Laboratory Memo No. 275, December 1975.

Braid,I.C. [1974], Designing with Volumes, Cantab Press, Cambridge, England, 1974.

Braid,I.C. [1975], The Synthesis of Solids Bounded by Many Faces, Communications of the ACM, Volume 18, No. 4, p. 209, April 1975.

Brice,C.R. and Fennema,C.L. [1970], Scene Analysis using Regions, Artificial Intelligence 1, 1970, pp. 205-226.

Chien,R.T. and Jones,V.C. [1975], Acquisition of Moving Objects and Hand-Eye Coordination, The Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, September 3, 1975, pp. 737-741.

Coons,S.A. [1967], Surfaces for Computer-aided Design of Space Forms, MIT Project MAC, MAC-TR-41, June 1967.

Corwin, Daniel W. [1971], Visual Position Extraction Using Stereo Eye Systems with a Relative Rotational Motion Capability, MIT Vision Flash No. 22, January 1971.

Davis, Randall [1976], Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases, Stanford Artificial Intelligence Memo No. 283, July 1976.

Deutsch,E.S. and Belknap,N.J. [1972], Texture Descriptors using Neighborhood Information, Computer Graphics and Image Processing 1, August, 1972, pp. 145-168.

Dewar,R., Lewis,N.R., Rossol,L., and Olsztyn,J. T. [1973], An Application of Computer Vision to do Automatic Wheel Mounting, The First International Joint Conference on Pattern Recognition, October 1973.

Duda,R.D. and Hart,P.E. [1973], Pattern Classification and Scene Analysis, Wiley, 1973.

Duncan, Acheson J. [1952], Quality Control and Industrial Statistics, Richard D. Irwin, Inc., Chicago, Illinois, 1952.

Falk, Gilbert [1970], Computer Interpretation of Imperfect Line Data as a Three-Dimensional Scene, Stanford Artificial Intelligence Project Memo No. 132, August 1970.

Feldman,J.A. and Rovner,P.D. [1969], An ALGOL-Based Associative Language, Communication of the ACM 12,8, August 1969, pp. 439-449.

Feldman,J.A., Low,J., Taylor,R.H., and Swinehart,D. [1972], Recent Developments in SAIL, an Algol Based Language for Artificial

Intelligence, Proceedings of the FJCC, pp. 1193-1202, Stanford Artificial Intelligence Project Memo No. 176, November 1972.

Feldman,J.A. and Sproull,R.F. [1974], Decision Theory and Artificial Intelligence: An Approach to Generating Efficient Plans, draft, July 1974.

Fikes, Richard E. and Nilsson, Nils J. [1971], STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving, Artificial Intelligence, Vol. 2, 1971.

Finin,T. [1973], Tracking Wires on a Circuit Board, MIT Vision Flash No. 52, October 1973.

Finkel,R., Taylor,R., Bolles,R.C., Paul,R. and Feldman,J. [1974], 'AL, A Programming System for Automation, Stanford Artificial Intelligence Project Memo No. 243, November 1974.

Finkel,R., Taylor,R., Bolles,R.C., Paul,R., and Feldman,J. [1975], An Overview of AL, A Programming System for Automation, Proceedings of Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, September 1975, pp. 758 - 765.

Finkel, Raphael A. [1976], Construction and Debugging of Manipulator Programs, Stanford Artificial Intelligence Project Memo No. 284, August 1976.

Fischler, Martin A. [1971a], Aspects of the Detection of Scene Congruence, Proceedings of the Second Annual International Joint Conference on Artificial Intelligence, September 1971, pp. 88-100.

Fischler, Martin A., and Elschlager, Robert A. [1971b], The Representation and Matching of Pictorial Structures, Lockheed Missiles and Space Company, LMSC-D243781, September 1971.

Flock,H.R. [1965], Optical Texture and Linear Perspective as Stimuli for Slant Perception, Psychological Review 72, 1965, pp. 505-514.

Forrest,A.R. [1972], Mathematical Principles for Curve and Surface Representation, Curved Surfaces in Engineering, Churchill College, Cambridge, March 1972.

Forsythe,G.E. and Moler,C.B. [1967], Computer Solution of Linear Algebraic Systems, Prentice-Hall, 1967.

Freuder, Eugene C. [1972], Recognition of Real Objects, MIT Vision Flash No. 33, October 1972.

Ganapathy, Sundaram [1975], Reconstruction of Scenes Containing Polyhedra from Stereo Pair of Views, Stanford Artificial Intelligence Project Memo No. 272, December 1975.

Garvey, Thomas D. [1976], Perceptual Strategies for Purposive Vision, SRI Technical Note 117, September 1976.

Gennery, Donald B. [1975], Least-Squares Stereo-Camera Calibration, Stanford Artificial Intelligence Project internal memo, 1975.

Gill, Aharon [1972], Visual Feedback and Related Problems in Computer Controlled Hand-Eye Coordination, Stanford Artificial Intelligence Project Memo No. 178, October 1972.

Gordon,W.J. and Riesenfeld,R.F. [1972], Bernstein-Bezier Methods for the Computer-aided Design of Free-form Curves and Surfaces, General Motors Research Publication GMR1176, March 1972.

Gould,S.S. [1972], Surface Programs for Numerical Control, Proceedings of the Curved Surfaces in Engineering Conference, Cambridge 1972 pp. 14-18.

Gouraud, Henri [1971], Computer Display of Curved Surfaces, University of Utah Technical Report, UTEC-CSc-71-113, June 1971.

Grape, Gunnar R. [1973], Model Based (Intermediate Level) Computer Vision, Stanford Artificial Intelligence Project Memo No. 201, May 1973.

Graybill,F.A. [1961], An Introduction to Linear Statistical Models, Volume I, McGraw-Hill Book Company, 1961.

Grossman, David D. [1975a], Procedural Representation of Three-Dimensional Objects, IBM Research Report RC-5314, T. J. Watson Research Center, Yorktown Heights, N.Y., March 1975.

Grossman, David D. and Taylor, Russell H. [1975b], Interactive Generation of Object Models with a Manipulator, Stanford Artificial Intelligence Project Memo No. 274, December 1975.

Grossman, David D. [1976], Monte Carlo Simulation of Tolerancing in Discrete Parts Manufacturing and Assembly, Stanford Artificial Intelligence Project Memo No. 280, May 1976.

Hannah, Marsha Jo [1974], Computer Matching of Areas in Stereo Images, Stanford Artificial Intelligence Project Memo No. 239, July 1974.

Hoel,P.G. [1971], Introduction to Mathematical Statistics, John Wiley and Sons, Inc., 1971.

Holland,S.W. [1975] A Programmable Computer Vision System based on Spatial Relationships, Digital Systems Laboratory Technical Report #104, December 1975.

Horn, Berthold K. P. [1970], Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View, MAC-TR-79, MIT, Cambridge, November 1970.

Horn, Berthold K. P. [1971], The Binford-Horn LINE-FINDER, MIT Vision Flash No. 16, July 1971.

Horn, Berthold K. P. [1973], On Lightness, MIT Artificial Intelligence Laboratory Memo No. 295, October 1973.

Horn, Berthold K. P. [1974], The Application of Linear Systems Analysis to Image Processing. Some Notes. MIT Working Paper No. 100, 1974.

Horn, Berthold K. P. [1975a], The Facts Of Light, MIT Working Paper No. 97, May 1975.

Horn, Berthold K. P. [1975b], Image Intensity Understanding, Massachusetts Institute of Technology AIM No. 335, August 1975.

Hueckel, Manfred H. [1969], An Operator Which Locates Edges in Digitized Pictures, Stanford Artificial Intelligence Project Memo No. 105.

Kanade,T. [1973], Picture Processing System by Computer Complex and Recognition of Human Faces, Kyoto University, November 1973.

Kelly, Michael D. [1970], Visual Identification of People by Computer, Stanford Artificial Intelligence Project Memo No. 130.

Krakauer,L.J. [1971], Computer Analysis of Visual Properties of Curved Objects, MAC-TR-82, MIT, Cambridge, May 1971.

Lavin,M.A. [1973], The Gloss of Glossy Things, MIT Vision Flash No. 41, March 1973.

Levine,M.D., O'Handley,D.A., and Yagi,G.M. [1973], Computer Determination of Depth Maps, Jet Propulsion Laboratory, Pasadena, Ca., 1973.

Lieberman, Lawrence I. [1974], Computer Recognition and Description of Natural Scenes, Moore School of Electrical Engineering Technical Report No. 74-08.

Lieberman, Lawrence I. and Wesley, M. A. [1975a], The Design of a Geometric Data Base for Mechanical Assembly, IBM Research Paper No. RC 5489,

June 1975.

Lieberman, Lawrence I. and Wesley, M. A. [1975b], AUTOPASS: A Very High Level Programming Language for Mechanical Assembler Systems, IBM Research Paper No. RC 5599, August 1975.

Lozano-Perez,T. [1973], Finding Components on a Circuit Board, MIT Vision Flash, No. 51, September 1973.

Marr,D. [1975a], ANALYZING NATURAL IMAGES: a computational theory of texture vision, MIT Artificial Intelligence Laboratory Memo No. 334, June 1975.

Marr,D. [1975b], Early Processing of Visual Information, MIT Artificial Intelligence Laboratory Memo No. 340, December 1975.

Maruyama,K. [1972], A Study of Visual Shape Perception, Ph.D. Thesis, University of Illinois at Urbana-Champaign, October, 1972.

Mathlab Group [1974], MACSYMA Reference Manual, Massachusetts Institute of Technology, September 1974.

McCarthy, John [1958], Programs with Common Sense, MIT Artificial Intelligence Laboratory Memo No. 17, November 1958.

Miyamoto, Eiichi and Binford, Thomas O. [1975], Display Generated by a Generalized Cone Representation, Computer Graphics and Image Processing Conference, Anaheim, Ca., May 1975.

Montanari,U. and Reddy,R. [1971], Computer Processing of Natural Scenes: Some Unsolved Problems, in Evans, T. G. (ed.), Artificial Intelligence, AGARD Conference Proceedings No. 94-71, London, 1971.

Moravec, Hans and Gennery, Donald [1976], Cart Project Progress Report, Stanford Artificial Intelligence Project internal memo, October 1976.

Nevatia,R. and Binford,T.O. [1973], Structural Description of Complex Objects, Proceedings of the Third International Conference on Artificial Intelligence, Stanford, August 1973, pp. 641-647.

Nevatia, Ramakant [1974], Structured Descriptions of Complex Curved Objects for Recognition and Visual Memory, Stanford Artificial Intelligence Project Memo No. 250, Octobe[illegible]974.

Nevatia, Ramakant [1975], Object Boundary Determination in a Textual Environment, Proceedings of the ACM Annual Conference, October 1975.

Nevins,J., Whitney,D., Drake,S., Killoran,D., Lynch,M., Seltzer,D., Simunovic,S., Spencer,R.M., Watson,P., and Woodin,A. [1975], Exploratory Research in Industrial Modular Assembly, Charles Stark Draper Laboratory Report R-921, Cambridge, Massachusetts, December 1, 1974 to August 31, 1975.

Newman, William M. and Sproull, Robert F. [1973], Principles of Interactive Computer Graphics, McGraw-Hill, Inc. 1973.

Nilsson, Nils J. [1971], Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, Inc., 1971.

Nilsson, Nils J. (ed.) [1975], Artificial Intelligence - Research and Applications, Stanford Research Institute, May 1975.

PADL [1974], An Introduction to PADL: Characteristics, Structure, and Rationale, University of Rochester Technical Memorandum 22, December 1974.

Paul, Richard [1972], Modelling, Trajectory Calculation and Servoing of a Computer Controlled Arm, Stanford Artificial Intelligence Project Memo No. 177, November 1972.

Paul,R., Bolles,R.C., and Pingle,K.K. [1973], Automated Pump Assembly, Film (16mm, color, silent, 7 minutes), Stanford Artificial Intelligence Project, April 1973.

Paul,R., Bolles,R.C., and Pingle,K.K. [1974], Programmable Assembly, Three Short Examples, Film (16mm, color, sound, 8 minutes), Stanford Artificial Intelligence Project, October 1974.

Perkins, Walter, A. and Binford, Thomas O. [1973], A Corner Finder for Visual Feedback, Stanford Artificial Intelligence Project Memo No. 214, September 1973.

Pingle, Karl K. and Thomas, Arthur J. [1974], A Fast, Feature-Driven Stereo Depth Program, Stanford Artificial Intelligence Project Memo No. 248, May 1975.

Pratt, William K. (ed.) [1976], University of Southern California Semiannual Technical Report, Image Processing Institute, USCIPI Report No. 660, March 1976.

Quam, Lynn H. [1971], Computer Comparison of Pictures, Stanford Artificial Intelligence Project Memo No. 144, May 1971.

Quam, Lynn H., Liebes, Sidney Jr., Tucker, Robert B., Hannah, Marsha Jo and Eross, Botond G. [1972], Computer Interactive Picture Processing,

Stanford Artificial Intelligence Project Memo No. 166, April 1972.

Quam, Lynn H. and Hannah, Marsha Jo [1974], Stanford Automatic Photogrammetry Research, Stanford Artificial Intelligence Project Memo No. 254, November 1974.

Reiser, John F. [1975], BAIL -- A Debugger for SAIL, Stanford Artificial Intelligence Project Memo No. 270, October 1975.

Riesenfeld, Richard F. [1972], Doctoral Thesis, Syracuse University, New York, 1972.

Riesenfeld, Richard F. [1973], Applications of B-spline Approximation to Geometric Problems of Computer-aided Design, University of Utah Technical Report UTEC-CSc-73-126, March 1973.

Riseman,E.M. and Hanson,A.R. [1974], Design of a Semantically Directed Vision Processor, COINS Technical Report 74C-1, University of Massachusetts at Amherst, January 1974.

Roberts,L.G. [1963], Machine Perception of Three-Dimensional Solids, Massachusetts Institute of Technology Technical Report No. 315, May 1963.

Roberts,L.G. [1965], Homogeneous Matrix Representation and Manipulation of N-Dimensional Constructs, Document No. MS1045, Lincoln Laboratory, Massachusetts Institute of Technology, May 1965.

Rosen,C., Nitzan,D., Agin,G., Andeen,G., Berger.J., Eckerle,J., Gleason,G., Hill,J., Kremers,J., Meyer,B., Park,W., and Sword,A. [1974], Exploratory Research in Advanced Automation, Stanford Research Institute Project 2591 Report 2, Menlo Park, California, August 1974.

Rosenfeld, Azriel [1969], Picture Processing by Computer, Computing Surveys, Vol. 1, No. 3, September 1969, pp.147-174.

Rosenfeld, Azriel [1973], Progress in Picture Processing: 1969-71, Computing Surveys, Vol. 5, No. 2, June 1973, pp.81-108.

Sacerdoti, Earl D. [1975a], The Nonlinear Nature of Plans, Stanford Research Institute Artificial Intelligence Center Technical Note 101, January 1975.

Sacerdoti, Earl D. [1975b], A Structure for Plans and Behavior, Stanford Research Institute Artificial Intelligence Center Technical Note 109, August 1975.

Shewhart, Walter A. and Deming, W. Edwards [1939], Statistical Method from the Viewpoint of Quality Control, Lancaster Press, Inc., Lancaster, Pa. 1939.

Shirai, Yoshiaki [1972], Visual Feedback of a Robot in Assembly, ETL A.I.R. Group Memo #1, 1972.

Shirai, Yoshiaki [1973], A Heterarchical System for Recognition of Polyhedra, Artificial Intelligence, Vol. 4, No. 2, 1973.

Shortliffe,E.H. and Buchanan,B.G. [1975], A Model of Inexact Reasoning in Medicine, Mathematical Biosciences 23, 351-379, 1975.

Sobel, Irwin [1970], Camera Models and Machine Perception, Stanford Artificial Intelligence Project Memo No. 121, May 1970.

Sobel, Irwin [1974], On Calibrating Computer Controlled Cameras for Perceiving 3-D Scenes, Artificial Intelligence, Vol. 5, 1974, pp. 185-198.

Sproull, Robert F. [1977], Strategy Construction using a Synthesis of Decision Theoretic and Heuristic Techniques, Stanford University Ph.D. Thesis, 1977.

Taylor, Russell H. [1976], The Synthesis of Manipulator Control Programs from Task-level Specifications, Stanford University Ph.D. Thesis, August 1976.

Tenenbaum, Jay M. [1970], Accommodation in Computer Vision, Stanford Artificial Intelligence Project Memo No. 134, September 1970.

Tenenbaum, Jay M. [1973], On Locating Objects by Their Distinguishing Features in Multisensory Images, Computer Graphics and Image Processing, Vol. 2, No. 3/4, December 1973.

Tenenbaum, Jay M. and Garvey, Thomas D. [1975], Research in Interactive Scene Analysis, Stanford Research Institute Annual Report for Project No. 8721, March 1975.

Tenenbaum,J.M. and Barrow,H.G. [1976], Experiments in Interpretation-Guided Segmentation, Stanford Research Institute Technical Note 123, March 1976.

Thomas, Arthur J. and Binford, Thomas O. [1974], An Information Processing Analysis of Visual Perception: A Review, Stanford Artificial Intelligence Project Memo No. 227, June 1974.

Turner, Kenneth J. [1974], Computer Perception of Curved Objects using a

Television Camera, Ph.D. Thesis, University of Edinburgh, 1974.

VanLehn, Kurt A. (ed.) [1973], SAIL User Manual, Stanford Artificial Intelligence Project Memo 204, July 1973, Updated by Low,J.R., Samet,H.J., Sproull,R.F., Swinehart,D.C., Taylor,R.H., and VanLehn,K.A., March 1974.

Wichman, William [1967], Use of Optical Feedback in the Computer Control of an Arm, Stanford Artificial Intelligence Project Memo No. 56, August 1967.

Widrow, Bernard [1973a], The "Rubber-Mask" Technique -- I. Pattern Measurement and Analysis, Pattern Recognition, Vol. 5, 1973, pp. 175-197.

Widrow, Bernard [1973b], The "Rubber-Mask" Technique -- II. Pattern Storage and Recognition, Pattern Recognition, Vol. 5, 1973, pp. 199-211.

Will, Peter M. and Grossman, David D. [1975], An Experimental System for Computer Controlled Mechanical Assembly, IEEE Transactions on Computers, Vol. C-24, No. 9, September 1975.

Winston,P.H. (ed.) [1975], New Progress in Artificial Intelligence, MIT Artificial Intelligence Laboratory TR-310, September 1974, revised 1975.

Yakimovsky,Y. [1973a], Scene Analysis using a Semantic Base for Region Growing, Stanford Artificial Intelligence Project Memo No. 209, July 1973.

Yakimovsky,Y. and Feldman,J. [1973b], A Semantics-based Decision Theory Region Analyzer, Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, August 1973, pp. 580-588.

Zusne,L. [1970], Visual Perception of Form, Academic Press, New York, 1970.

APPENDIX I. ASSIGNMENTS FOR TWO OPERATORS

This appendix develops the formula for the probability that an assignment of known alternatives to the results of two operators is correct, given the position and value information produced by the operators. Assume that operator 1 has M known alternatives and operator 2 has N known alternatives. Let f1,f2,...fM be the alternatives for operator 1 and g1,g2,...gN be the alternatives for operator 2. Then Bayes' theorem states:

$$(I.1) \quad P[fj,gk \mid v1,p1,v2,p2] = \frac{1}{1 + \frac{P[v1,v2,p1,p2|\neg(fj,gk)]}{P[v1,v2,p1,p2|fj,gk]} * \frac{P[\neg(fj,gk)]}{P[fj,gk]}}.$$

The probability P[fj,gk|v1,p1,v2,p2] represents the probability that the first operator has located alternative fj and that the second operator has located alternative gk, given the value and position information produced by the two operators. In other words, it is the probability that the assignment of fj to operator 1 and gk to operator 2 is correct.

