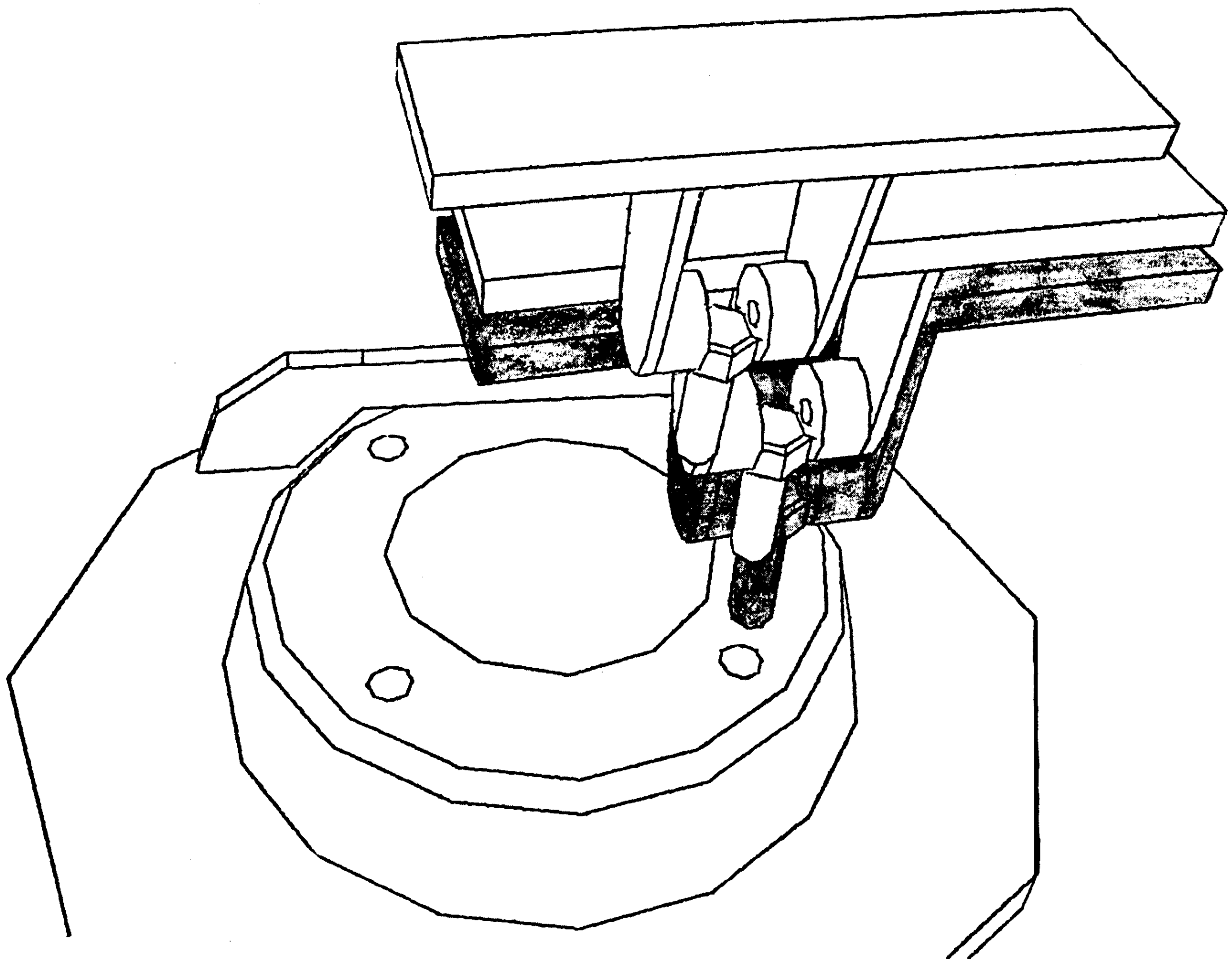
Stanford Artificial Intelligence Laboratory
Memo AIM-275

December 1975

Computer Science Department
Report No. STAN-CS-75-537

VERIFICATION VISION WITHIN A PROGRAMMABLE ASSEMBLY SYSTEM: AN INTRODUCTORY DISCUSSION

by Robert C. Bolles



Research sponsored by

Hertz Foundation
and
Advanced Research Projects Agency
ARPA Order No. 2494

Stanford Artificial Intelligence Laboratory
Memo AIM-275

December 1975

Computer Science Department
Report No. STAN-CS-7 5-53 7

VERIFICATION VISION WITHIN A PROGRAMMABLE ASSEMBLY SYSTEM: AN INTRODUCTORY DISCUSSION

by

Robert C. Bolles

ABSTRACT

This paper defines a class of visual feedback tasks called *Verification Vision* which includes a significant portion of the feedback tasks required within a programmable assembly system, It characterizes a set of general-purpose capabilities which, if implemented, would **provide** a **user** with a system in which to write programs to perform such tasks. Example tasks and **protocols** are used to'motivate these semantic capabilities. Of particular importance are the tools **required** to extract as much information as possible from planning and/or training sessions. Four different levels of, verification systems are discussed. They range from a straightforward. interactive system which could handle a subset of the verification vision tasks, to a **completely** automatic system which could plan its own strategies and handle the total range of verification **tasks**. Several unsolved problems in the **area**.**are** discussed.