The standard assumption about the conditional independence of the value and position information can be used to reduce (I.1) to

$$(I.2) \quad P[fj,gk \mid v1,p1,v2,p2] =$$

$$\frac{1}{1 + \frac{P[v1,v2|\neg(fj,gk)]}{P[v1,v2|fj,gk]} * \frac{P[p1,p2|\neg(fj,gk)]}{P[p1,p2|fj,gk]} * \frac{P[\neg(fj,gk)]}{P[fj,gk]}}$$

or

(I.3) $$P[fj,gk \mid v1,p1,v2,p2] = \frac{1}{1 + \frac{P[v1|\neg(fj,gk)]}{P[v1|fj]} * \frac{P[v2|\neg(fj,gk)]}{P[v2|fj]} * \frac{P[p1,p2,\neg(fj,gk)]}{P[p1,p2,fj,gk]}}.$$

The probability $P[\neg(fj,gk)]$ can be expressed as

(I.4) $$P[\neg(fj,gk)] = \sum_{\neg(x=j \ \& \ y=k)} P[fx,gy].$$

That is, the probability that at least one of the assignments is incorrect can be expressed as the sum of the probabilities of all of the other possible assignments besides $[fj,gk]$. Given this expression for $P[\neg(fj,gk)]$ it is possible to express $P[p1,p2,\neg(fj,gk)]$ as

(I.5) $$P[p1,p2,\neg(fj,gk)] = \sum_{\neg(x=j \ \& \ y=k)} P[p1,p2,fx,gy]$$

(see section 3.5). The probability $P[v1|\neg(fj,gk)]$ can be expanded in a similar way. Since the value of v1 is assumed to be independent of gy, some of the conditional probabilities can be simplified.

(I.6) $$P[v1 \mid \neg(fj,gk)] = \frac{P[v1, \neg(fj,gk)]}{P[\neg(fj,gk)]}$$

or

(I.7) $$P[v1 \mid \neg(fj,gk)] = \frac{\sum_{\neg(x=j \ \& \ y=k)} P[v1,fx,gy]}{P[\neg(fj,gk)]}$$

or

(I.8) $$P[v1 \mid \neg(fj,gk)] = \frac{\sum_{\neg(x=j \ \& \ y=k)} (P[v1|fx]*P[fx,gy])}{P[\neg(fj,gk)]}.$$

Since

$$P[fx,gy \mid \neg(fj,gk)] = \frac{P[fx,\ gy,\ \neg(fj,gk)]}{P[\neg(fj,gk)]}, \tag{I.9}$$

and

$$P[fx,\ gy,\ \neg(fj,gk)] = P[fx,\ gy] \quad (\text{if } \neg(x=j \ \&\ y=k)), \tag{I.10}$$

formula I.9 can be rewritten as

$$P[fx,gy \mid \neg(fj,gk)] = \frac{P[fx,\ gy]}{P[\neg(fj,gk)]} \quad (\text{if } \neg(x=j \ \&\ y=k)). \tag{I.11}$$

Then formula I.8 simplifies to

$$P[v1 \mid \neg(fj,gk)] = \sum_{\neg(x=j \ \&\ y=k)} (P[v1|fx]*P[fx,gy|\neg(fj,gk)]). \tag{I.12}$$

Formulas I.5 and I.12 can be used to rewrite formula I.3 as

$$P[fj,gk \mid v1,p1,v2,p2] = \frac{1}{1 + \dfrac{\sum_{\neg(r=j \ \&\ s=k)}(P[v1|fr]*Q[r,s]) * \sum_{\neg(w=j \ \&\ x=k)}(P[v2|gx]*Q[w,x]) * \sum_{\neg(y=j \ \&\ z=k)} P[p1,p2,fy,gz]}{P[v1|fj]*P[v2|gk]*P[p1,p2,fj,gk]}}, \tag{I.13}$$

where $Q[r,s]$ represents $P[fr,gs|\neg(fj,gk)]$. Formula I.13 can be further simplified to

(I.14) $P[fj,gk \mid v1,p1,v2,p2] =$

$$\frac{1}{1 + \sum_{\neg(r=j \,\&\, s=k)} \sum_{\neg(w=j \,\&\, x=k)} \sum_{\neg(y=j \,\&\, z=k)} \frac{P[v1|fr]}{P[v1|fj]} * Q[r,s] * \frac{P[v2|gx]}{P[v2|gk]} * Q[w,x] * \frac{P[p1,p2,fy,gz]}{P[p1,p2,fj,gk]}},$$

which is convenient because the important probabilities form ratios that can be easily evaluated from the density functions.

APPENDIX II. BEST KNOWN ALTERNATIVES

This appendix develops a formula that incorporates an operator's position information into the mechanism that decides which known alternative is the best match. The previous method made this decision solely upon the operator's local value information (see formula 3.5.8). That is, the alternative, fj, with the highest value of

(II.1) $$P[fj \mid v]$$

was chosen as the best match.

Consider the following probability:

(II.2) $$P[fj \mid v,p]$$

where p represents the position of the match for the operator, the fj's (j>0) represent the known alternatives for the operator, and f0 represents the surprise (if any) for the operator. Bayes' theorem states:

(II.3) $$P[fj|v,p] = \frac{1}{1 + \frac{P[v,p|\neg fj] * P[\neg fj]}{P[v,p|fj] * P[fj]}},$$

which can be expanded into

(II.4) $$P[fj|v,p] = \frac{1}{1 + \frac{P[v|\neg fj;p] * P[p|\neg fj] * P[\neg fj]}{P[v|fj;p] * P[p|fj] * P[fj]}}.$$

Using the standard assumption that the v's are conditionally independent of the p's this becomes

$$P[fj|v,p] = \frac{1}{1 + \frac{P[v|\neg fj] * P[p,\neg fj]}{P[v|fj] * P[p,fj]}}. \tag{II.5}$$

Since

$$P[\neg fj] = \sum_{k \neq j} P[fk], \tag{II.6}$$

then

$$P[p,\neg fj] = \sum_{k \neq j} P[p,fk]. \tag{II.7}$$

Since

$$P[v|\neg fj] = \frac{P[v,\neg fj]}{P[\neg fj]}, \tag{II.8}$$

the probability $P[v|\neg fj]$ can be expanded in a way similar to (II.7). Then formula II.5 can be rewritten as

$$P[fj|v,p] = \frac{1}{1 + \frac{\sum_{k \neq j} \frac{P[v|fk]*P[fk]}{P[\neg fj]} * \sum_{k \neq j} P[p,fk]}{P[v|fj] * P[p,fj]}}, \tag{II.8}$$

which reduces to

$$P[fj|v,p] = \frac{1}{1 + \sum_{m \neq j} \sum_{n \neq j} \frac{P[v|fm]}{P[v|fj]} * \frac{P[p,fn]}{P[p,fj]} * \frac{P[fm]}{P[\neg fj]}}. \tag{II.9}$$

All of the component probabilities in this formula have been discussed except the P[p|fj] (for all j). What should their values be? One possibility is to use the original constraints on the object to develop an *a priori* probability that a known alternative will appear at a certain location. Another possibility, if sufficient statistics are available, is to use the densities observed during the training session to estimate the probability of finding a match at that point.

The probability for the *surprise* (i.e., P[p|f0]) has to be computed separately because there is no single position associated with a surprise. It can be approximated as follows:

$$\text{(II.10)} \qquad \frac{\langle \text{the sub-region that is consistent with the previous features} \rangle}{\langle \text{the original tolerance region about f1} \rangle}.$$

This ratio is an estimate of the probability that the surprise will accidently be consistent with the previous features. The more features that have been matched, the harder it is for a surprise to look consistent.

The result of this appendix is an improved formula to be used to decide which known alternative is the best match. The formula is an improvement because it is based upon the position information of an operator in addition to the value information.

APPENDIX III. EXPECTED LOG-LIKELIHOOD RATIO FORMULA

Consider an operator that is to be used in an inspection task. If there are no known alternatives or surprises for the operator, and if the operator produces values that are normally distributed, what is the expected value of the logarithm of its likelihood ratio? That is, what is the expected contribution of the operator? Given the means, M1 & M2, and standard deviations, SD1 & SD2, for the two distributions and the *a priori* probability that the object is present, P, there is a simple formula that computes the desired quantity.

This section derives the appropriate formula. After some initial simplifications, MACSYMA is used to perform the necessary symbolic integration.

The expected value can then be computed as follows:

$$\text{expected\_log-ratio} = \int_{-\infty}^{+\infty} \text{log-ratio}(X) * \text{density}(X)\, dX, \tag{III.1}$$

where

$$\text{log-ratio}(X) = \log\left(\frac{\text{On\_density}(X)}{\text{Off\_density}(X)}\right), \tag{III.2}$$

$$\text{density}(X) = P*\text{On\_density}(X) + (1-P)*\text{Off\_density}(X), \tag{III.3}$$

$$\text{On\_density}(X) = \frac{1}{\text{SD1}*\text{SQRT}(2*\pi)} * e^{-\frac{1}{2}*\left(\frac{X - M1}{SD1}\right)^2} \tag{III.4}$$

and

$$\text{(III.5)}\qquad \text{Off\_density}(X) = \frac{1}{\text{SD2}*\text{SQRT}(2*\pi)} * e^{-\frac{1}{2}*\left(\frac{X-M2}{\text{SD2}}\right)^2}$$

Formula III.2 can be simplified as follows:

$$\text{(III.6)}\qquad \text{log-ratio}(X) = \log\left(\frac{\text{SD2}}{\text{SD1}} * e^{\left(-\frac{1}{2}*\left(\frac{X-M2}{\text{SD2}}\right)^2 - -\frac{1}{2}*\left(\frac{X-M1}{\text{SD1}}\right)^2\right)}\right)$$

$$\text{(III.7)}\qquad \text{log-ratio}(X) = \log(\text{SD2}) - \log(\text{SD1}) + -\frac{1}{2}*\left(\frac{X-M2}{\text{SD2}}\right)^2 - -\frac{1}{2}*\left(\frac{X-M1}{\text{SD1}}\right)^2 .$$

Since the integral from minus infinity to plus infinity of a density function is 1.0, (III.1) reduces to:

$$\text{(III.8)}\qquad \text{expected\_log-ratio}(X) = \log(\text{SD2}) - \log(\text{SD1}) + \int_{-\infty}^{+\infty} \left(-\frac{1}{2}*\left(\frac{X-M2}{\text{SD2}}\right)^2 - -\frac{1}{2}*\left(\frac{X-M1}{\text{SD1}}\right)^2\right) * (P*\text{On\_density}(X)+(1-P)*\text{Off\_density}(X))dX.$$

The integral term can be factored into four smaller integrals that are all essentially the same. Their general form is:

$$(III.9)\qquad I(S,M) = \int_{-\infty}^{+\infty} \frac{1}{2}*\frac{(X-M)^2}{S^2} * \frac{1}{S*SQRT(2*\pi)} * e^{-\frac{1}{2}*\frac{(X-M)^2}{S^2}}\, dX.$$

MACSYMA was used to compute I(S,M) and the final formula. A trace of the session with MACSYMA is presented below:

```
THIS IS MACSYMA 259

FIX 259 DSK MACSYM BEING LOADED
LOADING DONE

(C1) F(X,S,M):=(1/(S*SQRT(2*%PI)))*%E↑(-1/2*((X-M)/S)↑2);

                                                    X - M 2
                                                - 1/2 (-----)
                                 1                    S
(D1)          F(X, S, M) := ------------- %E
                            S SQRT(2 %PI)

(C2) G(X,S,M):=(1/2*((X-M)/S)↑2);

                                      1  X - M 2
(D2)                    G(X, S, M) := - (-----)
                                      2    S

(C3) P*G(X,S2,M2)*F(X,S1,M1);
```

$$(D3) \qquad \frac{P\ (X - M2)^2\ \%E^{-\frac{(X - M1)^2}{2\ S1^2}}}{2\ SQRT(2)\ SQRT(\%PI)\ S1\ S2^2}$$

```
(C4) INTEGRATE(D3,X,MINF,INF);


DEFINT FASL DSK MACSYM BEING LOADED
LOADING DONE

LIMIT FASL DSK MACSYM BEING LOADED
LOADING DONE

RESIDU FASL DSK MACSYM BEING LOADED
LOADING DONE
Is  S1  zero or nonzero?

NONZERO;
```

$$(D4) \quad P\ (2\ SQRT(2)\ SQRT(\%PI)\ S1^3 + SQRT(2)\ SQRT(\%PI)\ (2\ M2^2 - 4\ M1\ M2 + 2\ M1^2)\ S1)/(4\ SQRT(2)\ SQRT(\%PI)\ S1\ S2^2)$$

```
(C5) RATSIMP(%);
```

```
                          2      2              2
                       P S1  + (M2  - 2 M1 M2 + M1 ) P
(D5)                   -------------------------------
                                     2
                                  2 S2
```

(C6) FACTOR(%);

```
                           2     2              2
                        P (S1  + M2  - 2 M1 M2 + M1 )
(D6)                    -----------------------------
                                     2
                                  2 S2
```

(C7) R(SD1,MN1,SD2,MN2):=(SD2↑2+(MN1-MN2)↑2)/(2*SD1↑2);

```
                                          2              2
                                       SD2  + (MN1 - MN2)
(D7)        R(SD1, MN1, SD2, MN2) := -------------------
                                                2
                                           2 SD1
```

(C8) P*R(S2,M2,S1,M1)+(1-P)*R(S2,M2,S2,M2)-
P*R(S1,M1,S1,M1)-(1-P)*R(S1,M1,S2,M2);

```
                  2            2         2            2
       (1 - P) (S2  + (M1 - M2) )   P (S1  + (M2 - M1) )   P   1 - P
(D8) - -------------------------- + --------------------- - - + -----
                    2                          2           2     2
                 2 S1                       2 S2
```

The final formula for the expected contribution of an operator that produces normally distributed values is:

(III.10) expected_log-ratio = log(SD2) - log(SD1) + 1/2 - P

$$+ P*\frac{SD1^2 + (M2 - M1)^2}{2 * SD2^2} - (1-P)*\frac{SD2^2 + (M2 - M1)^2}{2 * SD1^2}.$$

APPENDIX IV. SCREW-INSPECTION EXAMPLE

Appendices IV and V are designed to demonstrate the capabilities of the current implementation of the VV system. Both of the appendices consist of annotated traces of the interaction between a programmer and the VV system. Appendix IV describes the standard screw-inspection task. Appendix V describes the location of the screw hole.

Appendix IV has four sections:

(1) The first section is a trace of all four stages for the screw-inspection task: programming, training, planning, and execution.

(2) At the end of the planning stage the program outputs a file that contains all of the information needed to perform the task. The second section discusses this task file.

(3) In the first section a calibration object is used to calibrate the camera. However, since the calibration is essentially another VV task, it is possible to use a location-type VV task to calibrate the camera directly with respect to some of the fixed objects in the scene, such as the screw dispenser. The third section shows how the system can be used to perform this type of calibration for the screw-inspection task.

(4) The VV program that performs the screw-inspection task has been interfaced to the arm control system, AL. The fourth section describes the interface and presents the arm program that uses visual feedback to check for a screw on the end of the screwdriver.

The notational conventions for the traces are: (1) the output of the computer is normally left-justified, (2) when a response is requested from the user, the request ends in a colon and the user's response immediately follows the colon, and (3) the comments that have been added to the trace are in italics and are indented several spaces.

Section 1

TRACE OF A SCREW-INSPECTION TASK

This trace demonstrates the interaction between a programmer and the VV system. Of particular interest is the planning stage that ranks the potential operator/feature pairs and estimates the expected number of operators required to achieve a certain confidence.

PROGRAMMING TIME

The programming stage involves (1) calibrating the camera to the workstation and (2) pointing out distinctive features on the object of interest. Operators will be used to try to locate these features. The results of the operators will be used to decide whether or not the object is present.

Do you want to use the camera? (Y or N):N

A yes answer to this question means that live pictures from the camera will be taken when they are needed. A no answer means that stored disk pictures will be used.

Calibration picture:SCAL1.PIC

SCAL1.PIC is a picture of the calibration object, which has been placed at a known location within the workstation (see figure IV.1.1).

Display on the synthesizer, halftones, or none? (S,H,or N):H

THE PICTURE'S CHANNEL AND THE OVERLAY CHANNEL: 27 28

The synthesizer and halftone displays are two different ways to display the pictures of interest. Both methods display two pictures side-by-side. The planning picture is normally on the left and the test picture is on the right.

Want to use a calibration file? (Y or N):N

Since the calibration file is about to be defined, this question is answered with a no.

Start calibrating the camera.

Size of the interest operator (4,8,...):8

The interest operator is the operator that is applied to a planning picture in order to locate promising features.

Size of the correlation operator (eg. 9):9

Automatic, Manual, or Standard Calibration? (A, M, or S):M

The standard calibration is the calibration for the camera when it is aimed at the middle of the workstation. It is used when the precision of the calibration is not important.

The automatic calibration uses a calibration file to recalibrate the camera, which is assumed to be close to the location specified in the file. The default calibration file is the standard calibration file.

The manual calibration asks the user to point out features in the picture of the calibration object. The interest operator is applied to the picture of the calibration object, one suggestion at a time is shown on the display, and the user chooses the features he likes. The programmer has several alternatives when presented with a suggestion: (1) accept it, (2) reject it, (3) locally adjust its position, (4) apply the associated operator to the test picture on the right (if there is one), (5) display its screen and workstation coordinates, (6) apply the associated operator to the trial picture on the right in order to find the three or four best alternatives (in addition to the expected match), and (7) stop considering

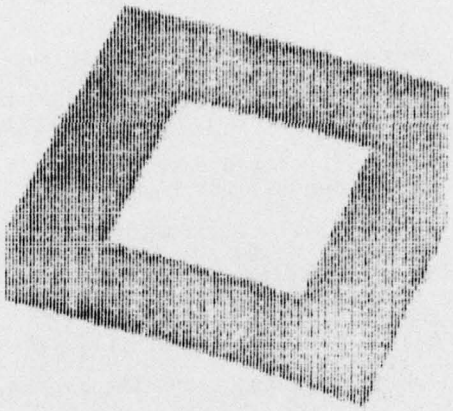


Figure IV.1.1

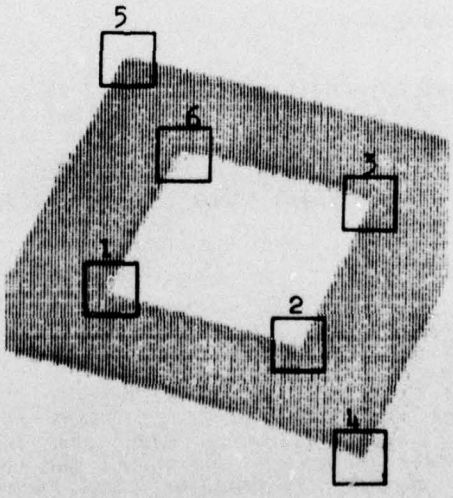


Figure IV.1.2

suggestions produced by the interest operator. The same selection process is used to define features for a task as for a calibration.

The system displays the first suggestion in the planning picture and presents the user with the following menu of responses.

```
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):L
X and Y:-2
X and Y:
```

The user moved the suggestion two pixels to the left in order to center it better on the feature of interest. The null response after the second request for an X and Y is simply a carriage return, <cr>, which indicates that there is no further adjustment desired.

```
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):A
The feature's X,Y,Z: .5 1.5 0
```

The user specifies the workstation coordinates for each feature that he accepts. Given the screen coordinates and the workstation coordinates for several features, the program can calibrate the camera.

```
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):A
The feature's X,Y,Z: .5 .5 0
```

After two or three suggestions the system abbreviates the menu.

```
(A,R,L,T,C,F, or S):L
X and Y:+0 -1
X and Y:+0 -2
X and Y:
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: 1.5 .5 0
(A,R,L,T,C,F, or S):L
X and Y:-2 +2
X and Y:
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: 0 0 0
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):L
X and Y:+0 -7
X and Y:
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: 2.0 2.0 0
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):S
Do you want to point out any additional features? (Y or N):Y
```

The user decided to stop considering the suggestions of the interest operator and to specify one more feature. Currently typed numbers are used to position and adjust the cursor. A joystick or mouse would be more efficient.

```
X and Y for new feature (in pixels) (<cr> to stop):40 75
(A,R,L,T,C,F, or S):L
X and Y:+0 -1
X and Y:
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: 1.5 1.5 0
X and Y for new feature (in pixels) (<cr> to stop):
```

Figure IV.1.2 shows the features that were defined on the calibration object.

```
Approximate lens center (X,Y,Z):-22.5 -6.25 18.75
```

The approximate position of the lens center is used in the camera calibration routine. Given that information, the system can calibrate the camera and output the calibration file, which contains enough information to recalibrate the camera automatically from a picture taken at a slightly different location. The system does not immediately output the

calibration file because the camera may have to be adjusted in order to accomplish the desired task. Thus, the user states the task and makes sure that the desired results can be produced with the camera at the current location before outputing the calibration file.

State the task. Specify the type of task and the known constraints on the object of interest.

```
TYPE OF TASK? (0, location; 1, inspection):1
Size of the interest operator (4,8,...):8
Size of the correlation operator (eg. 9):9
```

This version of the VV system only has one type of operator: Moravec's correlation operator.

```
Known limits on X, (-dx,+dx):-.3 +.3
Known limits on Y, (-dy,+dy):-.3 +.3
Known limits on Z, (-dz,+dz):-.3 +.3
Expected precision of the correlation operator, (eg. 1 pixel):1
```

The expected precision of an operator is the average distance between the estimate of the location produced by the operator and the actual location of the feature. Given the expected precision of an operator in a two-dimensional picture and a camera calibration, it is possible to determine the expected three-dimensional precision about a point in the workstation coordinate system. For example, as stated below, given the current calibration, an error of one pixel can produce as much as an error of .028548 inches along the X-axis when the correlation tries to locate the lower left corner of the calibration object.

```
Do you want to determine the picture size? (Y or N):N
```

The above question asks the user if he wants to use the expected precision of the operators to determine (1) the size of the picture required to insure that the features of interest will be in the picture and (2) the resolution of the picture required to achieve the desired precision. Since the user can easily see that the uncertainties associated with the screw will not move the screw outside the current picture, and since precision is not important in this case, the user decides not to determine the picture size.

If the user is not sure about the size and resolution of the picture, he can use the system to determine them. The camera may have to be moved and/or zoomed so that it takes the correct pictures. If it is adjusted in any way, it has to be recalibrated. Appendix V presents an example of this adjustment and recalibration process.

Since the user decided not to adjust the camera for this task, the system states the current expected three-dimensional precision of a correlation operator centered on the origin of the calibration object and then asks the user if he wants to save the calibration file.

```
X,Y,Z precision of op:      .028548     .022472     .022626
Do you want to save this calibration file? (Y or N):Y
Output file name:IV.CAL
```

The calibration file has been completed. If the camera is moved to take a picture for another task and then moved back to approximately the same location for the screw-inspection task, this file can be used to recalibrate the camera automatically.

```
Use a task file?.(Y or N):N
```

Choose the potential operator/feature pairs for the screw inspection.

```
Planning picture:SPLAN1.PIC
Picture:SIEST1.PIC
```

The two pictures are shown in figure IV.1.3. The first suggestion produced by the interest operator is also shown on the left.

```
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):T
```

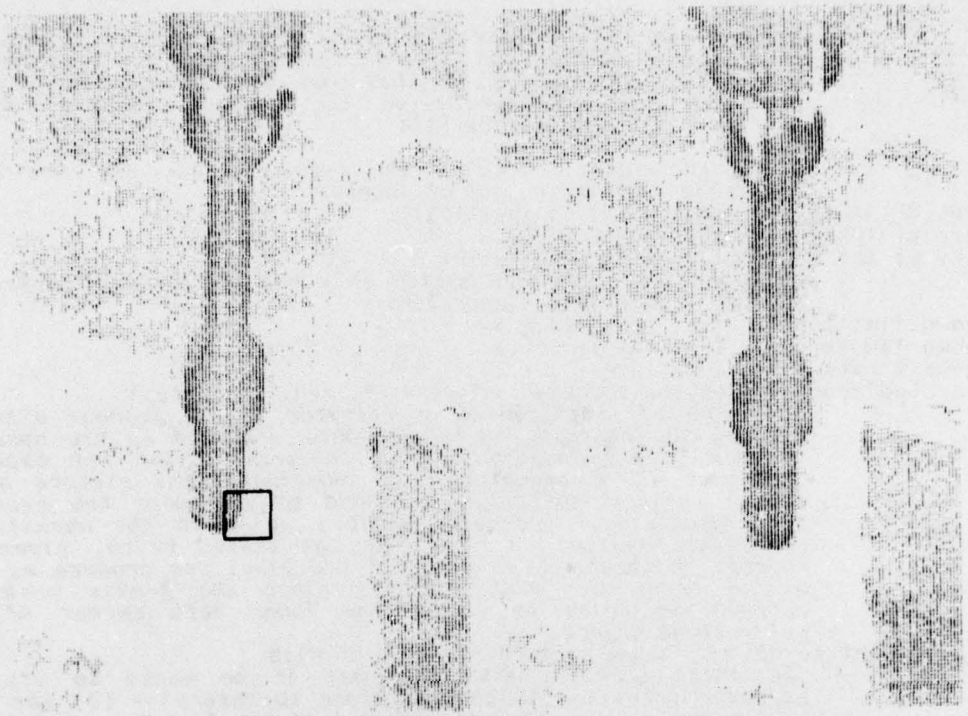


Figure IV.1.3

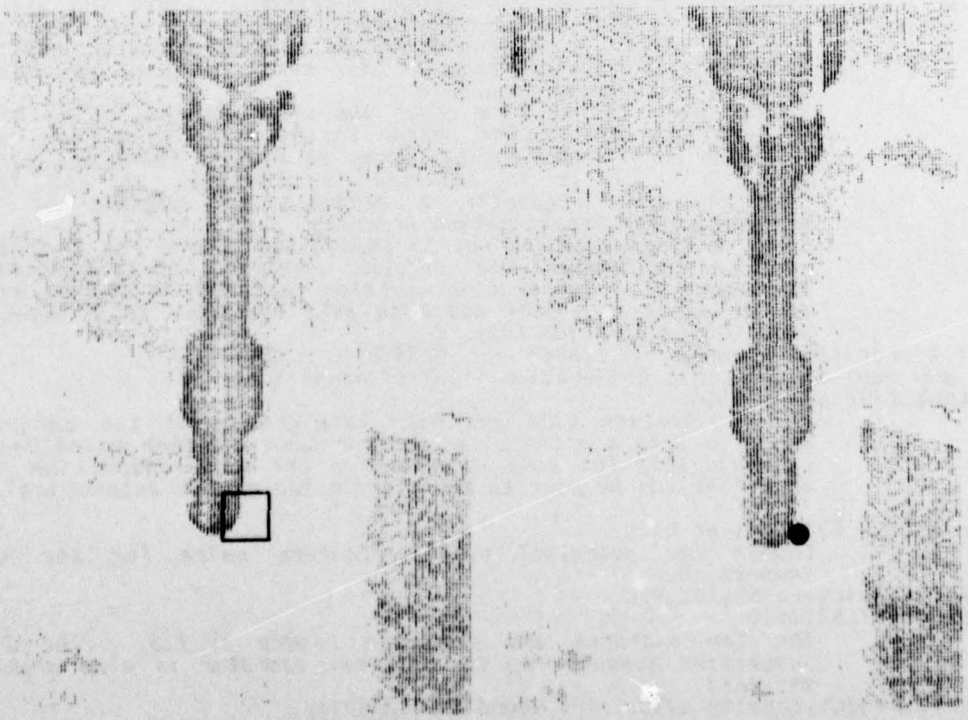


Figure IV.1.4

```
CORRELATION VALUE  .922
```

Each suggestion defines an operator/feature pair. The feature is the part of the object that produces the portion of the picture covered by the suggestion. The operator is the correlation operator defined by the array of intensity values in the suggested portion of the picture.
The user decided to test the operator defined by the first suggestion. The operator was applied to the test picture and a dot was placed on its best match (see figure IV.1.4). The value at the best match was also printed. If the value is too low, or if the operator finds the wrong feature, the user may decide to discard the feature, locally adjust it, or try to characterize the known alternatives.

```
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):C
```

The user asks for the coordinates of the current suggestion. After asking for the height of the feature the system states three sets of coordinates for the feature: (a) the pixel position within the picture, (b) the position within the picture stated in terms of percentages, and (c) the three-dimensional position within the calibration object's coordinate system.

```
Height of the feature:0.0
PIXELS (X,Y): 46 111
(0 → 1)(X,Y):  .460  .743
3-D  (X,Y,Z):  .441  .957  .000
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):A
Z:0.0
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):T
CORRELATION VALUE  .890
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):A
Z:2.5
(A,R,L,T,C,F, or S):T
CORRELATION VALUE  .913
(A,R,L,T,C,F, or S):L
X and Y:+1 -1
X and Y:
(A,R,L,T,C,F, or S):T
CORRELATION VALUE  .937
(A,R,L,T,C,F, or S):A
Z:.5
(A,R,L,T,C,F, or S):L
X and Y:-2
X and Y:
(A,R,L,T,C,F, or S):A
Z:.5
(A,R,L,T,C,F, or S):L
X and Y:-4 -1
X and Y:
(A,R,L,T,C,F, or S):A
Z:.75
(A,R,L,T,C,F, or S):L
X and Y:+0 -2
X and Y:
(A,R,L,T,C,F, or S):T
CORRELATION VALUE  .934
(A,R,L,T,C,F, or S):A
Z:1.0
```

The system ran out of good suggestions, so it asks the user if he would like to define any other features interactively.

```
Do you want to point out any additional features? (Y or N):Y
X and Y for new feature (in pixels) (<cr> to stop):40 80
(A,R,L,T,C,F, or S):L
X and Y:+2 -5
```

```
X and Y:+0 -1
X and Y:
(A,R,L,T,C,F, or S):T
CORRELATION VALUE  .931
(A,R,L,T,C,F, or S):A
Z:.75
X and Y for new feature (in pixels) (<cr> to stop):
Approximate screen coords of the point of interest (x,y):45 110
X and Y:+1
X and Y:
The Z of the point of interest, Z:0.0
X,Y,Z precision of op:  .029  .023  .022
```

Figure IV.1.5 shows the features defined for the task. Notice that one of the features (number 2) is on the screwdriver. It is not expected to contribute very much, but it is included to show how the ranking process (performed during planning time) rates such features. Feature number 6 is similar. It is based upon the screw dispenser and hence should not be able to distinguish between the situation in which the screw is on the end of the screwdriver and the situation in which the screw is missing. Feature number 1 is based upon the tip of the screw. Since the tip of the screwdriver looks very similar to the tip of the screw, operator 1 is not expected to be very distinguishing either.

TRAINING TIME

During the training session the operators are applied to several example pictures and the results are used to form the expected distributions of values for the operators and the expected reliabilities of the operators.

```
Training Picture:SIEST1.PIC
```

For each training picture the planning positions for the features are shown on the left and the actual matches are shown on the right (see figure IV.1.6). If one of the operators misses its designated feature, the user specifies its number and the program incorporates that information into its reliability statistics. The operators did not miss any of their features in this training picture.

```
Do you want to include the results of this trial? (Y or N):Y
Number of a feature that missed, (N or <cr>):
Is the object there or not? (T or N):T
Another trial? (Y or N):Y
Training Picture:SIEST2.PIC
Do you want to include the results of this trial? (Y or N):Y
Number of a feature that missed, (N or <cr>):2
Number of a feature that missed, (N or <cr>):
```

Operator 2 missed its designated feature because the feature is not in the picture.

```
Is the object there or not? (T or N):T
Another trial? (Y or N):Y
Training Picture:SIEST3.PIC
Do you want to include the results of this trial? (Y or N):Y
Number of a feature that missed, (N or <cr>):
Is the object there or not? (T or N):T
Another trial? (Y or N):Y
Training Picture:SIEST4.PIC
```

Figure IV.1.7 shows SIEST4.PIC in which the screw is missing. The overlay shows the best match for each of the operators. Operators two and six found their correct matches. The others, since they are based upon the screw, which is not present, can not possibly locate their correct matches. So they find something else. These makeshift matches, when the object is missing, are not considered to be mismatches in the reliability statistics for the operator. A more general system would assign two correct feature matches for each operator: one when

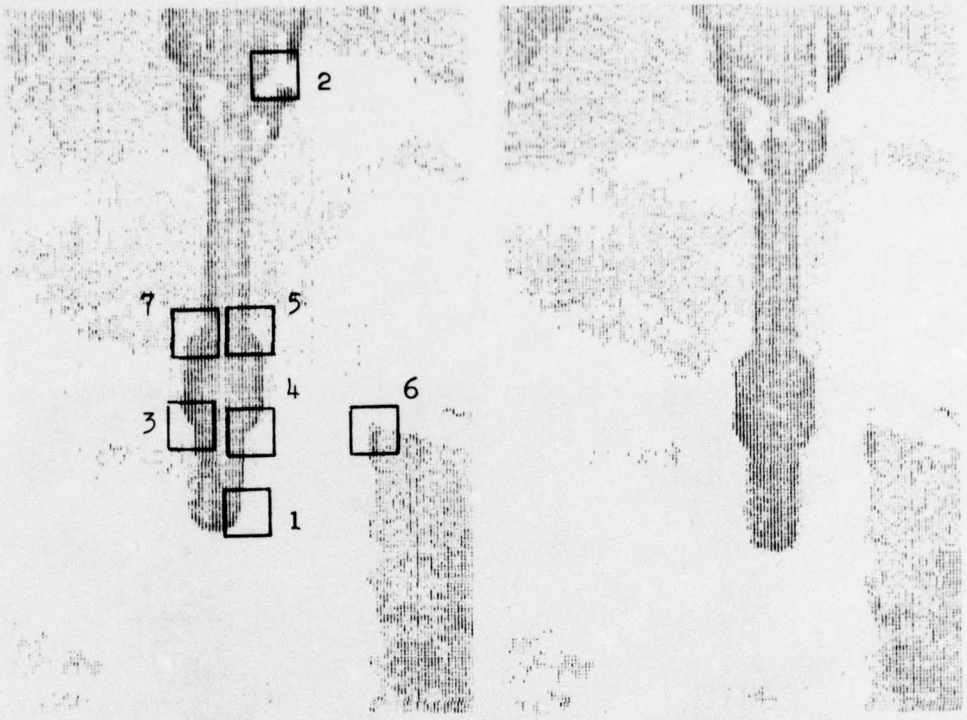


Figure IV.1.5

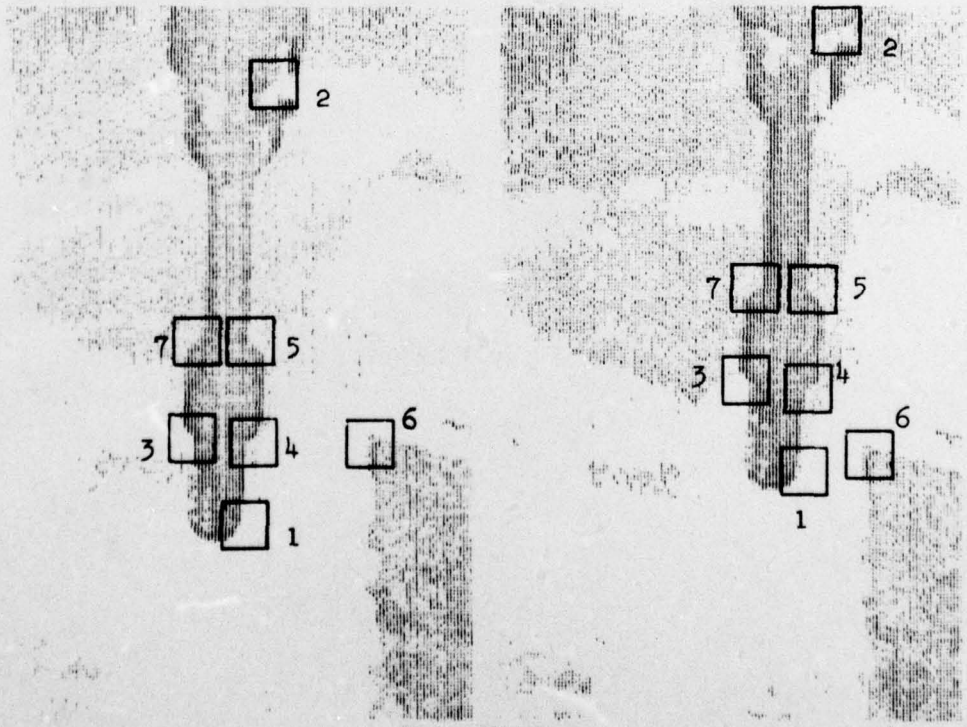


Figure IV.1.6

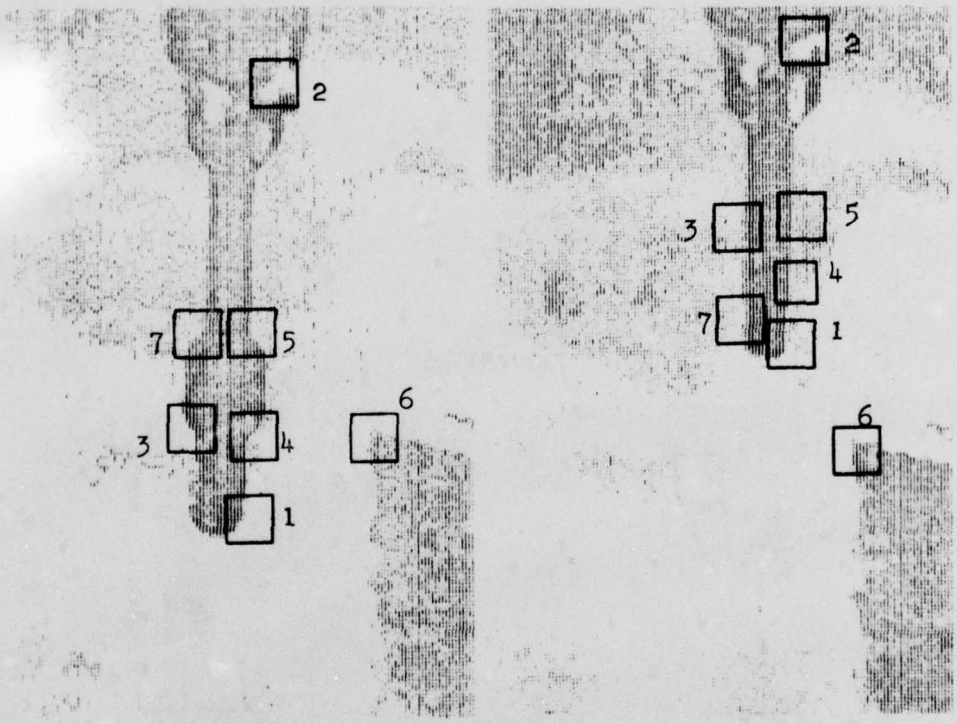


Figure IV.1.7

the object is present and one when the object is missing. An even more general system would allow the user to define features in two planning pictures: one with the object present and one with the object missing.

```
Do you want to include the results of this trial? (Y or N):Y
Number of a feature that missed, (N or <cr>):
Is the object there or not? (T or N):N
Another trial? (Y or N):Y
Training Picture:STEST5.PIC
Do you want to include the results of this trial? (Y or N):Y
Number of a feature that missed, (N or <cr>):
Is the object there or not? (T or N):N
Another trial? (Y or N):Y
Training Picture:STEST6.PIC
Do you want to include the results of this trial? (Y or N):Y
Number of a feature that missed, (N or <cr>):2
Number of a feature that missed, (N or <cr>):
Is the object there or not? (T or N):N
Another trial? (Y or N):Y
```

. *A total of forty training picture were used. Given the training pictures, the training only takes about twenty to thirty seconds elapsed time per picture.*

.

```
Another trial? (Y or N):N
```

The system gathered statistics on the correlation coefficient values produced by the operators. It actually uses the change of variable discussed in section 3.6.2, which produces a distribution that is closer to a normal distribution. The change of variable also modifies the range of values from [-1.0,+1.0] for the correlation coefficients to [-∞,+∞] for the changed values.