*This **research** was supported **by** the Hertz Foundation and the Advanced Research Projects Agency of the Department of **Defense** under Contract DAHC 15-73-C-0435 . **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, **Hertz** Foundation, **ARPA**, or the **U**. S. Government.*

Reproduced in the U.S.A. *Available **from the** National **Technical Information** Service, **Springfield,** Virginia 2215 1.*

TABLE OF CONTENTS

INTRODUCTION

Verification vision, like most visual processing, can be roughly described as the process of using a model of a scene and a set of pictures of the scene to find objects of interest in the scene. The characteristics which distinguish verification vision from the other types of visual processing are: (1) the model states EXACTLY WHICH objects will appear, APPROXIMATELY WHERE they will appear, and APPROXIMATELY HOW they will appear, and (2) the goal is to determine PRECISELY WHERE they appear. A good example of a verification vision task is the task of determining the "exact" location of a pump base which has been placed in a vise. There is no question about what will appear, only some uncertainty about where.

A slightly more general characterization of verification vision includes the case in which the presence of one of the objects may be in question. The model states approximately where and how this object might appear. The goal is to decide if it is present and, if so, to determine precisely where it is. A typical example is the task of deciding whether or not there is a screw--on the end of the screwdriver. The model states what will be in the background, where the screwdriver will probably be, and how the screw will appear, if it is present.

Verification vision has been used in various 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 is supposed to verify that the edge is there and return its position and angle. Notice that the model includes exactly what will appear (the edge), approximately where (at such-and-such a place and within a certain range of angles), and approximately how it will appear (with an approximate contrast of X). There are several systems in which this type of verification vision plays a major role (see [FALK], [SHIRAI], and [TENENBAUM]). Another place where the same idea has been used is in narrow-angle stereo programs. A model in such a system 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 exactly what (the unnamed features which produce the correlation patches), approximately where (near the back-projection of the ray), and approximately how (a slight variation from the correlation patch). See [QUAM 1974], [HANNAH], and [THOMAS] 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, a task such as "insert a screw in a hole" can be reduced to a few subtasks, each of which could involve verification vision:

(1) locate the hole without the screw being in the picture (see figure 1),

(2) move the screwdriver and screw into the picture and locate them against the now. known background (see figure 2),

and (3) decide how to move the screw closer to the hole, move it, stop, locate 'it again, etc.

Notice that there isn't a question of WHAT will appear in the pictures, only WHERE and HOW. Also notice that there are several pieces of information which can be used to give approximate answers to these questions: the expected tolerances on the positions for the objects, the precision of the arm, previous views of similar scenes, etc. This is exactly the type of information that a verification vision system should be able to make use of. The whole purpose behind verification vision is to use this information to determine the cheapest and most reliable way of locating an object to within the desired precision. Thus, cost, efficiency, and confidence considerations play major roles in this discussion. A potential feature is judged by what its expected cost is and what it is expected to contribute toward locating the object.

There have been a few special-purpose programs written which perform verification vision tasks within programmable assembly environments (eg. [BOLLES] and [DEWAR]), but there has been no conscientious attempt to isolate and identify techniques which are generally applicable to such tasks or any effort made to incorporate these capabilities into a system for programming verification vision tasks. The purpose of this report is to do essentially that. It will attempt to define verification vision, motivate the facilities required to accomplish such tasks, outline various levels of verification vision systems, and finally discuss several of the unsolved problems in the area Some of the more theoretical discussions consider verification vision in general, but the examples and suggested systems concentrate on the more restricted environment of programmable assembly.

There are two basic assumptions behind this paper: (1) there is a large class of useful tasks which fit the verification vision paradigm and (2) there are GENERAL techniques which can be used to solve such problems. Programmable assembly provides an almost unlimited number of tasks which are suitable for verification vision. For example, consider the task of assembling a hinge on a doorjamb. A high-level description of the task might be as follows:

(I) Pick up the first hinge piece and align it with the holes on the door jamb.

(2) Pick up the screwdriver and screw in the four screws.

(3) Replace the screwdriver.

(4) Pick up the second hinge piece and align its pin holes with the first piece.

(5) Pick up the pin and insert it to complete the assembly.