FINAL STATISTICS

		(there)			(not there)	
op #	N	MEAN	SD	N	MEAN	SD
1	19.000000	1.238000	.3400000	20.0000000	1.196000	.4630000
2	15.000000	1.224000	.1830000	14.0000000	1.199000	.1500000
3	20.000000	2.053000	.2850000	20.0000000	1.252000	.3110000
4	20.000000	1.917000	.3370000	20.0000000	1.223000	.3280000
5	20.000000	1.870000	.2660000	20.0000000	1.179000	.3620000
6	19.000000	1.245000	.4150000	20.0000000	1.143000	.2540000
7	20.000000	1.947000	.2140000	20.0000000	1.312000	.1490000

PLANNING TIME

The main purpose of the planning-time processing is to decide which operator/feature pairs are the best. That is, determine which operators are expected to contribute the most toward the decision. For an inspection task the contribution of an operator/feature pair is measured by its expected log-likelihood ratio. The log-likelihood ratio is a measure of the operator's ability to distinguish between the two possibilities: (1) the screw is present and (2) the screw is missing. The larger, the better.

After computing the expected log-likelihood ratios for all of the operator/feature pairs, the planning stage forms an ordered list of operators to be applied to a picture. This ordered list of operator/feature pairs is essentially the program to be used to decide whether or not a screw is present. The operators will be applied one at a time and their results will be sequentially incorporated into the overall probability that the screw is present.

Given the ordered list of operator/feature pairs, the planning-time program can use the theorem discussed in section 5.5 to estimate the number of operators required to reach a certain confidence.

```
What is the a priori prob that the object is present:.9
THE ORDERED SET OF FEATURES ... # and the expected log-likelihood ratio
7    7.875
3    2.596
4    1.803
5    1.361
6     .363
1     .062
2     .049
```

Notice that the three dubious features are given considerably lower ratings than the other four features, which were expected to be distinguishing. Figure IV.1.8 shows the pairs of distributions associated with all of the operators. The graphs are presented left-to-right and top-to-bottom in the order determined by the ranking scheme. Notice that the first four operators produce distinct distributions, one when the screw is present and a different one when the screw is missing. The two distributions for each of the worst three operators are essentially the same. They do not distinguish between the two possibilities.

```
Expected log-likelihood ratios given THERE & NOT THERE
1      9.2506001      -4.5068238
2      3.3239587      -3.9576148
3      2.2391645      -2.1211752
4      1.8999531      -3.4920176
5       .4244353      -.2084504
6       .0825244      -.1260499
7       .0592360      -.0441135
```

These conditional expected log-likelihood ratios are needed by the theorem that estimates the number of operators required to reach a certain confidence. Notice that the operator/feature pairs have been renumbered according to their expected log-likelihood ratios. Given these ratios, the user can specify the a priori probability of the object being there and the desired confidence and the system will estimate the number of operators that will have to be applied to achieve the desired confidence.

In this situation, a desired confidence of .99 means that the estimate of the probability that the object is present must be above .99 in order to say that the object is present. In order to say that the object is not present, the estimate of the probability that the object is present must be less than .01.

```
a priori probability of THERE & desired confidence, (P & C):.9 .99
Expected number of operators: 1
a priori probability of THERE & desired confidence, (P & C):.9 .999
Expected number of operators: 1
a priori probability of THERE & desired confidence, (P & C):.9 .9999
Expected number of operators: 2
a priori probability of THERE & desired confidence, (P & C):.5 .99
Expected number of operators: 1
a priori probability of THERE & desired confidence, (P & C):.5 .999
Expected number of operators: 2
a priori probability of THERE & desired confidence, (P & C):
```

If the desired confidence is .999, changing the a priori probability that the screw is present from .9 to .5 increases the expected number of operators from one to two.

```
Do you want to save this task? (Y or N):Y
Output file name:IV.1SK
```

An annotated listing of this task file is presented in the second section of this appendix.

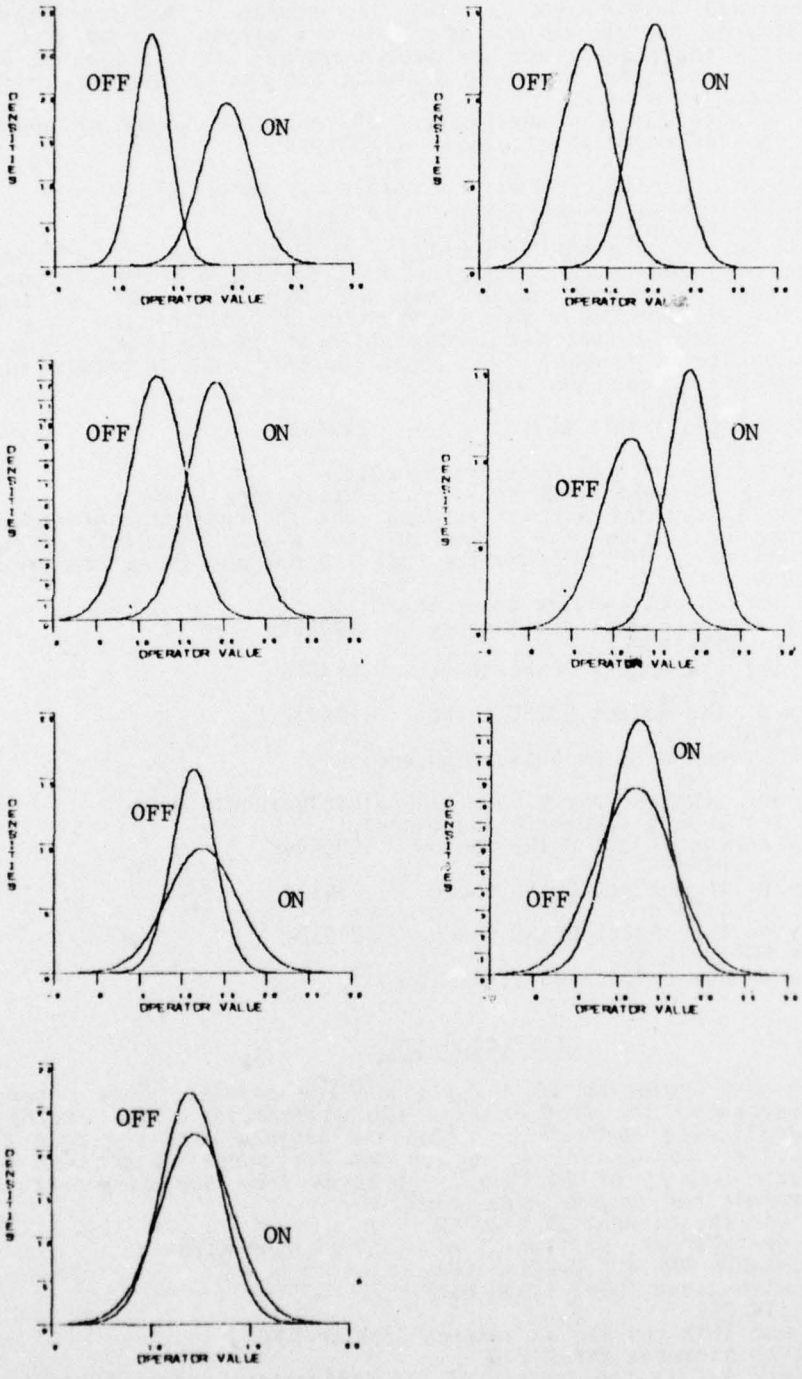

OFF
ON
OPERATOR VALUE
OFF
ON
OPERATOR VALUE
OFF
ON
OPERATOR VALUE
ON
OFF
OPERATOR VALUE
OFF
ON
OPERATOR VALUE
ON
OFF
OPERATOR VALUE
OFF
ON
OPERATOR VALUE

Figure IV.1.8

TRIAL EXECUTION TIME

The trial executions provide a way for the user to check the overall performance of the VV program. He specifies the desired confidence and the a priori probability of the screw not being present and the system applies one operator at a time and prints the resulting estimate for the probability that the object is present.
The user can also doublecheck the expected number of operators required to reach a certain confidence.

```
Picture:SEXEC1.PIC
Desired confidence, (0:take anything → 1:absolutely sure):.99
Probability of the object **NOT** being there:.1
The probability of the object BEING there      .900000
```

Since the a priori probability that the screw is missing has been stated to be .1, the estimate for the probability that the screw is present, before any operators have been applied, is .9. The system prints the overall probability that the screw is present before each application of an operator. Thus, the sequence of probabilities shows the effect of incorporating the results of each new operator.

```
OPERATOR VALUE    1.730069
The probability of the object BEING there      .994820
THE OBJECT IS THERE
Same test picture, New one, or Quit; (S,N,orQ):S
Desired confidence, (0:take anything → 1:absolutely sure):.999
```

The same trial picture is used, but the desired confidence has been raised to .999. One operator was sufficient to reach a confidence of .99. Notice that two operators are required for .999.

```
Probability of the object **NOT** being there:.1
The probability of the object BEING there      .900000
OPERATOR VALUE    1.730069
The probability of the object BEING there      .994820
OPERATOR VALUE    1.847515
The probability of the object BEING there      .999012
THE OBJECT IS THERE
Same test picture, New one, or Quit; (S,N,orQ):N
Picture:SEXEC2.PIC
Desired confidence, (0:take anything → 1:absolutely sure):.999
Probability of the object **NOT** being there:.1
The probability of the object BEING there      .900000
OPERATOR VALUE    1.242828
The probability of the object BEING there      .030154
OPERATOR VALUE    1.180364
The probability of the object BEING there      .000321
THE OBJECT IS **NOT** THERE
Same test picture, New one, or Quit; (S,N,orQ):Q
```

EXECUTION TIME

At the beginning of a day's run (or possibly more often, if necessary) the program uses the calibration file, IV.CAL, to recalibrate the camera. Then the program uses the task file, IV.TSK, to decide whether or not the screw is present after each attempt at picking up a screw from the dispenser. An example run is presented below.

```
Do you want to use the camera? (Y or N):N
Display on the synthesizer, halftones, or none? (S,H,or N):H
THE PICTURE'S CHANNEL AND THE OVERLAY CHANNEL:        24          25
Want to use a calibration file? (Y or N):Y
Input file name:IV.CAL
Do you want to use this cal for an auto-cal? (Y or N)Y
Second calibration picture: SCAL2.PIC
```

SCAL2.PIC is the picture of the calibration object taken by the camera at its current location. Given this new picture and the

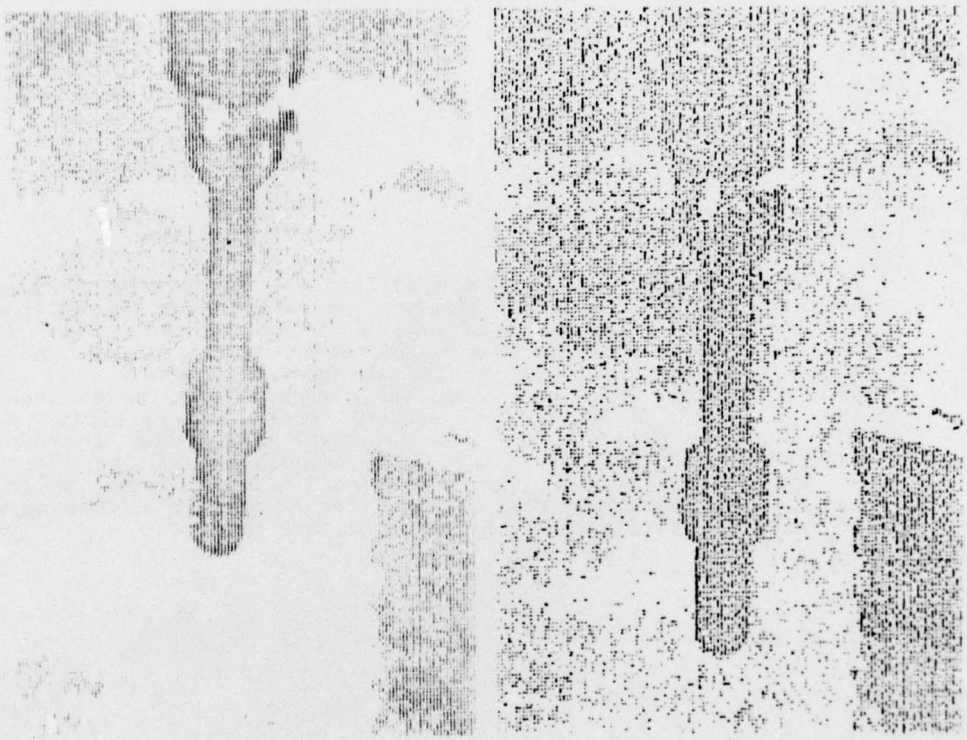


Figure IV.1.9

```
                calibration file, the system can automatically recalibrate the
                camera by applying the operators and recomputing the
                transformations from the resulting positions.
Display the calibration pictures? (Y or N):N
Use a task file? (Y or N):Y
Input file name:IV.TSK
Desired confidence, (0:take anything → 1:absolutely sure):.99
Probability of the object **NOT** being there:.1
                The current task file does not contain the a priori probability
                that the object is missing or the desired confidence. The user
                has to specify these values at the beginning of the run.
Picture:SEXEC3.PIC
THE OBJECT IS THERE
Picture:SEXEC4.PIC
THE OBJECT IS **NOT** THERE
Picture:SEXEC5.PIC
THE OBJECT IS THERE
Picture:SEXEC6.PIC
OPERATOR VALUE  .666
*** AN UNEXPECTED VALUE FOR THE OP ***
OPERATOR VALUE  .782
*** AN UNEXPECTED VALUE FOR THE OP ***
OPERATOR VALUE  .711
*** AN UNEXPECTED VALUE FOR THE OP ***
*** SOMETHING IS GLOBALLY WRONG ***
COULD NOT MAKE A DECISION
                If an operator is applied to a picture and it returns a value
                that is not within three standard deviations of one of the
                expected values, the value is labelled an unexpected value. If
                three unexpected values are found during the execution of a
                task, the program gives up and decides that there must be
                something globally wrong. Consider figure IV.1.9, which shows
                the picture SEXEC6.PIC. Notice that the picture is
                considerably darker and less distinct than the training
                pictures. Such a picture may occur because one of the lights
                at the workstation is out or because the sensitivity of the
                camera is incorrectly set. Since three unexpected values were
                produced for this picture, the program gave up.
Picture:SEXEC7.PIC
THE OBJECT IS THERE
Picture:

End of SAIL execution
```

Section 2
TASK FILE FOR THE SCREW-INSPECTION TASK

The VV system outputs two types of files: calibration files and task files. They contain enough information to perform the specified calibration or VV task. They contain a statement of the task, the positions of the features, the statistical information, and some calibration information. The formats for the two types of tasks are similar.

This section lists the task file, IV.TSK, produced at the end of the planning stage in the trace presented in the previous section. The explanatory comments are surrounded by

parentheses.

```
          1      (type of task)
          8      (size of interest operator)
          9      (size of correlation operator)
          7      (number of operator/feature pairs)

            (the coordinates of the features)
#  (screen coordinates)      (workstation coordinates)
0  .5000000    .7466667    .4857472   1.0245698    .0000000 (pt of interest)
1  .4500000    .4200000    .5777173   1.1651461    .7500000 (lower lt corner)
2  .4400000    .5800000    .3500250   1.0988000    .5000000 (upper lt corner)
3  .5700000    .5866667    .4171681    .8819159    .5000000 (upper rt corner)
4  .5800000    .4266667    .6495881    .9423332    .7500000 (lower rt corner)
5  .7800000    .6200000   -.0484375    .1298235   1.0000000 (dispenser)
6  .5000000    .7466667    .4843507   1.0222903    .0000000 (tip of screw)
7  .5800000    .1486487   -.3501775    .5763804   2.5000000 (screwdriver)

      (the statistics for the features)
0  .0000000    .0000000    .0000000    .0000000  (point of interest)
                .0000000    .0000000    .0000000
1  .0000000  20.0000000   1.947000    .2140000  (lower left corner)
             20.0000000   1.312000    .1490000
2  .0000000  20.0000000   2.053000    .2850000  (upper left corner)
             20.0000000   1.252000    .3110000
3  .0000000  20.0000000   1.917000    .3370000  (upper right corner)
             20.0000000   1.223000    .3280000
4  .0000000  20.0000000   1.870000    .2660000  (lower right corner)
             20.0000000   1.179000    .3620000
5  .0000000  19.0000000  1.5280833    .1528299  (dispenser)
             20.0000000   .9890672    .4479825
6  .0000000  19.0000000   1.238000    .3400000  (tip of screw)
             20.0000000   1.196000    .4630000
7  .0000000  15.0000000  1.2240288    .1829201  (screwdriver)
             14.0000000  1.1985279    .1500302

SPLAN1.PIC    (the planning picture)

(the known and desired constraints & confidences)
  -.3000000    .0000000    .3000000
  -.3000000    .0000000    .3000000
  -.3000000    .0000000    .3000000
   .0000000    .0000000    .0000000
   .0000000    .0000000    .0000000
   .0000000    .0000000    .0000000
   .4857472   1.0245698    .0000000
   .0000000    .0000000    .0000000
   .0000000    .0000000    .0000000
   .0000000    .0000000    .0000000
   .0000000    .0000000    .0000000
   .0000000    .0000000    .0000000
   .0000000    .0000000    .0000000
  1.0000000
   .0000000

(the expected precision of one operator)
   .0280977    .0233368    .0216070

 (the calibration information)
 23.6444930-57.5268910 99.7464230
-35.3809860-17.8524160 152.6100400
   .0862069    .0038601  1.0000000
-22.5000000 -6.2500000 18.7500000
   .0000000    .0000000   .0000000
   .0059713   -.0187518  2.2660957
  -.0157162   -.0048718  2.3111191
  -.0004541    .0016353   .7957259
```

```
.0000000   .0000000   .0000000
.0000000   .0000000   .0000000
```

Section 3
CALIBRATION WITH RESPECT TO THE SCREW DISPENSER

A camera calibration and a VV location task are very similar. Both tasks locate features in a picture of the scene and use the correspondence between the screen positions of the features and the workstation positions of the features to compute a set of parameters. The difference is in the set of parameters that are computed. In a camera calibration the parameters define the camera's position and orientation with respect to the objects in the scene. In a VV location task the parameters specify the relative change of an object from one location in the workstation coordinate system to another.

It would be possible to extend the VV system to compute the parameters for the camera. However, this section does not make that extension. Instead, it shows that it is possible to assume that the changes in the appearance of an object caused by a mislocation of the camera are due to a change in the location of the object. This assumption is possible because there is an equivalent change in the location of a object for each change in the location of the camera. If the changes are relatively small, it does not matter which assumption is made. Thus, a standard VV location task can be used to locate the object. Having found it, the screen positions of the features and the workstation positions of the features can be given to the calibration routine to compute the necessary parameters for the camera. This particular example calibrates the camera with respect to the screw dispenser.

```
Do you want to use the camera? (Y or N):N
Calibration picture:DCAL1.PIC
```

Consider figure IV.3.1, which shows the new calibration picture, DCAL1.PIC. The picture shows the screw dispenser on the right and the screwdriver with a screw on the end at the left. The programmer is going to use the four corners on top of the screw dispenser to calibrate the camera.

```
Display on the synthesizer, halftones, or none? (S,H,or N):H
THE PICTURE'S CHANNEL AND THE OVERLAY CHANNEL:        20          21
Want to use a calibration file? (Y or N):N
Size of the interest operator (4,8,...):8
Size of the correlation operator (eg. 9):9
Automatic, Manual, or Standard Calibration? (A, M, or S):M
```

The programmer uses the interest operator to pick out the first two corners on the screwdispenser and then gives up on it. He points out the final two corners himself.

```
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):L
X and Y:+2
```

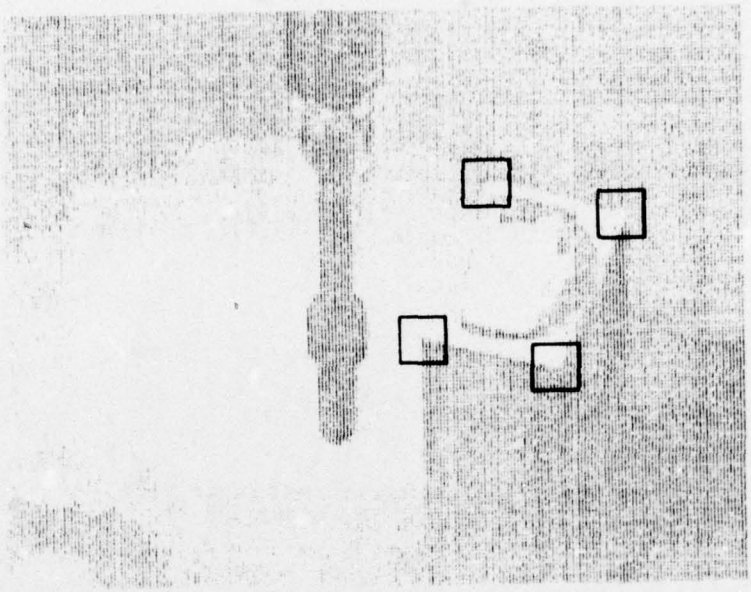


Figure IV.3.1

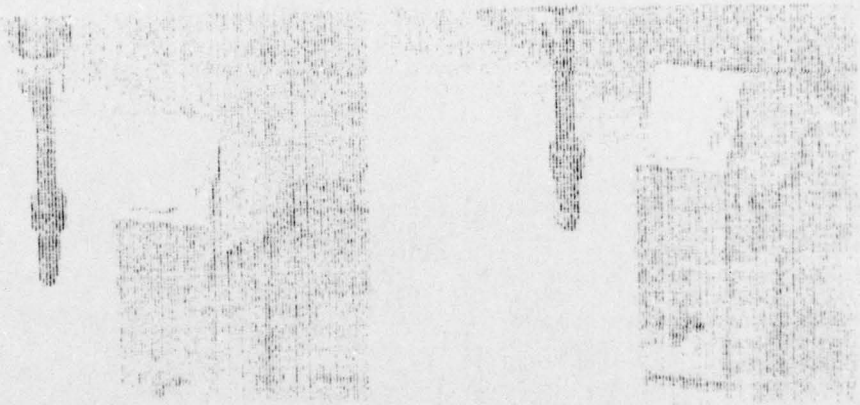


Figure IV.3.2

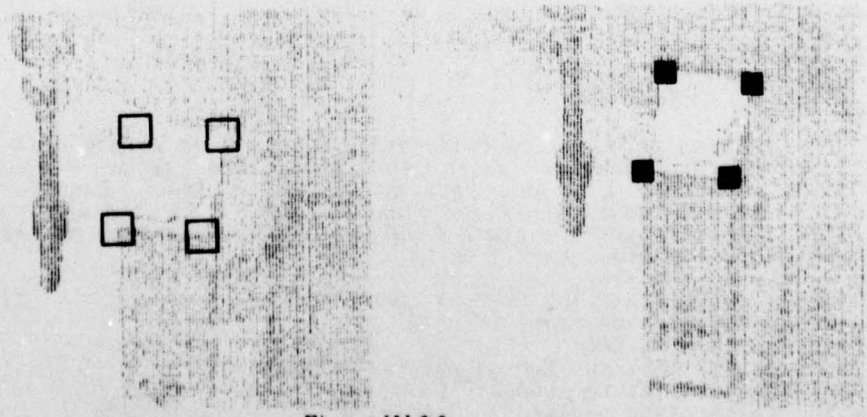


Figure IV.3.3

```
X and Y:+0 +1
X and Y:+0 +1
X and Y:+1
X and Y:
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):A
The feature's X,Y,Z: -.625 -.875 1.0
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):L
X and Y:+2 -2
X and Y:+0 -1
X and Y:
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: .625 -.875 1.0
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):S
Do you want to point out any additional features? (Y or N):Y
X and Y for new feature (in pixels) (<cr> to stop):100 90
(A,R,L,T,C,F, or S):L
X and Y:-5 -4
X and Y:
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: -.625 0 1.0
X and Y for new feature (in pixels) (<cr> to stop):110 70
(A,R,L,T,C,F, or S):L
X and Y:-10 -20
X and Y:+0 -2
X and Y:+5
X and Y:-1
X and Y:
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: .625 0 1.0
X and Y for new feature (in pixels) (<cr> to stop):
Approximate lens center (X,Y,Z):25.25 3.75 18.25
TYPE OF TASK? (0, location; 1, inspection):0
Size of the interest operator (4,8,...):8
Size of the correlation operator (eg. 9):9
Known limits on X, (-dx,+dx):-.6 .6
Known limits on Y, (-dy,+dy):-.6 .6
Known limits on α, (-dα,+dα):-10 10
Desired limits on X, (-dx,+dx):-.1 .1
Desired limits on Y, (-dy,+dy):-.1 .1
Desired limits on α, (-dα,+dα):-2 2
RX:    6.000000     RY:   6.000000
Expected precision of the correlation operator, (eg. 1 pixel):1
Do you want to determine the picture size? (Y or N):N
X,Y,Z precision of op:     .033397     .023377     .020751
Do you want to save this calibration file? (Y or N):Y
Output file name:DISPEN.CAL
```

Having defined the calibration file, it is possible to use the operator/feature pairs to locate the screw dispenser at execution time and recalibrate the camera. Consider figure IV.3.2, which shows the planning picture for the calibration on the left and a picture from a new camera position on the right.

```
Do you want to use the camera? (Y or N):N
Display on the synthesizer, halftones, or none? (S,H,or N):H
THE PICTURE'S CHANNEL AND THE OVERLAY CHANNEL:        20          21
Want to use a calibration file? (Y or N):Y
Input file name:DISPEN.CAL
Do you want to use this cal for an auto-cal? (Y or N)Y
Second calibration picture: DCAL2.PIC
Display the calibration pictures? (Y or N):N
```

Figure IV.3.3 shows the matches for the four features. These results are used to recalibrate the camera. Having recalibrated the camera the standard task file can be used to decide whether or not the screw is present.

```
Use a task file? (Y or N):Y
Input file name:IV.TSK
Picture:DEXEC1.PIC
THE OBJECT IS THERE
Picture:DEXEC2.PIC
THE OBJECT IS **NOT** THERE
Picture:DEXEC3.PIC
        .
        .
        .
End of SAIL execution
```

Section 4
INTERFACE TO THE MECHANICAL ARM

This section briefly describes the interface between the arm control system, AL (see [Finkel 75]), and the VV system. The purpose of the interface is to make it possible for a programmer and several processes (possibly running on several different processors) to work cooperatively on a task. Finkel designed and implemented an interface called ALAID for the Stanford hand/eye system (see Finkel [76]). ALAID provides several capabilities including the following:

(1) One process can ask another process for information, such as status information or the value of a variable.

(2) One process can ask another process to change some of its internal information, such as the value of one of its variables.

(3) The processes can use the standard synchronization primitives, SIGNAL and WAIT, to control their interactions.

These capabilities are sufficiently flexible to provide the basic tools needed for a sophisticated debugging system and visual feedback.

For visual feedback the system consists of several different components that have to communicate with each other in various ways:

(A) the arm and the PDP11 that servos the arm,

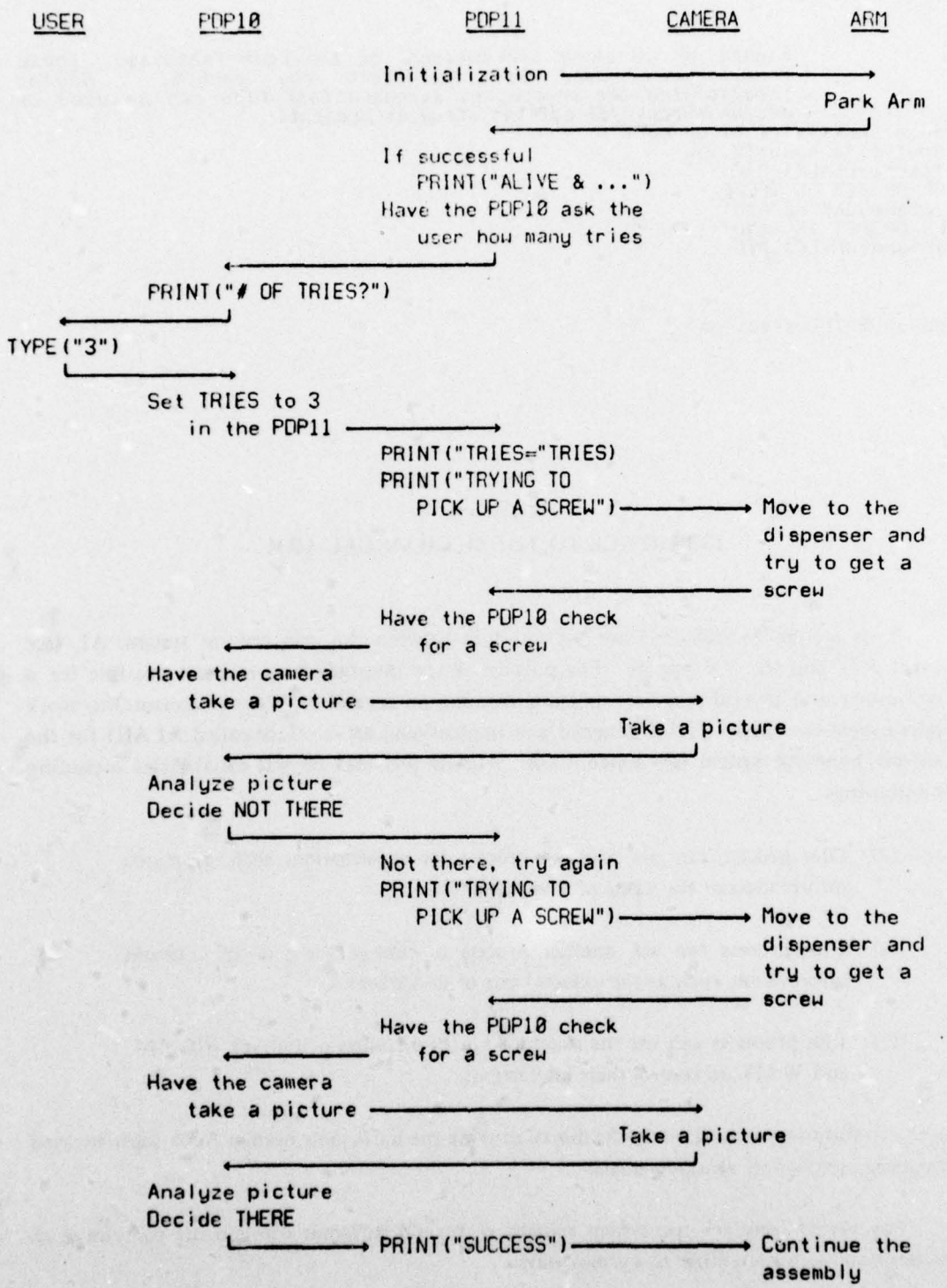

USER
PDP10
PDP11
CAMERA
ARM
Initialization
Park Arm
If successful
PRINT("ALIVE & ...")
Have the PDP10 ask the
user how many tries
PRINT("# OF TRIES?")
TYPE("3")
Set TRIES to 3
in the PDP11
PRINT("TRIES="TRIES)
PRINT("TRYING TO
PICK UP A SCREW")
Move to the
dispenser and
try to get a
screw
Have the PDP10 check
for a screw
Have the camera
take a picture
Take a picture
Analyze picture
Decide NOT THERE
Not there, try again
PRINT("TRYING TO
PICK UP A SCREW")
Move to the
dispenser and
try to get a
screw
Have the PDP10 check
for a screw
Have the camera
take a picture
Take a picture
Analyze picture
Decide THERE
PRINT("SUCCESS")
Continue the
assembly

Figure IV.4.1

(B) the camera and the PDP10 that analyzes the pictures,

and (C) the user.

The basic feedback loop is straightforward: the arm moves to the screw dispenser and tries to pick up a screw; it stops just after leaving the dispenser; the camera takes a picture; the VV program on the PDP10 analyzes the picture, decides whether or not the screw is present, and tells the PDP11 its decision; and the arm either tries again to pick up a screw or moves to the assembly to screw in the screw. However, even for this relatively simple task there is a great deal of communication involved. Consider figure IV.4.1, which shows the flow of control associated with a typical execution of the PICK_UP_A_SCREW_TASK.

The rest of this section is a listing of the AL program that performs the PICK_UP_A_SCREW_TASK, using VV to check for the screws.

```
BEGIN {screw checker}
DEFINE pick_up_screw = "MOVE BARM TO check_screw";

FRAME check_screw;          {The place where visual checking will occur}
FRAME use_screw;                          {The place to insert the screw}
ROT r;                {A rotation variable to represent "straight down"}
SCALAR type;  {TYPE holds  the type  of signal  being sent  from the  arm
                    program to the  vision program, eg.   to say "I  AM
                    READY FOR YOU  TO CHECK THE  SCREW" the AL  program
                    puts a "2" in type  and signals SAIL ...  when  the
                    SAIL routine  finishes, it  signals  AL so  the  AL
                    program can continue ...  the codes are:
                          0: not set
                          1: do the initialization (set TRIES, etc.)
                          2: check  for a  screw and  set THERE (1  is
                             true)
                          3: ask  the  user  to  place  the  screw  on
                             manually
                          4: success, a screw is on the end}
SCALAR tries;          {Specifies the max number of tries to get a screw}
SCALAR there;  {Variable set by  the screw checking  routine to  indicate
                    whether the  vision  routine thinks  the  screw  is
                    there or not,  a 1 (actually anything  greater than
                    0) indicates that the screw is there}
EVENT sail; {This is the universal event used to signal the SAIL  routine
                    that something is to be done}
EVENT al; {The universal event  used by the SAIL  routine to tell the  AL
                    program to continue, the SAIL part has been done}
SCALAR k; {A loop variable}
```

```
r ← ROT(YHAT, 180);                              {Set r to "straight down"}

{Define the two frames in terms of r}
check_screw ← FRAME(r, VECTOR(50.04, 46.07, 5.16));
use_screw ← FRAME(r, VECTOR(30.00, 35.00, 8.00));

{Print something to indicate you are alive and park the arm}
PRINT("ALIVE AND GESTICULATING");
MOVE BARM TO BPARK;
  {WITH DURATION ≥ 2.0*SEC}

type ← 1;       {Ask the SAIL routine to initialize things  ... eg. tries}
SIGNAL sail;
   {Expect the SAIL routine to ask the user appropriate
    questions, set the associated variables (using SETVAL), and
    then return control to this AL program}
WAIT al;
PRINT("tries",tries);          {Just to doublecheck that the value was set}


there ← 0;
k ← 1;
WHILE ((k ≤ tries) and (there ≠ 1)) DO
      BEGIN {pick up loop}
      k ← k + 1;
      ASSERT BARM = BPARK;
      PRINT("TRYING TO GET A SCREW");
      pick_up_screw;   {A macro that moves the arm to the dispenser to
                        pick up a screw.  The arm stops just after
                        leaving the screw dispenser.}
      MOVE BARM TO check_screw;

      type ← 2;
      SIGNAL sail;
         {Have the SAIL program take a picture and analyze it to
          decide whether or not the screw is present.  If it is
          there, have the program set the variable THERE to one.}
      WAIT al;
      END; {pick up loop}
ASSERT BARM = check_screw;

{Now see what the results were  ... either the vision routine decided
   that the screw was there and set THERE or the program ran out of
   tries}
{If the screw is still not on the end, pause and ask the user to put it
   on the end}
IF there ≠ 1 THEN
    BEGIN {screw not on yet}
    PRINT("PLEASE LET ME HAVE A SCREW");
    type ← 3;
    SIGNAL sail;
       {Have the SAIL program make sure that the user places a
```

```
        screw on  the end  of the  screwdriver and  gets the  arm
        ready to proceed}
    WAIT a1;
    END;{screw not on yet}

{Use the screw and then continue with the rest of the assembly}
MOVE BARM TO use_screw;
PRINT("SUCCESS!");
END;{screw checker}
```

Since only one or two correlation operators are required to decide whether or not the screw is on the end of the screwdriver, the time to take the picture and the time to send messages back and forth between the processors dominate the VV time, i.e., the time to locate the matches and compute the probabilities. The overall elapsed time between the AL request and the answer is approximately two seconds.

APPENDIX V. HOLE LOCATION EXAMPLE

This section is a trace of a programmer defining a VV program to locate a screw hole. At programming time the expected precision of the correlation operator is used to determine the resolution required to produce the desired precision. The training stage demonstrates the culling of bad matches. The planning stage ranks the operator/feature pairs according to their average structural consistency. Operators that find their features at locations that are consistent with the other matches are given good ratings. The execution stage demonstrates the use of previous matches to reduce the tolerance region about a new feature to be found.

PROGRAMMING TIME

```
Do you want to use the camera? (Y or N):N
Calibration picture:WCALI.PIC
Display on the synthesizer, halftones, or none? (S,H,or N):H
THE PICTURE'S CHANNEL AND THE OVERLAY CHANNEL:          24          25
Want to use a calibration file? (Y or N):N

            Calibrate the camera.
Size of the interest operator (4,8,...):8
Size of the correlation operator (eg. 9):9
Automatic, Manual, or Standard Calibration? (A, M, or S):M

            Point out the features in the calibration picture.
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):L
X and Y:+1 +1
X and Y:+1
X and Y:
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):A
The feature's X,Y,Z: 2.0 0 0
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):L
X and Y:+3
X and Y:+1
X and Y:
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):A
The feature's X,Y,Z: 1.5 .5 0
(A,R,L,T,C,F, or S):L
X and Y:+0 +1
X and Y:
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: .5 .5 0
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: 0 0 0
(A,R,L,T,C,F, or S):L
X and Y:-2
X and Y:-1
X and Y:
```

```
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: 0 2.0 0
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: .5 1.5 0
(A,R,L,T,C,F, or S):L
X and Y:+1 -6
X and Y:+1
X and Y:
(A,R,L,T,C,F, or S):A
The feature's X,Y,Z: 2.0 2.0 0
(A,R,L,T,C,F, or S):S
Do you want to point out any additional features? (Y or N):N
```

Figure V.1 shows WCALI.PIC and the set of features defined by the preceding process.

```
Approximate lens center (X,Y,Z):-2.75 -12.5 13
TYPE OF TASK? (0, location; 1, inspection):0
Size of the interest operator (4,8,...):8
Size of the correlation operator (eg. 9):9
Known limits on X, (-dx,+dx):-.6 +.6
Known limits on Y, (-dy,+dy):-.1 +.1
Known limits on α, (-dα,+dα):-15 +15
Desired limits on X, (-dx,+dx):-.05 +.05
Desired limits on Y, (-dy,+dy):-.05 +.05
Desired limits on α, (-dα,+dα):-3 +3
RX:   12.000000     RY:   2.000000
Expected precision of the correlation operator, (eg. 1 pixel):1
Do you want to determine the picture size? (Y or N):Y
```

Determine the location and resolution of the camera: (1) use a sample set of features to determine the tolerance volume of the task, (2) project that volume onto the screen to form the task tolerance region, (3) overlay the task tolerance region on the planning picture on the left (4) scale the region to indicate whether the camera should be zoomed in or out to produce the desired resolution, (5) show the scaled task region on the right side of the display, and (6) zoom the camera appropriately.

```
Planning picture:WPLAN1.PIC
```

Point out a sample set of features (to define the task volume).

```
X and Y for new feature (in pixels) (<cr> to stop):70 200
(A,R,L,T,C,F, or S):L
X and Y:+5 -20
X and Y:+4 -4
X and Y:
(A,R,L,T,C,F, or S):A
Input the feature's height above the plane, ie. Z:0.0
X and Y for new feature (in pixels) (<cr> to stop):110 200
(A,R,L,T,C,F, or S):L
X and Y:+0 +12
X and Y:
(A,R,L,T,C,F, or S):A
Input the feature's height above the plane, ie. Z:-.5
X and Y for new feature (in pixels) (<cr> to stop):200 110
(A,R,L,T,C,F, or S):L
X and Y:-7 +8
X and Y:
(A,R,L,T,C,F, or S):A
Z:.25
X and Y for new feature (in pixels) (<cr> to stop):165 105
(A,R,L,T,C,F, or S):L
X and Y:-5 +12
X and Y:
(A,R,L,T,C,F, or S):A
Z:1.32
X and Y for new feature (in pixels) (<cr> to stop):
Approximate screen coords of the point of interest (x,y):100 200
```