Possible verification vision sub-tasks for this assembly include:

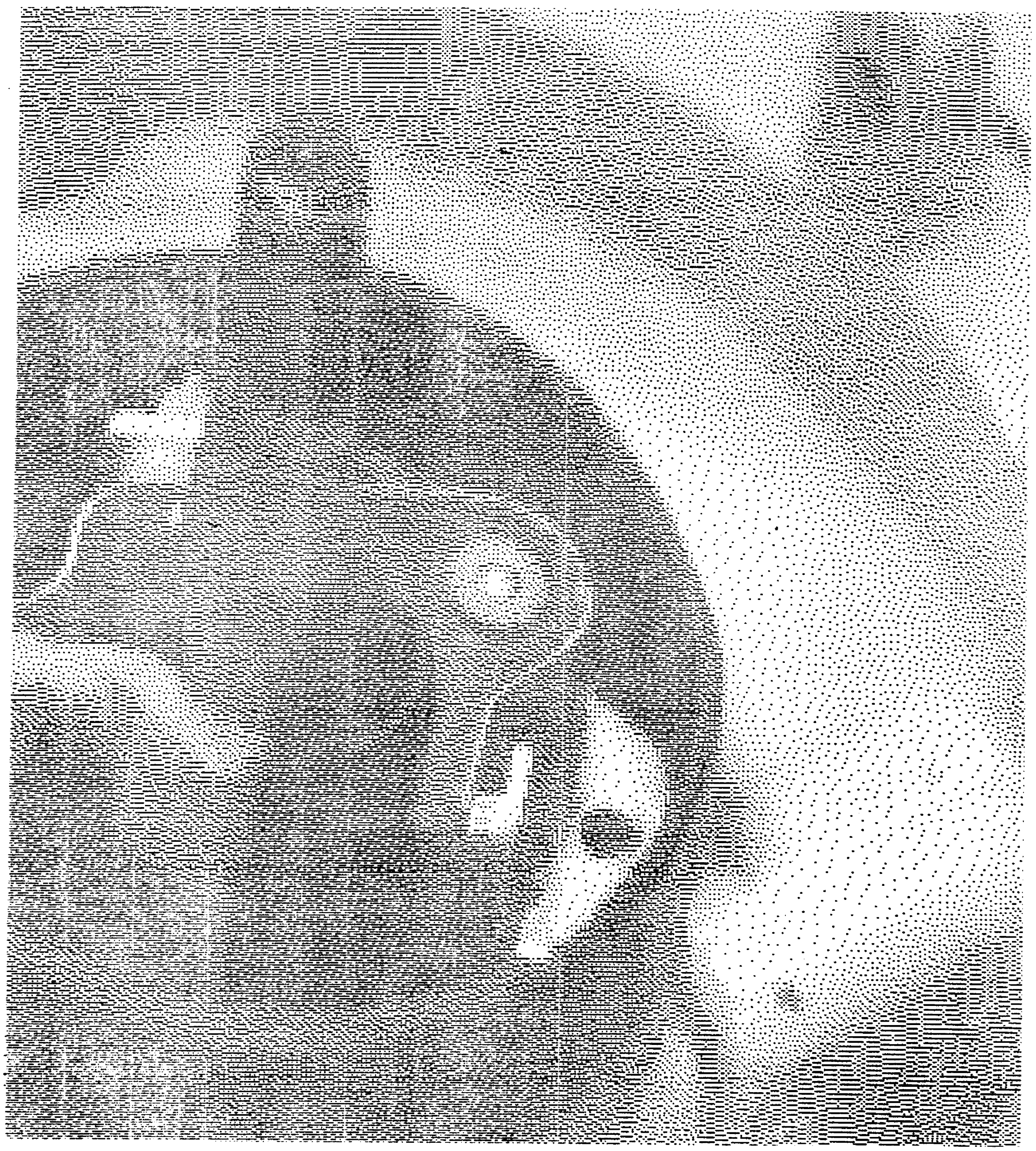


Figure 1.

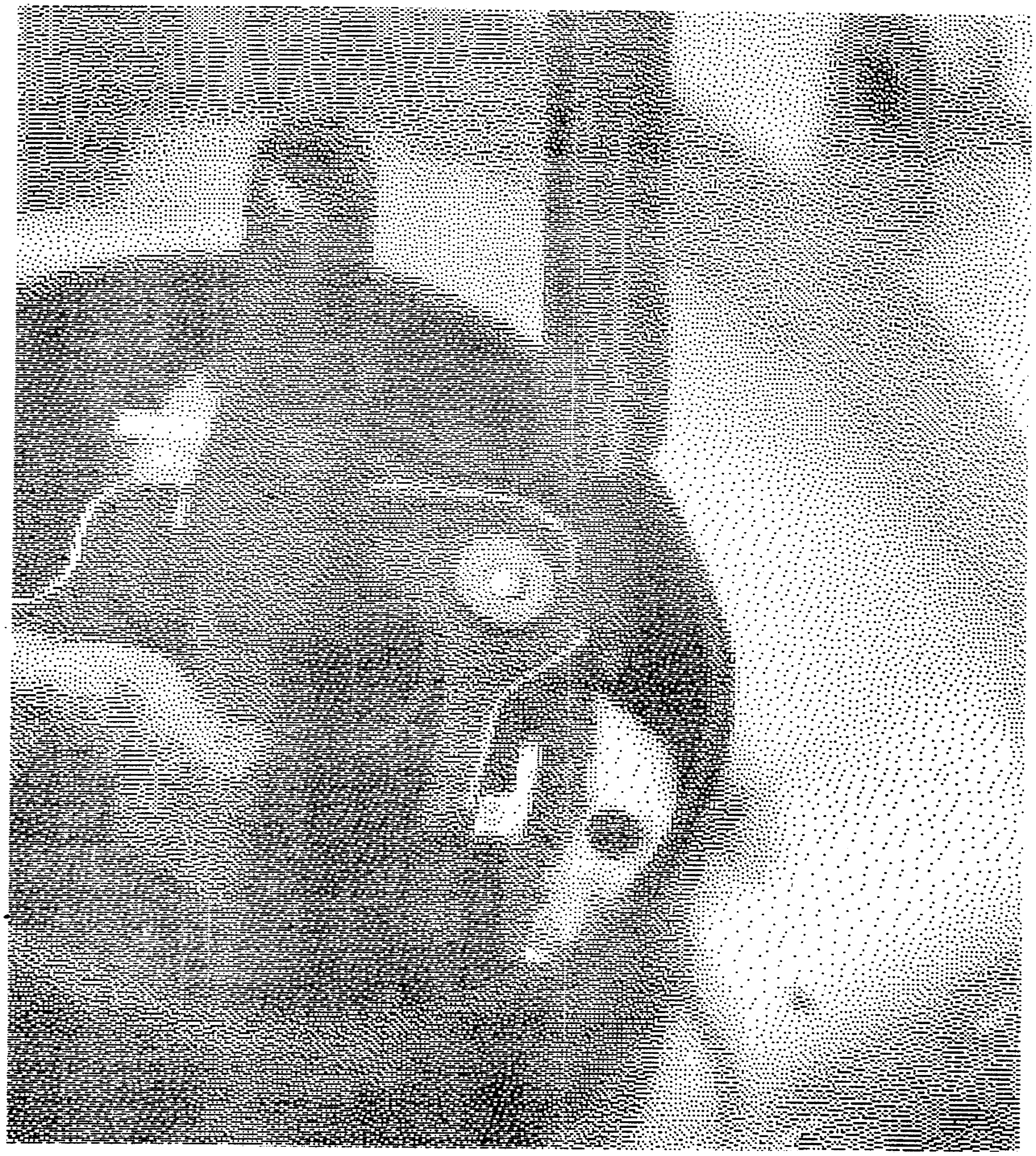


Figure 2.

(1) aligning the first hinge piece with the holes on the doorjamb,
(2) checking to make sure that the (magnetic) screwdriver picks up a screw from the dispenser,
(3) inserting a screw in a hole,
(4) aligning the second piece with the first,
and (5) inserting the pin in the hole.

In conjunction with these "basic" subtasks there are others which could be usefully accomplished by vision. Many of them fall into the category of "implicit" inspection, ie. doublechecking an assumption. Some examples are:

(I) make sure the first piece is present,
(2) make sure that there are holes on the doorjamb and in the hinge pieces,
(3) make sure that the screwdriver is there,
and (4) make sure the pin was inserted completely.

In addition to being widely applicable visual feedback has other (potential) advantages in comparison with touch and force feedback. It is passive. That is, information can be obtained about a part without disturbing it. This may be important for small, delicate parts. Vision offers a *potential* speed advantage because it functions at electronic speeds versus the mechanical speeds which limit touch and force. It also offers a speed advantage because of the possibility of doing the visual checking in parallel with the mechanical operations. For example, if the screwdriver almost always picks up a screw from the dispenser, it would be possible to take a picture of the end of the screwdriver as it was leaving the dispenser and do the verification processing as the arm continues to move toward the hole. If the verification system decides that the screw is there, the arm is free to continue along its path. However, if the screw is not there, the verification system can signal the arm to return to the dispenser to try again. The key phrase in this description is that "the screwdriver ALMOST ALWAYS picks up a screw." How economical this parallel checking is depends upon how often the screw is missed.