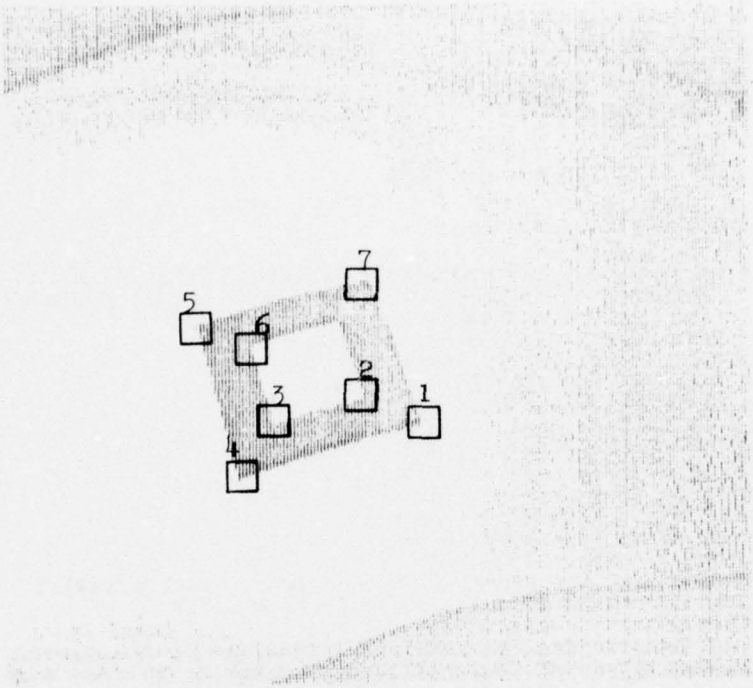


Figure V.1

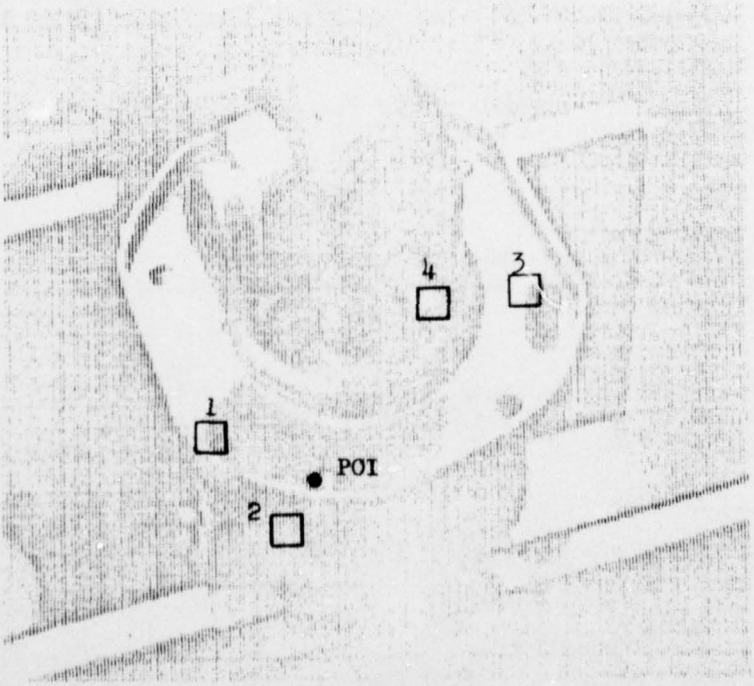


Figure V.2

```
X and Y:+15 -14
X and Y:
The Z of the point of interest, Z:0.0
```

Figure V.2 shows WPLAN1.PIC and the sample set of features used to define the tolerance volume for the task.

Determine the change in resolution.

```
RESOLUTION OF ONE PIXEL ALONG THE SCREEN'S X AXIS:
        (PLANE'S X:     .028701 PLANE'S Y:   -.010617)
RESOLUTION OF ONE PIXEL ALONG THE SCREEN'S Y AXIS:
        (PLANE'S X:   -.010865 PLANE'S Y:   -.035794)
X,Y,Z precision of op:      .029177      .034902      .033167
THE 'ZOOM' FACTOR:      .698042
```

A simple heuristic is used to determine the zoom factor required to achieve the desired resolution along each of the axes in the calibration plane: given the expected precision of one operator and the desired precision about the point of interest, the zoom factor is the ratio of the expected precision over the desired. The zoom factor for the whole picture is the maximum of the two zoom factors along the X-axis and the Y-axis. This overall zoom factor guarantees that both of the desired precisions will be met. It is a conservative estimate of the actual resolution required because it is based upon the expected precision of one operator instead of the expected precision from several operators, which would be greater. As mentioned in section 5.6 it would be possible to estimate the necessary resolution more accurately by performing some example location tasks with pictures at different resolutions.

```
THE ORIGINAL WINDOW IS:
    (MINX,MAXX):   45.928718  244.379760
    (MINY,MAXY):   73.201710  250.164150
THE RESULTING WINDOW IS:
    (MINX,MAXX):   75.542129  214.069290
    (MINY,MAXY):   90.352409  213.879620
```

Figure V.3 shows the task tolerance region overlayed on top of PLAN1.PIC and the scaled tolerance region or the right of the display.

```
R:12.000000    M:3    OPERATOR SIZE: 9
THE MINIMUM AMOUNT OF CORE FOR PICTURES IS:        2844 (36-BIT WORDS)
```

This number is the total amount of core required to hold the pictures used in this VV task. It is based upon one 4-bit test picture and the portions of the planning picture required by ten correlation operators.

```
New, zoomed picture:WPLAN2.PIC
Satisfied? (Y or N):N
New, zoomed picture:WPLAN3.PIC
Satisfied? (Y or N):Y
```

Recalibrate the camera at this new focal length.

```
Automatic or Manual Calibration? (A or M):A
New calibration picture: WCAL2.PIC
```

Figure V.4 shows WCAL1.PIC on the left and the new calibration picture on the right. The calibration object appears to be larger (i.e., at a higher resolution) on the right, but it is not. After the focal length of the camera has been decreased (decreasing the resolution of the picture and the size of the appearance of the calibration object), the portion of the picture outlined on the right in figure V.3 is extracted and expanded so that it fills up the same portion of the display as the original picture. Notice that the precision listed below is LOWER (i.e., less precise) than the precision mentioned earlier.

```
Display the calibration pictures? (Y or N):Y
```

Use the operator/feature pairs defined for the calibration to recalibrate the camera. That is, apply the operators and locate a sufficient number of the features to recompute the

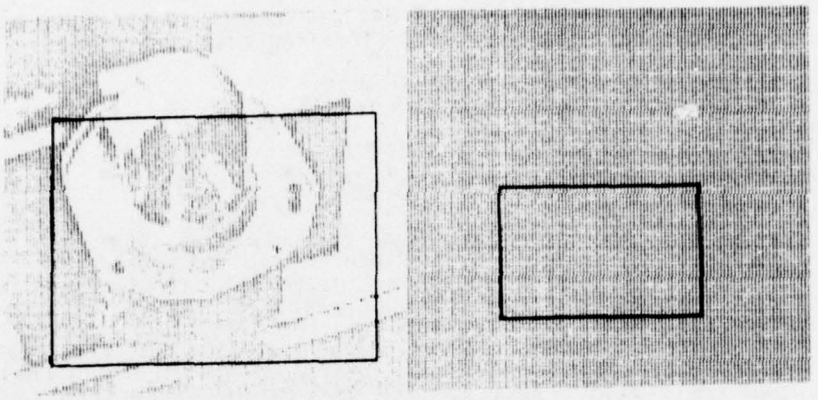


Figure V.3

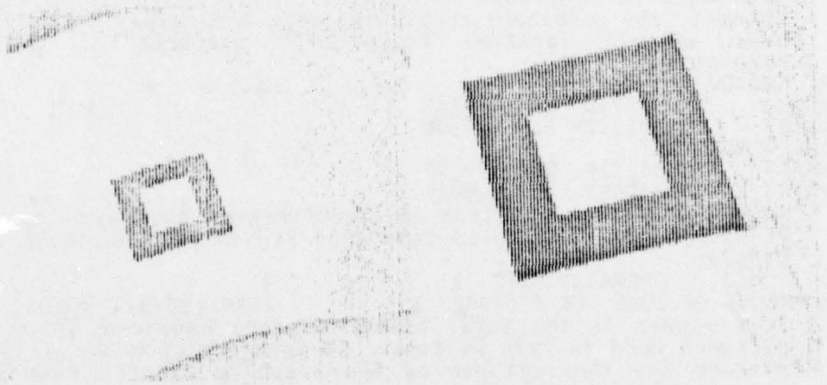


Figure V.4

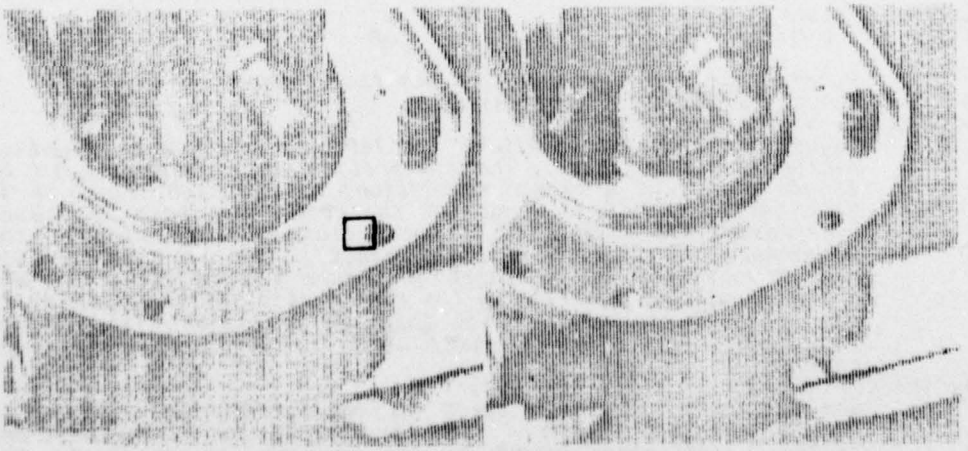


Figure V.5

colineation matrix.

```
X,Y,Z precision of op:  .041813  .050177  .047707
Do you want to save this calibration file? (Y or N):Y
Output file name:V.CAL
COLINEATION MATRIX:
 32.9790210  -5.2200699  86.1000000
 -9.1492472 -21.9564100  178.1000000
   .0039588    .0320665   1.0000000
Use a task file? (Y or N):N
```

Choose the potential operator/feature pairs: (1) apply the interest operator to WPLAN3.PIC and (2) select the promising pairs.

```
Training Picture:WTEST1.PIC
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):L
X and Y:-7
X and Y:-1
X and Y:-0 +1
X and Y:
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):A
Input the feature's height above the plane, ie. Z:0
```

Figure V.5 shows the new planning picture, WPLAN3.PIC, on the left and a test picture on the right. The first feature is shown on the left.

```
Accept,Reject,Locally adjust,Try,Coordinate display,
Find best, or Stop accepting suggestions (A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):L
X and Y:+30 -5
X and Y:+9 -4
X and Y:
(A,R,L,T,C,F, or S):T
VALUE   .884
(A,R,L,T,C,F, or S):A
Input the feature's height above the plane, ie. Z:-.5
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):T
VALUE   .820
(A,R,L,T,C,F, or S):A
Z:1.32
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):A
Z:.38
(A,R,L,T,C,F, or S):L
X and Y:+0 -3
X and Y:+10
X and Y:-10 -10
X and Y:+2
X and Y:+0 +10
X and Y:
(A,R,L,T,C,F, or S):T
VALUE   .740
(A,R,L,T,C,F, or S):A
Z:0.0
(A,R,L,T,C,F, or S):R
(A,R,L,T,C,F, or S):S
Do you want to point out any additional features? (Y or N):Y
X and Y for new feature (in pixels) (<cr> to stop):20 100
(A,R,L,T,C,F, or S):+0 -15
****UNRECOGNIZED ANSWER***
(A,R,L,T,C,F, or S):L
X and Y:+0 -15
X and Y:+0 -4
X and Y:
```

```
(A,R,L,T,C,F, or S):A
Z:0.0
X and Y for new feature (in pixels) (<cr> to stop):160 35
(A,R,L,T,C,F, or S):L
X and Y:-16
X and Y:-7
X and Y:+0 +1
X and Y:
(A,R,L,T,C,F, or S):T
VALUE  .888
(A,R,L,T,C,F, or S):A
Z:.25
X and Y for new feature (in pixels) (<cr> to stop):
Approximate screen coords of the point of interest (x,y):45 100
X and Y:+14 -3
X and Y:+0 -1
X and Y:
The Z of the point of interest, Z:0
```

This completes the programming time. The camera has been positioned and calibrated, the task has been stated, and the potential operator/feature pairs have been defined. Figure V.6 shows the features that have been chosen.

TRAINING TIME

Apply the operators to several training pictures and gather statistics on the structural consistency of the matches. All of the operators are applied to each training picture. Their matches are handed to the least-squares culling routine, which determines the best fit and culls out the matches with large residual errors. The culling is done in two steps: (1) permanently cull any match that is significantly inconsistent (i.e., more the three standard deviations away from its expected position), and (2) temporarily eliminate the worst remaining match and try another fit; if the fit is significantly better than the one including the match, permanently cull the questionable match; if the fit is not significantly better, leave the questionable match in the set of good matches and stop trying to cull matches.

```
Training Picture:WIESI1.PIC
Do you want to include the results of this trial? (Y or N):Y
THE WORST:          6     .233361   ...   ABSOLUTE
```

The match for operator 6 is so far away from the position predicted by the best fit that it is immediately culled from the set of matches for this training picture. Figure V.7 shows the match for operator 6. Notice that the correct match for operator 6 is right on the edge of the picture. For some reason the operator did not find it.

```
THE WORST:          5     .015230   ...   TENTATIVE
OLD AND NEW TOLERANCES ON α,dx,dy:
    .211152       .203766
    .003905       .003862
    .005012       .005281
TOLERANCES DID not IMPROVE BY DELETING THAT POINT
NO IMPROVEMENT
```

The match for operator 5 was the worst remaining match, but taking it out of the set did not significantly improve the fit, so it was left in the set.

#	PLANNED POSITION		BEST ESTIMATE		MEASURED	
1	2.014514	.096158	1.639305	.100788	1.628164	.094524
2	.006031	-.293669	-.379378	-.232212	-.380342	-.240852
3	1.340150	-.083038	.960153	-.059308	.961516	-.049120
4	.465959	.625199	.106293	.673312	.103857	.686125
5	.281509	-.085656	-.098140	-.032055	-.083456	-.036095
6	-.645620	.466445	-1.009322	.545985	-.737630	.686142
7	2.545940	.842806	2.191587	.832145	2.190081	.828087

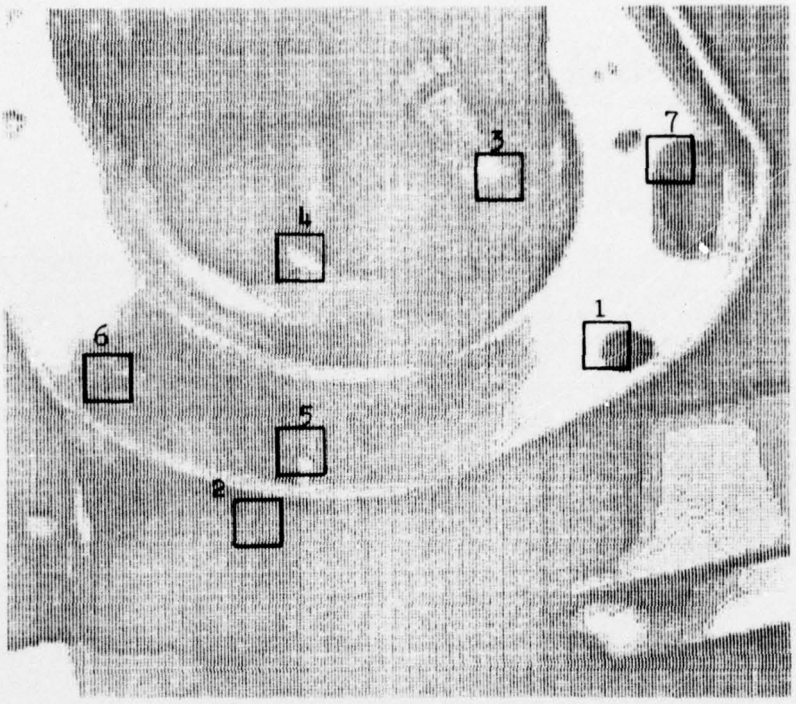


Figure V.6

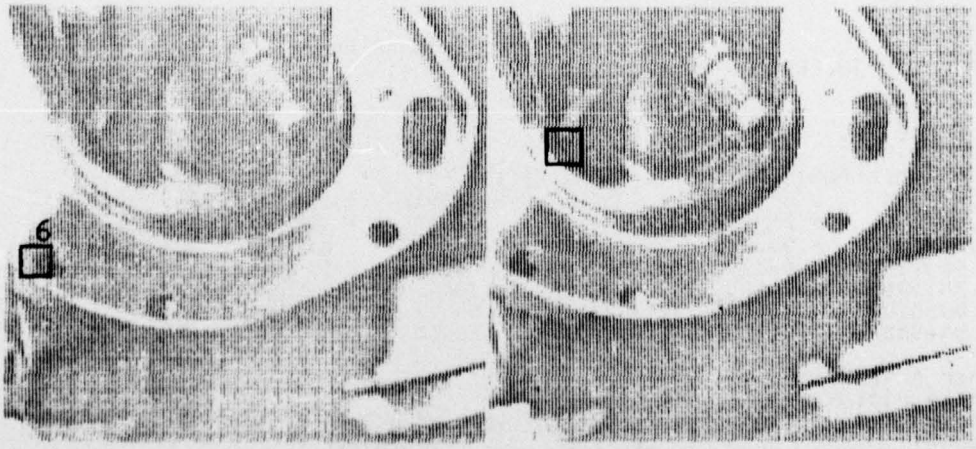


Figure V.7

```
Another trial? (Y or N):Y
Training Picture:WIEST2.PIC
Do you want to include the results of this trial? (Y or N):Y
THE WORST:              6     .352876   ...   TENTATIVE
OLD AND NEW TOLERANCES ON α,dx,dy:
   2.619640        .369691
    .053704        .006961
    .059813        .008677
IMPROVEMENT
```

This time the match for operator 6 was not far enough away from the position predicted by the fit to be immediately culled. However, it was the worst match and leaving it out significantly improved the fit, so it was removed from the set of consistent matches.

```
THE WORST:              7     .028848   ...   TENTATIVE
OLD AND NEW TOLERANCES ON α,dx,dy:
    .369691        .439893
    .006961        .006392
    .008677        .007696
NO IMPROVEMENT
#     PLANNED POSITION          BEST ESTIMATE          MEASURED
1  2.014514     .096158   1.735334     .193656   1.735452     .184310
2   .006031    -.293669   -.253491    -.286197   -.242347    -.293800
3  1.340150    -.083038   1.069752    -.015818   1.071208    -.015020
4   .465959     .625199    .164462     .662208    .153790     .675135
5   .281509    -.085656    .012310    -.066253    .036402    -.061121
6  -.645620     .466445   -.938811     .443405   -.715465     .976305
7  2.545940     .842806   2.232491     .963547   2.206353     .951339

Another trial? (Y or N):Y
Training Picture:WIEST3.PIC
Do you want to include the results of this trial? (Y or N):Y
THE WORST:              2     .286108   ...   TENTATIVE
OLD AND NEW TOLERANCES ON α,dx,dy:
   2.294197       1.584936
    .045913        .032051
    .053366        .039683
IMPROVEMENT
THE WORST:              6     .102611   ...   TENTATIVE
OLD AND NEW TOLERANCES ON α,dx,dy:
   1.584936       1.002098
    .032051        .016835
    .039683        .026401
IMPROVEMENT
THE WORST:              3     .066842   ...   TENTATIVE
OLD AND NEW TOLERANCES ON α,dx,dy:
   1.002098        .783424
    .016835        .014625
    .026401        .021650
NO IMPROVEMENT
#     PLANNED POSITION          BEST ESTIMATE          MEASURED
1  2.014514     .096158   1.333192    -.044914   1.300780    -.030977
2   .006031    -.293669   -.711628     .023521   -.403734    -.109135
3  1.340150    -.083038    .635842    -.069025    .673031    -.013483
4   .465959     .625199   -.058160     .816510   -.034034     .776823
5   .281509    -.085656   -.396662     .164779   -.395031     .163998
6  -.645620     .466445  -1.177125     .909937   -.964250    1.090785
7  2.545940     .842806   2.017903     .564240   1.981370     .535230

Another trial? (Y or N):Y
Training Picture:WIEST4.PIC
Do you want to include the results of this trial? (Y or N):Y
THE WORST:              5     .019895   ...   TENTATIVE
OLD AND NEW TOLERANCES ON α,dx,dy:
    .263823        .263642
    .005411        .005717
```

```
     .006022        .006472
TOLERANCES DID not IMPROVE BY DELETING THAT POINT
NO IMPROVEMENT
#     PLANNED POSITION          BEST ESTIMATE          MEASURED
1   2.014514     .096158   2.160502     .084181   2.167497     .091572
2    .006031    -.293669    .135249    -.206199    .123865    -.216194
3   1.340150    -.083038   1.478126    -.061565   1.461673    -.044214
4    .465959     .625199    .639897     .688888    .654979     .685162
5    .281509    -.085656    .420643    -.012014    .402623    -.003581
6   -.645620     .466445   -.478155     .585102   -.471116     .571449
7   2.545940     .842806   2.728075     .803736   2.744815     .797935

Another trial? (Y or N):Y
Training Picture:WIEST5.PIC
Do you want to include the results of this trial? (Y or N):Y
THE WORST:              1      .037772    ...    TENTATIVE
OLD AND NEW TOLERANCES ON α,dx,dy:
     .404727        .400978
     .008494        .008277
     .009060        .008279
NO IMPROVEMENT
#     PLANNED POSITION          BEST ESTIMATE          MEASURED
1   2.014514     .096158   2.366920     .239606   2.404679     .240593
2    .006031    -.293669    .403978    -.337308    .373446    -.317529
3   1.340150    -.083038   1.712388    -.002191   1.698851    -.008762
4    .465959     .625199    .775488     .620728    .746931     .634586
5    .281509    -.085656    .658681    -.104319    .675836    -.114726
6   -.645620     .466445   -.316244     .358179   -.309320     .338962
7   2.545940     .842806   2.825802    1.032906   2.836590    1.034477

Another trial? (Y or N):N
FINAL STATISTICS
#        N            MEAN          SD
1   5.000000      .019988      .012993
2   5.000000      .081760      .142108
3   5.000000      .023548      .025505
4   5.000000      .026411      .013491
5   5.000000      .016326      .008771
6   5.000000      .239727      .233803
7   5.000000      .021689      .016642
```

Notice that the statistics for each operator include the results from each trial (i.e., N=5) even though some of the matches were culled from the best fit. The worst matches were culled so the best possible fit could be determined. Then the residuals for all of the matches were computed and added to the statistical information. In this way, the operators that match the wrong features (and hence are far away from their predicted positions) produce large values for their average residual error.