Vision also offers a more global view of a situation than is generally possible from touch or force. This is rather vague, but the task of inserting a screw in a hole helps clarify it. Using force feedback it is virtually impossible to decide which way to proceed after one decides that the screw is not in the hole. A mechanical spiral search is time consuming and unesthetic because it is not clear when the arm should give up. Vision, on the other hand, has the potential of determining the dynamic corrections needed to avoid such searches.

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 supposedly servoing **a** screw into a hole, force feedback can indicate that the screw has missed the hole.

There is one other general remark which should be made at the beginning of this discussion. Although most of the examples and systems 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 range devices, laser trackers, or multiple touch sensors.

A DEFINITION OF VERIFICATION VISION

Baumgart has recently distinguished three types or modes of visual information processing: description, recognition, and verification (see [BAUMGART 1974b]). These terms designate general approaches used to solve visual tasks. These approaches can be roughly characterized as follows:

DESCRIPTION ---
almost strictly bottom-up, information is gleaned from the picture to grow a larger and larger model of the scene

RECOGNITION ---
a controlled mixture of bottom-up and top-down, heterarchical, the models suggest what to look for, any features that are found restrict the set of possible models, etc.

VERIFICATION ---
almost strictly top-down, the model is strong enough to dictate exactly what to look for.

Each of these terms implicitly determines a range of tasks for which its approach is appropriate. These ranges can be conveniently defined in terms of three factors: how much the system knows about WHAT can be in a scene (and hence apptar in a picture of the scene), how much the system knows about WHERE things might be with respect to the camera (and hence where they might appear in a picture), and what the GOAL of the task is. The types of tasks are:

DESCRIPTION --
the system only knows the types of features which the objects are composed of and how to build complex models from these features; it has no idea of what the objects are or where they are; the goal is to build a 'model which describes the scene

RECOGNITION ---
there is a fixed set of possible object models and some weak constraints on their position; the goal is to identify which object (or objects) are in the scene and possibly fill in a few parameters about them

VERIFICATION ---

the system knows 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.

These distinctions are not absolutely clear-cut. A classification may depend upon what is defined to be a feature and what is defined to be an object. For example, the intended interpretation is that features are such things as planes and corners, whereas objects are such things as blocks. In this interpretation the standard scene analysis program for the blocks world would be classified as a recognition program. It uses features to recognize objects from a fixed set of prototypes. However, if one considers blocks as primitive features, a similar program might be classified as a descriptive program. It locates features and constructs a model of the scene out of these features.

A more cryptic characterization of these types of tasks is:

DESCRIPTION - grow a model from scratch
'RECOGNITION - pick one of several models
VERIFICATION - locate a particular model.

In order to clarify these terms further, consider the following list of visual tasks and their classifications.

Build a model of the engine casing so it can be recognized as it comes down an assembly line (possibly up-side-down) --- DESCRIPTION

Locate a pump base (model XXX) which is sitting upright on the conveyor belt --- RECOGNITION because the various rotations present significantly different views of the object to the camera

Locate a pump base after it has been placed in a vise which is at a known position --- VERIFICATION if the base is placed at approximately the same place in the vise each time

Locate the gasket after the arm has positioned it 1 cm above a pump base which was just been located --- VERIFICATION

Locate the objects on top of the table so an arm can dust around them --- DESCRIPTION because the objects are described in terms of the volume they occupy without any concern for what they are

Describe what is on the table --- RECOGNITION if the types of objects are all known in advance

Locate the corner of the table --- VERIFICATION if it is a known table and almost at its expected position

Describe an unidentified flying object --- DESCRIPTION because one has to revert back to a composition of features: "it was grey, generally oval, with a bump on top"

Find the road in a picture (which contains a normal driver's view of an uncluttered road) --- RECOGNITION, unless the type of road and the view art standardized enough to predict where the edge of the road is, what it looks like, etc.

Having found the road in one scene, locate it again in a picture taken a few feet further along the road --- VERIFICATION because the previous picture provides an excellent model of the new view

Notice the frequency of such subjective words as: approximately, normal, standard, and predicted. These especially occur in the discussion of verification vision tasks. They occur because the distinction between recognition and verification is often pragmatically defined. If there is no significant question about what is being looked at and the available operators can locate the important features, the task can be considered a verification task.. However, if the views (even of a known object) are sufficiently different that different sets of features have to be used, then the problem is a recognition problem.

This suggests that verification is easier than recognition. In fact, verification is often a subtask of recognition; after a prototype has been chosen, a verification subtask is set up to verify that prototype.'The idea is that if there is enough information available to restrict the problem so that the features are reasonably distinct and there aren't many surprises, then the problem can be approached in a more direct way. So how is this done? What information can turn a problem into one of checking as opposed to choosing? What structures should bt available in a verification system so that this information can be integrated in the most effective way?

It is intuitively clear what makes a task easier, but it's not clear how all of the information should be combined. For example, consider the servo-a-screw-into-a-hole problem mentioned earlier. The steps involved are:

(I) locate the hole without the screw being in the picture,

(2) move the screw into the picture and locate it against the now known background,

and (3) decide how to move the screw closer to the hole, move it, locate it again, etc.

Assume that the arm picks up the part with the hole in it and places it in a vise (whose position is reasonably well known). In that case the hole may appear displaced in a picture at step (I) because of several reasons:

(a) the arm is not exact,

(b) the arm does not know exactly where to go, even if it could position itself precisely (it doesn't know where the part is to be picked up or exactly where the vise is),

(c) the part does not seat in the vice exactly as planned,

and (d) the calibration between the arm and the camera is not exact.

Having found the hole in step (1) there is enough information to reduce the problems caused by (c) and (d). Thus, there are fewer uncertainties for step (2). And for step (3) the main factor contributing to the error should be (a) since the problem will have been reduced to an analysis of the relative displacement between the tip of screw and the hole.

Also notice that more and more information about the expected appearance of the **objects** can be brought to bear as the system progresses from step to step. For step (I) the system may have a picture of this same step during a previous assembly and possibly a synthetic picture generated from its model of what is expected in the 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 system has all of the earlier pictures which show the actual glares, shadows, light levels, etc. as the screw approaches the hole.

Thus, the three steps offer three different sets of tolerances and levels of knowledge about the appearances of the objects. The increased information should make each successive step easier and faster. The next sections investigate various semantic systems which would make it possible to take advantage of this type of information.

VERIFICATION VISION SYSTEMS

INTRODUCTION

The overall goal of this section is to motivate and describe the capabilities for a system which can be easily "programmed,, to verify visually the presence and location of a desired object within a complex scene. One possible way to approach this would be to describe a completely automatic system. But this is unrealistic because most of the basic facilities would be overshadowed by fancy subsystems, all of which are beyond the current state of the art. Therefore this section presents an ordered set of verification vision systems, starting with a simple, interactive one and ending up with the fully automatic one.

Each system introduces new semantic structures to make the system more powerful and/or easier to use. These facilities are motivated out of need. That is, successively harder tasks are described, the current "best" solution Is analyzed, and new capabilities are suggested. In this way new facilities are derived to solve problems in weaker systems. Protocols are also presented to give a unified view of the the capabilities of the system under discussion.

Throughout the discussion some solutions (Ie. possible implementation ideas) will be suggested, but there is no claim that all possible solutions have been considered or that the ones mentioned are necessarily the best. The "best" solutions depend upon tht situation, the available equipment, and the goal.

A BASIC, INTERACTIVE SYSTEM

The format for this subsection consists of: (1) a description of a task and a proposed solution, and (2) critiques which suggest new capabilities.

TASK: DETERMINE WHETHER OR NOT THERE IS A SCREW ON THE END OF THE SCREWDRIVER --- (ASSUME THE SCREWDRIVER MOVES IN FRONT OF A CLEAR (IE. UNCLUTTERED) BACKGROUND AND THAT THE ARM CAN POSITION THE SCREWDRIVER WITHIN ±5 DEGREES OF VERTICAL AND WITHIN A SPHERE OF RADIUS 1 CM)

SOLUTION: USE A. SPECIAL-PURPOSE SCREW FINDER AND
SCAN THE WHOLE PICTURE

Why scan the whole picture?

Sometimes the screw will appear at one point in the picture, sometimes at another. If the total range of possible positions is only a small portion of the picture, there is no reason to scan the whole picture. But how can the region of possible positions be determined? One way would be to move the screwdriver manually around within its range of possibilities and keep track of where it appears in the picture. The system could provide the user with a representation for 2-D regions (such as rectangles or convex polygons) and a way of creating such regions. Finally the system should include a way of restricting the search to one of these regions. In this way the relevant region can be interactively determined and used.

The region of possible positions for a feature is called the "tolerance region" about that feature. The assumption is that the camera is at a fixed position and orientation. A feature's tolerance region is specified in terms of the camera's screen coordinate system. In order to find the feature one must only search that region. What appears in that part of the picture changes depending upon where the object (eg. the screw) happens to be during that assembly.

The tolerance region must be determined only once, but it is used each time the test for a screw is made. This distinction between advanced planning and execution is an important one in verification vision. The advanced planning or "training" session is designed to predict as much about the events during an execution as possible. The information gained in this process is used to make the execution phase more efficient.