PLANNING TIME

The ranking scheme simply orders the operator/feature pairs according to their average residual errors.

```
THE ORDERED SET OF FEATURES ... # and the ranking value
5   .016326
1   .019988
7   .021689
3   .023548
4   .026411
2   .081760
6   .239727
```

The system renumbers the features according to this ranking. Thus, from now on the old number five will be number one, etc.

```
Do you want to save this task? (Y or N):Y
Output file name:V.TSK
```

EXECUTION TIME

Three example execution-time performances are presented below. In the first one the first three operators locate the desired hole sufficiently precisely; in the second, four operators are needed; and in the third, the operators can not produce the desired precision.

```
Picture:WLXEC1.PIC
Do you want to see each feature's match as it is found? (Y or N):Y
*** for debugging ... set hyp[N] to V, (N & V):
```

HYP[...] holds the known and desired tolerances for the task. This statement provides a way for the user to temporarily change the these values in order to see how well the operator/feature pairs behave in different situations. For this particular execution of the task no changes were made.

```
How many features should the location routine start with: 3
<cr> to continue:
```

The system shows all of the features on the planning picture and asks for a carriage return to indicate that the user has seen them and is ready to apply one operator at a time. The system also shows the match for each operator and pauses again if the user specified that he wanted to see each feature's match.

```
VALUE  .856
<cr> to continue:
VALUE  .954
<cr> to continue:
VALUE  .921
<cr> to continue:
```

Figure V.8 shows the planning picture on the left and the execution-time picture on the right. The three features and the corresponding matches are outlined.

```
THE WORST:          2      .014804    ...   TENTATIVE
TWO LEFT, WORST IS NOT TOO BAD
FEATURE NUMBER            1 WAS USED
FEATURE NUMBER            2 WAS USED
FEATURE NUMBER            3 WAS USED
         3 OF THE INITIAL           3 WERE USED
X & Y TOL:     .015677     .019586
SUCCESS ... LOCATE WAS ABLE TO REACH THE TOLERANCES
         3 FEATURES WERE TRIED
         3 MATCHES WERE USED
THE CHANGE:
DX & 2*sd(DX):     -.030304      .015677
DY & 2*sd(DY):      .089459      .019586
Da & 2*sd(Da):    -3.717137    1.018084


Another picture? (Y or N):Y
Picture:WLXEC2.PIC
Do you want to see each feature's match as it is found? (Y or N):Y
*** for debugging ... set hyp[N] to V, (N & V):
How many features should the location routine start with: 3
<cr> to continue:
VALUE  .815
<cr> to continue:
VALUE  .778
<cr> to continue:
VALUE  .829
<cr> to continue:
THE WORST:          2      .033371    ...   TENTATIVE
TWO LEFT, WORST IS NOT TOO BAD
FEATURE NUMBER            1 WAS USED
FEATURE NUMBER            2 WAS USED
FEATURE NUMBER            3 WAS USED
```

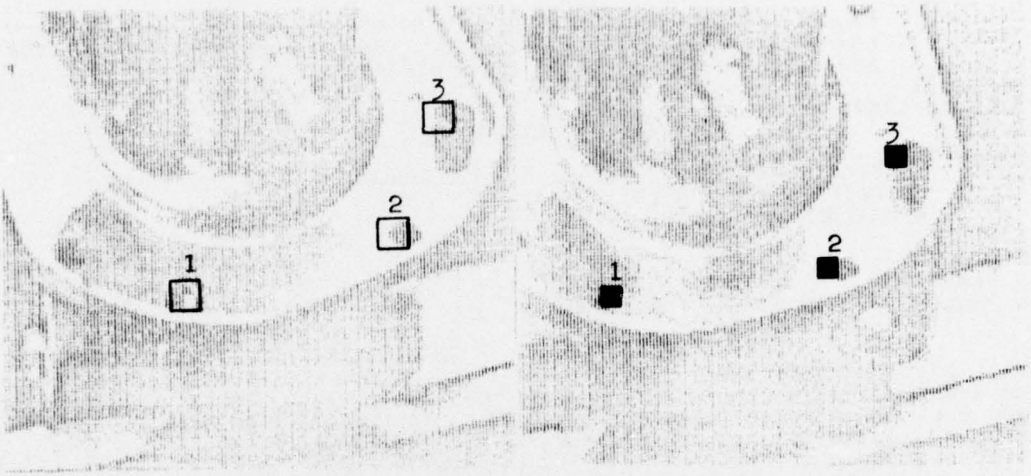


Figure V.8

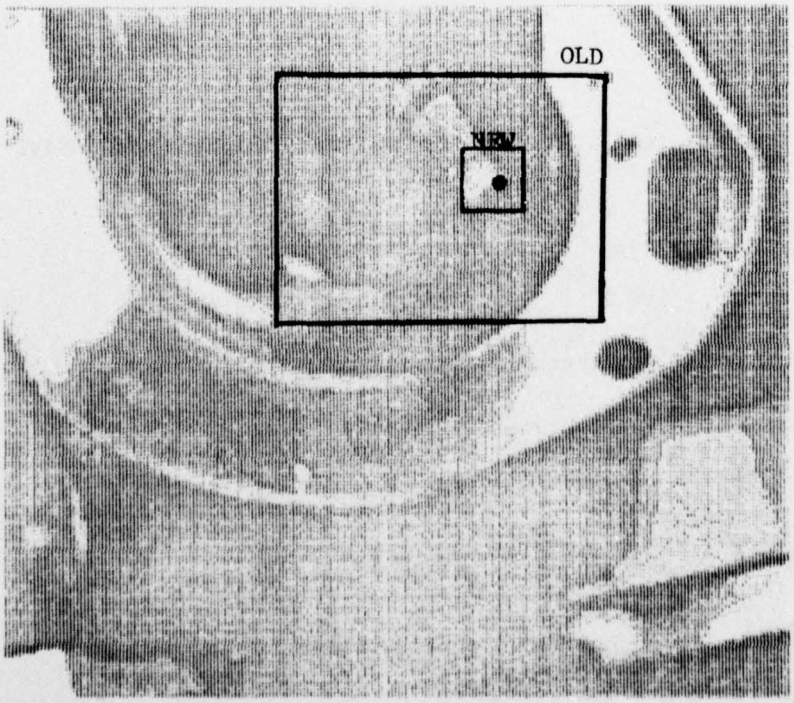


Figure V.9

```
          3 OF THE INITIAL              3 WERE USED
X & Y TOL:     .055957      .047569
NEW,R,&3*R ... FOR X AND Y
   2.016724      .055627      .106881
    .561883      .050647      .151941
THE TOLERANCE BOX ON THE SCREEN (in pixels):
        115          127           43           54
```

Figure V.9 shows the original tolerance region and the new tolerance region for operator 4. Notice the reduction in the size of the area that has to be searched.

```
VALUE  .921
<cr> to continue:
FEATURE            4 HAS BEEN ADDED PERMANENTLY
X & Y TOL:     .029249      .043301
SUCCESS ... ILOCATE WAS ABLE TO REACH THE TOLERANCES
          4 FEATURES WERE TRIED
          4 MATCHES WERE USED
THE CHANGE:
DX & 2*sd(DX):     -.694948      .029249
DY & 2*sd(DY):      .254547      .043301
Do & 2*sd(Do):   -13.057337     1.566848


Another picture? (Y or N):Y
Picture:WEXEC2.PIC
Do you want to see each feature's match as it is found? (Y or N):Y
*** for debugging ... set hyp[N] to V, (N & V):33 +.005
*** for debugging ... set hyp[N] to V, (N & V):36 +.005
*** for debugging ... set hyp[N] to V, (N & V):
```

The same picture is used for this execution as in the last example, but the desired precision has been increased from .05 to .005.

```
How many features should the location routine start with: 3
<cr> to continue:
VALUE  .815
<cr> to continue:
VALUE  .778
<cr> to continue:
VALUE  .829

<cr> to continue:THE WORST:           2      .033371    ...    TENTATIVE
TWO LEFT, WORST IS NOT TOO BAD
FEATURE NUMBER            1 WAS USED
FEATURE NUMBER            2 WAS USED
FEATURE NUMBER            3 WAS USED
          3 OF THE INITIAL              3 WERE USED
X & Y TOL:     .035957      .047569
NEW,R,&3*R ... FOR X AND Y
   2.016724      .035627      .106881
    .561883      .050647      .151941
THE TOLERANCE BOX ON THE SCREEN (in pixels):
        115          127           43           54
VALUE  .921
<cr> to continue:
FEATURE            4 HAS BEEN ADDED PERMANENTLY
X & Y TOL:     .029249      .043301
NEW,R,&3*R ... FOR X AND Y
    .625331      .032508      .097524
   -.082668      .031960      .095879
THE TOLERANCE BOX ON THE SCREEN (in pixels):
         69           79           36           44
VALUE  .897
<cr> to continue:
FEATURE            5 HAS BEEN ADDED PERMANENTLY
X & Y TOL:     .033670      .052802
NEW,R,&3*R ... FOR X AND Y
   -.711628      .041029      .123086
```

```
    .023521     .047722     .143165
THE TOLERANCE BOX ON THE SCREEN (in pixels):
        15          26         109         122
VALUE  .740
<cr> to continue:
IT IS TOO FAR OFF TO EVEN TRY IN THE FIT
FEATURE 6 DID NOT FIT IN
```

The program found a match for operator 6, but a prescreening showed that if it were included in the fit, it would not improve the precision because it was too far away from its predicted position.

```
X & Y TOL: .034  .053
NEW,R,&3*R ... FOR X AND Y
-).177  .042  .127
 .910  .052  .157
THE TOLERANCE BOX ON THE SCREEN (in pixels):
-9 2 63 75
VALUE  .744
<cr> to continue:
IT IS TOO FAR OFF TO EVEN TRY IN THE FIT
FEATURE 7 DID NOT FIT IN
X & Y TOL: .034  .053
ILOCATE COULD NOT MEET THE DESIRED TOLERANCES
***UNABLE TO DECIDE WHERE THE OBJECT IS***
Another picture? (Y or N):N

End of SAIL execution
```

APPENDIX VI. CURVES AS FEATURES

Curves are important within programmable assembly because several standard fabrication techniques produce smooth, curved surfaces. For example, drilling a hole produces a circle, which appears as an ellipse in a picture.

This appendix describes a new type of operator/feature pair based upon curves. The first section describes the basic rationale behind the use of curves; the second section describes the type of local information that can be used to filter out incorrect matches; the third section describes techniques to specify curves within a VV system; and the fourth section discusses the representation of curves and evaluates parameterized polynomials in some detail.

An operator/feature pair based upon linked, parameterized polynomials has been implemented and has been used within the VV system. Given an initial estimate for the position of a curve, the system can automatically refine the model of the curve and characterize its usefulness as an operator/feature pair.

Section 1
BASIC APPROACH

One of the basic approaches within VV is to locate large features that are easy to find in order to help locate small features that are harder to find. For example, when visually servoing a screw into a hole, a VV program may initially locate a point on the shaft of the screwdriver and use that information to help locate the tip of the screw. Consider figure VI.1.1. Figure VI.1.1a shows the screw on the end of the screwdriver as it is approaching the hole. The overlay indicates the planned position for the tip of the screw and the initial tolerance region about that point. Figure VI.1.1b shows a search pattern and a point on the shaft found by an edge operator. Figure VI.1.1c shows the new tolerance region about the tip of the screw implied by the information gained from the match of the edge operator.

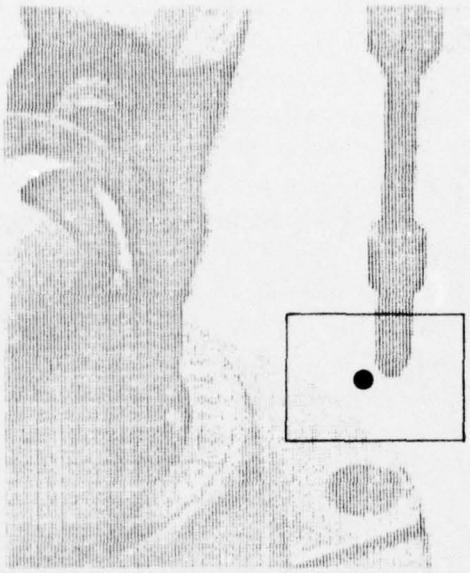


Figure VI.1.1.a

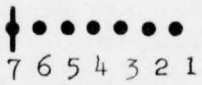


Figure VI.1.1.b

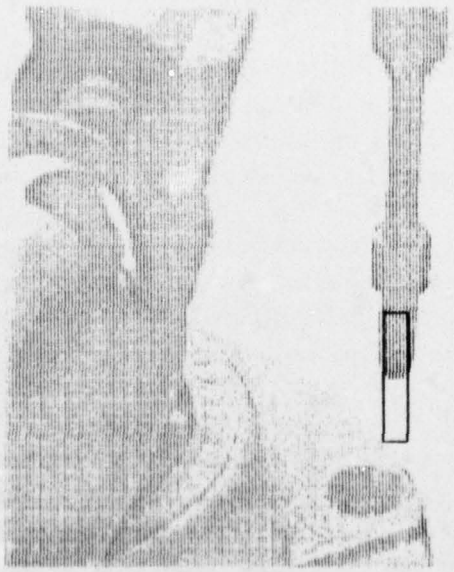


Figure VI.1.1.c

Curves can be used in essentially the same way. Having located a point on a curve, one can reduce the tolerance region about a feature of interest. For example, consider figure VI.1.2, which shows a brake assembly lying on a table. The goal of the task is to locate the bolt hole that is in the lower right-hand corner. Figure VI.1.2.a shows the initial tolerance region about that hole. Given a picture of the brake assembly at some unknown location, the edge operator can be applied in a search pattern to try to locate a point on the large curve. Figure VI.1.2.b shows such a search pattern and a matching point that locally appears to be a point on the large curve. If that is a point on the large curve, where would the bolt hole be? Figure VI.1.2.c shows the curve positioned so that (1) the curve passes through the matching point and (2) the slope of the curve matches the slope of the matching point. If the edge operator produces exact values for the position of the matching point and the slope of the curve, and if there is no angular uncertainty associated with the location of the brake assembly, the bolt hole would be at the position shown in figure VI.1.2.c.

Edge operators are not precise and there is usually some uncertainty about the orientation of the part. These uncertainties can be combined to form a new tolerance region about the hole. For example, if the orientation of the brake assembly is only known within plus or minus fifteen degrees, the bolt hole will lie on the arc of the curve shown in figure VI.1.2.d. If the estimate of the slope produced by the edge operator has an uncertainty of five degrees, the curve may be located at any position within the range shown in figure VI.1.2.e. And finally, if there is an uncertainty of one pixel about the position of the matching point, the implied tolerance region would be expanded slightly. Figure VI.1.2.f shows the final tolerance region about the hole produced by all of these uncertainties. Notice that the new tolerance region is considerably smaller than the original one. This process, which incrementally builds the new tolerance region from the different uncertainties, works the best when the uncertainties are mainly two-dimensional.

This technique is only one of three possible techniques to incorporate the information gained from a point on a curve: (a) direct incorporation into the least-squares fitting routine, (b) two-dimensional modelling of the uncertainties, and (c) statistical modelling from several training pictures. The three approaches cover a range from the analytic to the experimental.

The position and orientation information associated with a point on a simple curve, such as a circle, can be directly incorporated into the least-squares fitting routine. More complicated curves are more difficult because it is harder to specify the equations that relate the location of the curve to the position and orientation information produced by the edge operator.

If the curve is complex and the uncertainties change the two-dimensional appearance of the curve, it is still possible to extract a certain amount of information from a point on the curve by modelling the two-dimensional distribution of implied positions for the point of interest. That is, during the training stage, locate a point on the curve, determine the implied

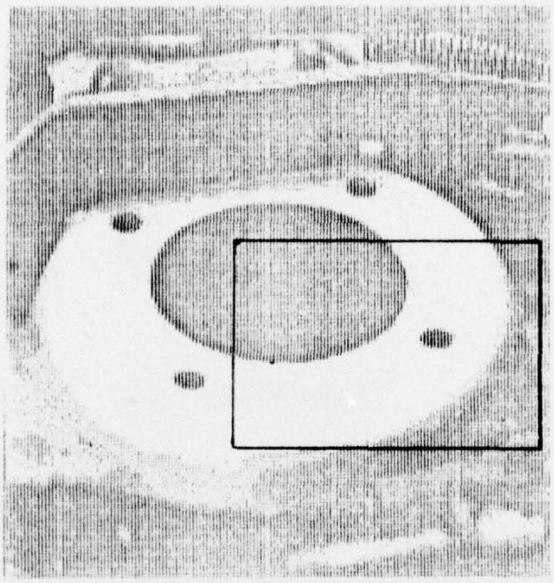


Figure VI.1.2.a

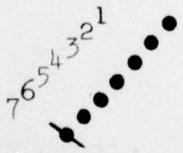


Figure VI.1.2.b

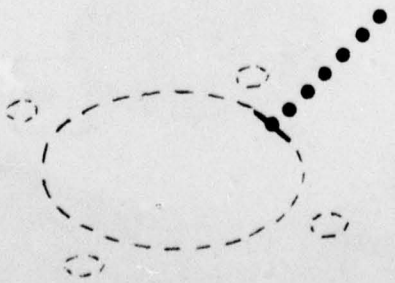


Figure VI.1.2.c

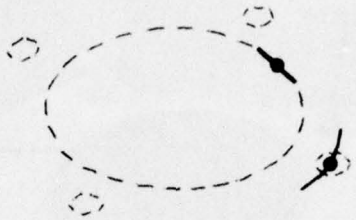


Figure VI.1.2.d

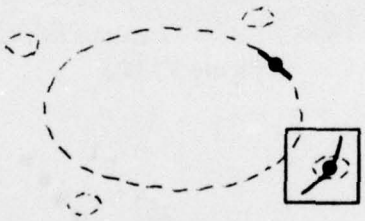


Figure VI.1.2.e

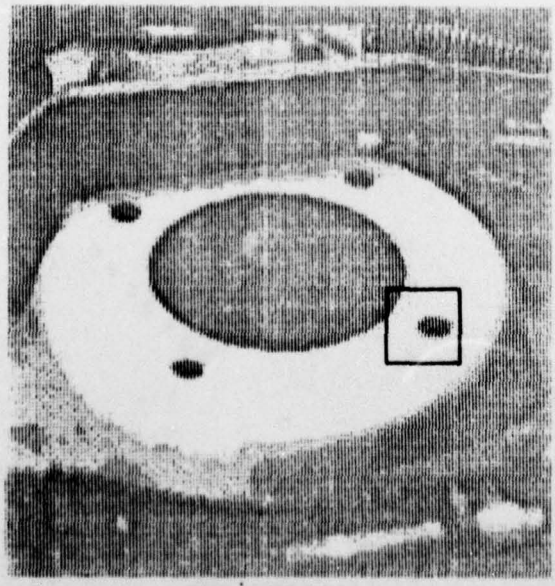


Figure VI.1.2.f

position for the point of interest, locate the actual position of the point of interest, and gather statistics on the errors between the implied and actual positions. At execution time these statistics can be used to produce a tolerance region about the implied position for the point of interest.