TASK: LOCATE A SCREWHOLE IN A LARGE OBJECT (EC. AN ENGINE CASING) --- (ASSUME THAT THE OBJECT IS SITTING UPRIGHT ON THE TABLE AND ITS LOCATION IS KNOWN TO WITHIN ±3CM IN X AND Y AND ±10 DEGREES ABOUT Z); THE GOAL IS TO LOCATE THE HOLE WITHIN A TOLERANCE OF ±.2CM IN X AND Y.

SOLUTION: USE A SPECIAL-PURPOSE HOLE FINDER AND
SCAN THE NECESSARY REGION

Generally the reason for scanning a special-purpose operator over a tolerance region is to locate a particular feature. In this case it would be nice to know in advance if there are other parts of the picture that appear similar to the desired feature and might appear within that region, especially if the operator may confuse one of them with the actual feature. If an operator happens to match several of these confusing "decoys, its discrimination should probably be improved (eg. by changing thresholds, or *by -using a* larger local *context) or it should be replaced. Since the operators are not foolproof, there is no way to guarantee that an operator won't locate an unforeseen decoy during an actual assembly Therefore, the execution system will have to be able to handle erroneous matches. But it would also be nice to have an estimate of how unique and reliable an operator* is *so that* it *can be improved or so that special steps can be taken to disambiguate the situation. Thus, another piece of information a training. session might try to approximate is the set of possible decoy matches for an operator. How can this be done?*

First, it is important to understand how confusions may be formed. In the previous task the background stays. fixed since it is formed by stationary objects on the table. The uncertainty about the position of the'screw makes it possible for the screw to move about in front of the background. The only ways a decoy match might arise are that (1) some part of the background looks like a screw (see figure 3) or (2) some part of the boundary of the screw and the background appears like the screw. Notice, however, that if the goal feature (eg. a hole) is part of a larger object which moves, the confusions only arise because some other part of the larger object looks like the goal.

One *way* of locating possible decoys is the following:

(1) determine the tolerance region about the hole (as in the previous example),

(2) set up several example scenes such that the hole appears at different places within the tolerance region (in accordance with the constraints on the part),

and (3) scan the operator over the whole tolerance region in each of the resulting pictures, seeking decoys.

Figure 4a shows the camera's view of an abstract scene. A potential feature is indicated by the arrow. Figure 4b shows the tolerance region for that feature overlayed on top of the picture. Notice the screen coordinates, X and Y. Figure 4c shows the camera's view after the object has been moved and figure 4d includes the same tolerance region. Since the tolerance region is defined in terms of the camera's screen coordinate system it stays fixed while the features move around underneath it; it is at the same place in figures 4b and 4d. In both cases the desired feature appears within the tolerance region (as it is must). However, notice that there are other portions of the picture that resemble the feature and, in fact, one

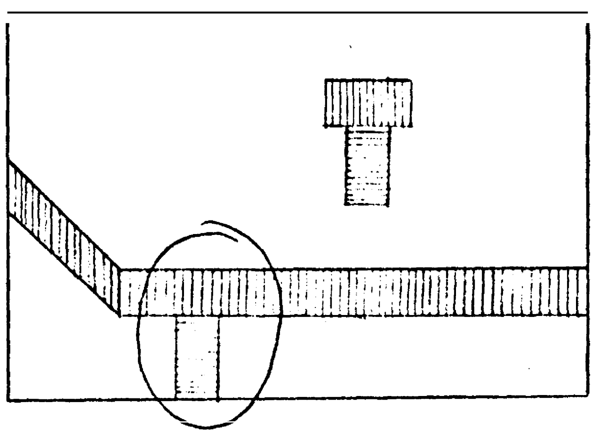


Figure 3.