Section 2
LOCAL TESTS FOR A POINT ON A CURVE

Given an analytic curve and a match of an edge operator, is the matching edge a point on the curve? Put another way, what tests can be applied to the local information in order to filter out incorrect matches? Essentially the same types of tests can be applied to decide if an edge is on a curve as to decide if an edge is on a line. The tests are based upon five types of information:

(1) contrast,
(2) distinctness,
(3) slope,
(4) continuity,
and (5) curvature.

When an edge operator is used to locate a point on a line, its parameters are set to locate potential edges that have a certain contrast and distinctness. Having found a candidate that meets these qualifications, the candidate is checked to make sure that it is within the correct range of angles. For example, if the line is vertical, the candidate edge is horizontal, and the angular uncertainty is only twenty degrees, the candidate edge can not be on the line. If the candidate edge is within the correct range of angles, its local continuity can be checked. To perform this test the edge operator is applied on either side of the candidate edge. If there are no edges, the candidate is discarded. If the new edges are at inconsistent angles or positions, the candidate edge is rejected because its local curvature is incorrect.

The generalization from lines to curves is straightforward. The edge operator is tuned to locate candidate edges that have a certain contrast and distinctness. The range of acceptable angles includes the complete set of slopes along the curve and is augmented by the angular uncertainty associated with the part. For example, if the slope of the curve varies from 130 degrees to 270 degrees, and if there is a ten-degree uncertainty associated with the object upon which the curve is based, the total range of the acceptable angles is 120 degrees to 280 degrees. The continuity test is the same for curves as it is for lines. The curvature test

checks the angles and positions of the two or three edge points used in the continuity test. If they are not consistent with the curvature of the analytic curve, the candidate edge is rejected. For example, consider the brake assembly example discussed in the previous section. If the edge operator locates a candidate edge with the correct contrast, distinctness, and angle, and if the edge operator is applied once on either side of the candidate to check the continuity and curvature, what do the results shown in figure V1.2.1 imply? They appear to imply that the curvature at the original candidate edge is greater than the curvature of the large curve. Thus, the edge is probably on the small curve formed by the bolt hole, not the large curve.

Figure V1.2.1

This type of checking is referred to as *local* checking because the decisions are based upon a small portion of the picture. A curve may be two hundred pixels long, but the decision whether or not the candidate edge is on the curve is based upon the intensities in a small area, such as a 10x10 array of pixels. *Global* consistency checking is performed by the least-squares culling routine.

Section 3
SPECIFICATION OF A CURVE

In the current implementation of the VV system a programmer can specify a curve feature by pointing out a few points near the curve in a planning picture and letting the system automatically locate and characterize the curve that actually appears in the picture. This interactive process involves the following steps:

(1) Use a cursor to indicate a few points near the curve. Possibly include

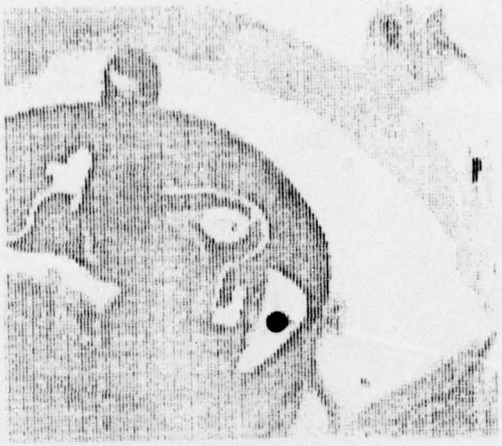


Figure VI.3.1.a

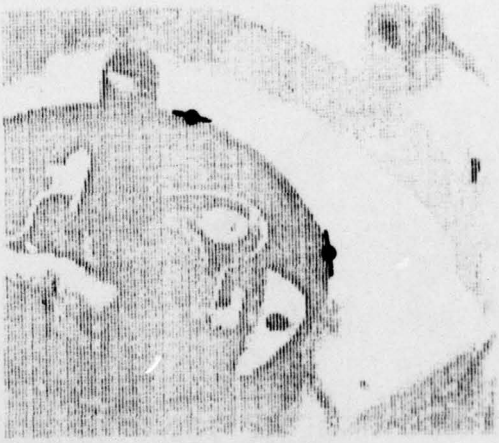


Figure VI.3.1.b

Figure VI.3.1.c

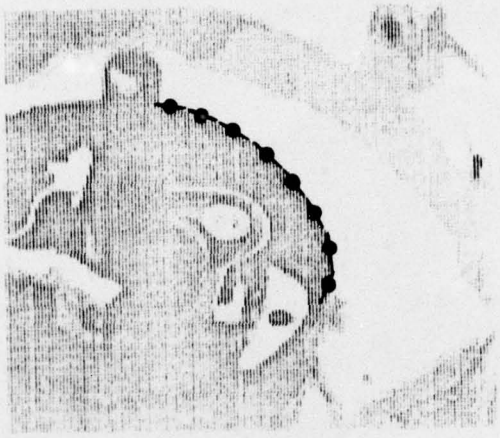


Figure VI.3.1.d

Figure VI.3.1.e

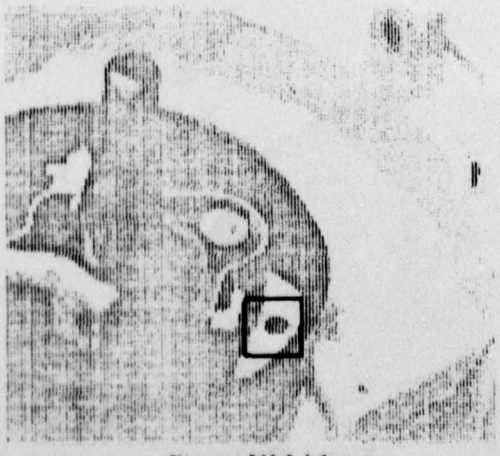


Figure VI.3.1.f

the approximate slopes at the indicated points.

(2) Have the system use an edge operator to search for the closest edge in the picture to each indicated point.

(3) Fit an analytic curve through the actual edge points.

(4) Have the system follow the curve as far as it can in order to pick up as much as possible of the actual curve in the picture.

(5) Have the system walk along the complete curve and gather statistics on (a) the distance between each actual edge point and the closest point on the analytic curve and (b) the distance between each actual edge point and the point on the analytic curve that has the same slope.

(6) If the statistics indicate that the analytic curve does not closely approximate the actual curve, refine the analytic curve by incorporating more of the actual edge points.

The automatic refinement and characterization of a curve mentioned in steps 5 and 6 make it possible for the user to specify the desired range of residual errors in advance. The system stops refining the curve as soon as the predicted residual errors are sufficiently small. The residual errors are important because they determine the size of the tolerance region implied about another feature, given one point on the curve.

Consider figure VI.3.1, which shows a partially completed chainsaw engine. The goal is to locate the dark screw hole that is marked in figure VI.3.1.a. The user indicates the two points and associated slopes shown in figure VI.3.1.b. The system locates the actual edge points in the picture and fits an analytic curve through the points (as shown in VI.3.1.c). It then follows the curve until it loses the curve in the shadow at the lower right end and until it finds a dramatic change in curvature at the upper left corner (see figure VI.3.1.d). Then the system walks along the complete curve gathering statistics. It decides that the middle of the curve needs to be better approximated by the analytic curve, so it adds another point to the analytic curve fitting routine. The refined curve is shown in figure VI.3.1.e. Given the new curve, the potential tolerance region about the screw hole implied by one point on the curve is shown in figure VI.3.1.f.

The only reason that this technique is referred to as an *interactive* method instead of an *automatic* method is that the user has to point out a few points near the curve. If that subtask could be done automatically, the whole suggestion process could be done automatically. How can the system automatically suggest potentially useful curves? If the system had a three-dimensional model of the objects in the scene, it could produce a synthetic

picture (possibly just a line drawing), which it could analyze for long, distinct curves. For example, consider figure VI.3.2, which shows a hidden-line drawing of the expected scene when the chainsaw engine is at its planning location.

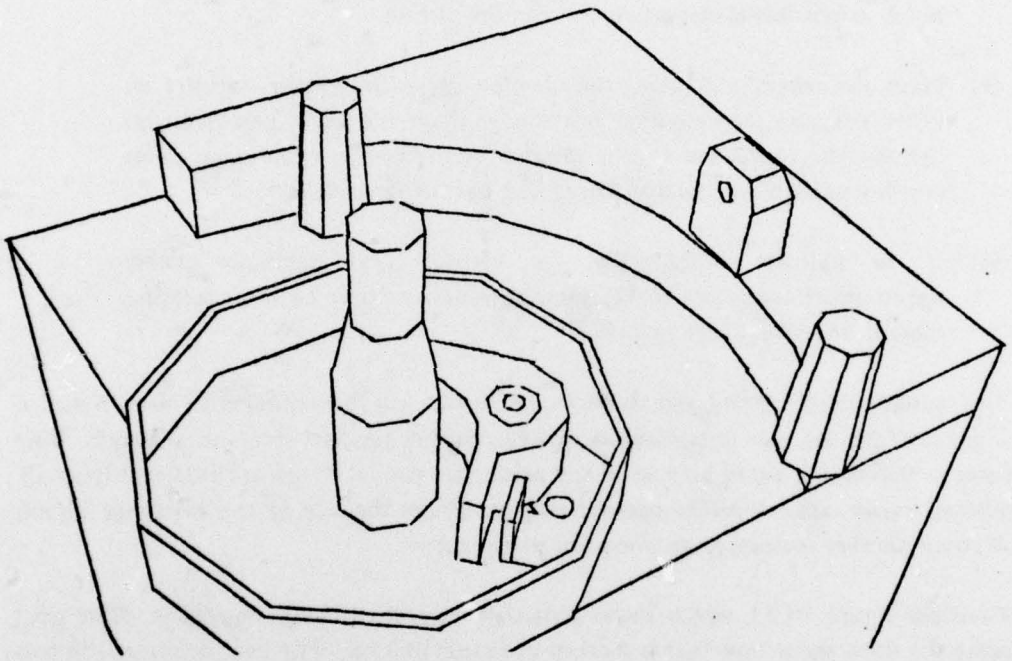


Figure VI.3.2

The current system is a first step toward such an automatic feature suggestion system. The user interactively defines the three-dimensional models of the objects in the scene. The system automatically produces the hidden-line drawing in which each curve is approximated by a sequence of line segments. The system maintains an internal description that distinguishes between the line segments that are used for this type of approximation and those that represent actual line segments. The user interactively points out potential curves in the line drawing. The system automatically locates and characterizes the curve that actually appears in the planning picture.

Section 4
REPRESENTATION OF A CURVE

The first three sections of this appendix have described the use of curves assuming that a representation for curves exists that makes it possible to compute the necessary information, such as the length of the curve. This section briefly describes desirable properties of such a representation and discusses possible representations.

Desirable properties of the representation include:

(1) The representation should be able to model a wide variety of curves, such as ellipses and French curves.

(2) There should be a way to locate the point or points on a curve that have a certain slope.

(3) It should be easy to move a curve around, e.g., translate it and rotate it in the plane of the picture.

(4) There should be a way to slide along the curve a certain distance.

(5) It should be easy to change the form of a curve incrementally.

(6) There should be a way to determine the length of a curve.

(7) There should be a way to locate the closest point of approach to a curve from a given point.

There are several possible ways to represent curves. Some satisfy these requirements better than others. Four reasonable possibilities are:

(1) linear segments,
(2) parameterized polynomials,
(3) regular splines,
and (4) B-splines.

Linear segments are simple, but a large number of points are needed to approximate a curve. Parameterized polynomials require fewer points than linear segments, but they are very

sensitive to the slope of the curve when the curve is almost linear. Regular splines provide a nice range of curves, but they are difficult to modify locally. B-splines can be locally modified, but they require a great deal of computation to locate points with matching slopes or the point of closest approach. Thus, all of the alternatives have their advantages and disadvantages.

The representation that has been implemented is a piecewise polynomial. The representation is an ordered list of point-slopes. That is, each entry in the list has a two-dimensional position and a slope. The curve is the simplest curve that passes through the points and has the correct slope at the two points. (As such, it is a simple, parameterized spline through the points.) The representation of the arc between two point-slopes is:

$$X(t) = A*t^2 + B*t + C \qquad (0.0 \le t \le 1.0) \tag{VI.1}$$

$$Y(t) = D*t^2 + E*t + F,$$

which fits a parabola between two point-slopes.

The coefficients, A through F, can be computed from two point-slope pairs, (x0,y0),s0 and (x1,y1),s1. Let S(t) represent the slope of the curve at the position (X(t),Y(t)). Then S(t) can be computed as follows:

$$S(t) = \frac{\frac{d\,Y(t)}{dt}}{\frac{d\,X(t)}{dt}} \tag{VI.2}$$

or

$$S(t) = \frac{2*D*t + E}{2*A*t + B}. \tag{VI.3}$$

Then the six equations that relate the coefficients to the point-slope values are:

(VI.4)

```
x0 = X(0.0)
y0 = Y(0.0)
s0 = S(0.0)
x1 = X(1.0)
y1 = Y(1.0)
s1 = S(1.0)
```

or

(VI.5)

```
x0 = C
y0 = F
s0 = E/B
x1 = A + B + C
y1 = D + E + F
s1 = (2*D+E)/(2*A+B).
```

These six equations can be solved for A through F to produce the following set of computations for the coefficients:

(VI.6)

$$B \leftarrow \frac{2*s1*(x2 - x1) - 2*(y2 - y1)}{(s2 - s1)}$$

```
A ← (x2 - x1) - B
C ← x1
E ← s1*B
D ← (y2 - y1) - E
F ← y1.
```

It is also a straightforward computation to determine the point on a curve segment that has a specific slope. Let s be the slope in question. Then s and the curve parameter, t are related by the following equation:

(VI.7)

$$s = \frac{2*D*t + E}{2*A*t + B}.$$

Solving (VI.7) for t produces

(VI.8)

$$t = \frac{s*B - E}{(2*D - 2*A*s)}.$$

Thus, the point with a slope of s is (X(t),Y(t)), where t is given by (VI.8).

This representation requires only a few point-slopes to represent a complex curve precisely, i.e., so that the position and curvature of the analytic curve closely matches the position and curvature of the actual curve. For example, consider figure VI.4.1. It shows a picture of a curve and the analytic approximation to the curve formed from seven point-slopes (i.e., six arcs). The curve can be easily improved by including an additional point between two of the current points.

The main drawbacks to this representation are that (1) there are some degenerate cases that arise when the slopes are vertical, (2) inflection points have to be interactively pointed out (a parabola can not fit through an inflection point), and (3) the computations required to find the point of closest approach, etc. increase linearly as the number of point-slopes used to approximate the curve increases.

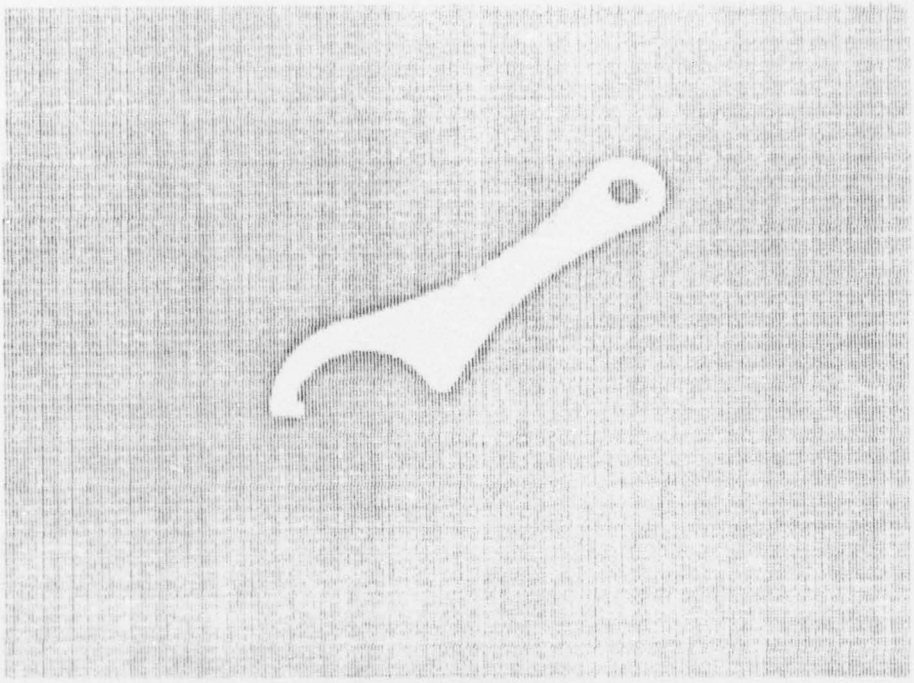


Figure VI.4.1.a

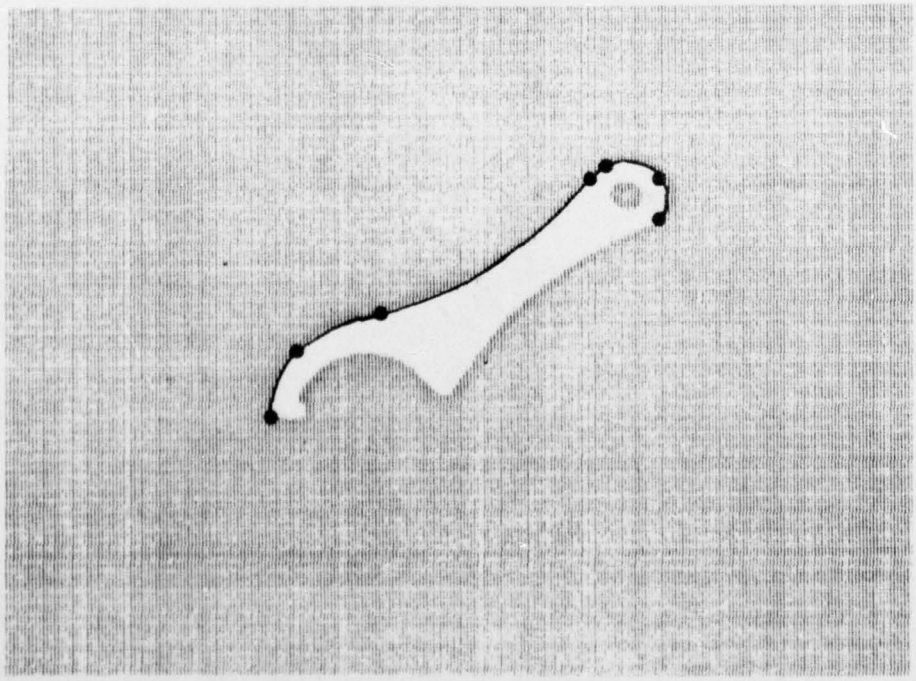


Figure VI.4.1.b