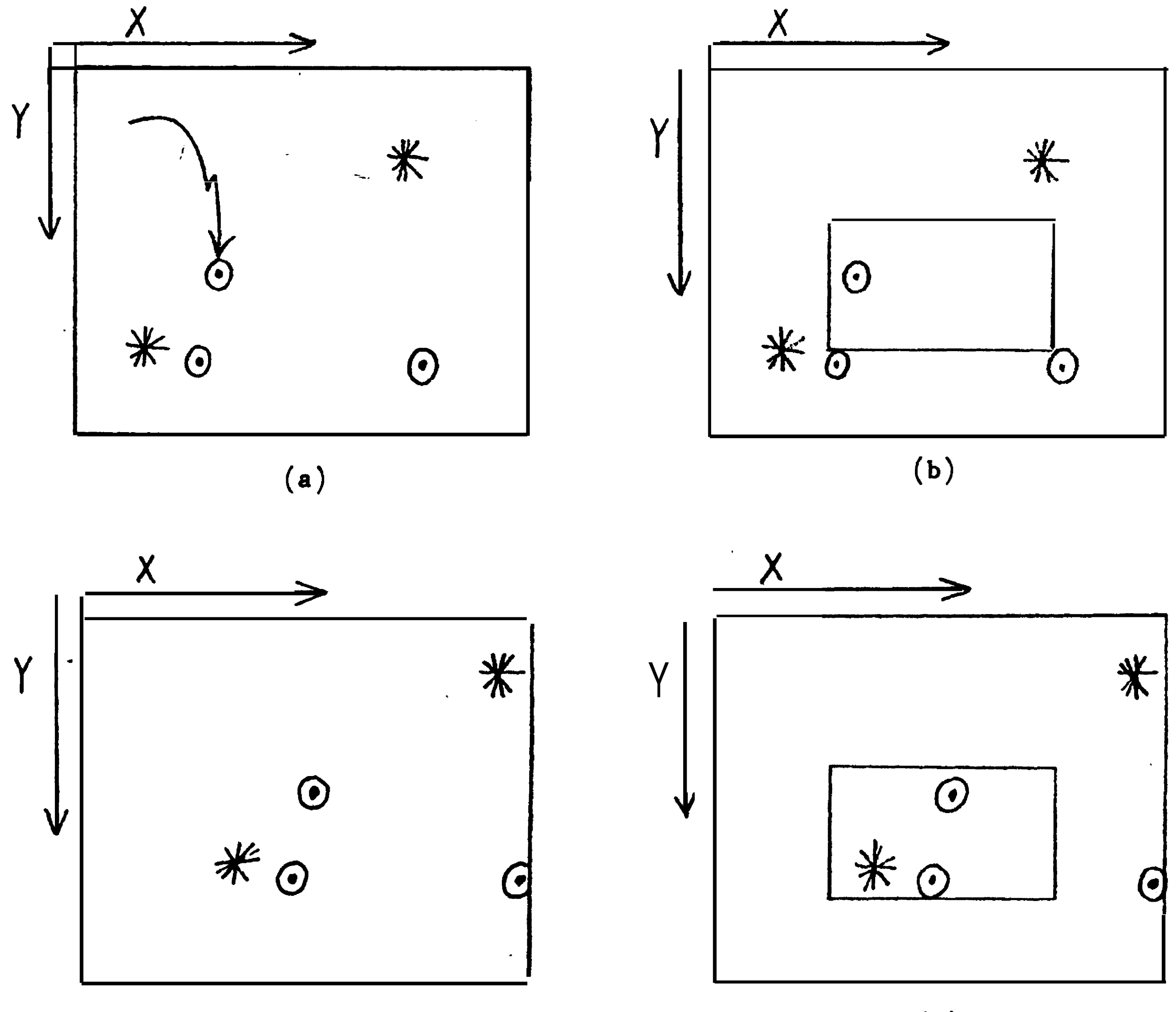


Figure 4.

such decoy shows up inside the tolerance region in figure 4d. This means that if the object happens to be in that position during an actual assembly and if the operator is scanned over the tolerance region, it may locate either one of these matches first. The algorithm mention& above would locate both the desired feature and the decoy and alert the system that there is a possible confusion for that operator

This algorithm would work, but it would rquire the analysis of several different pictures in order to cover all possible situations. There is another way of checking for confusions which. only rquires one training picture, but it rquires two simplifying assumptions:

(1) Any feature matched by an operator remains essentially the same throughout the range of possible views of the scene (ie. no perspective-induced changes of appearance).

and (2) Translational uncertainties dominate rotational uncertainties; all views of a scene can be approximated as two-dimensional translations of one canonical view. This is called the translation assumption.

The goal of the training for decoys is to locate any portions of the scene that may appear inside the tolerance region for a feature and look similar to that feature. Figure 4 shows that different features appear inside the tolerance region depending upon where the object is in the scene. The idea for a new training algorithm is this: consider only one picture of the scene, but check all points in the picture that might be moved into the tolerance region by some movement of the objects in the scene (in accordance with their constraints). For example, figure 5 graphically develops the necessary region in the case that the tolerance region is a rectangle. If the feature appears in the upper right-hand corner of the tolerance region (see figure 5a), the portion of the picture that is included in the tolerance region is shown in figure 5b. If the feature appears in the upper left-hand corner, the portion of the picture (with respect to the feature) that would show up in the tolerance region is outlined with dashes in figure 5d. The complete region to be checked is four times the size of the original region and is centered about the feature. Figure 6 develops the case when the tolerance region is a triangle. Notice that the new region is six times as large. A circle rquires that a circle with twice the diameter be checked. Non-convex regions can produce regions which are any number of times the size of the original (see figure 7).

If the translation assumption is not true, the methods mentioned above do not work. For example, consider the case (shown in figure 8) where there is only an angular uncertainty. The featurecan appear anyplace along the indicated arc. If rectangles are being used to represent tolerance regions, the appropriate region is shown in figure 8b. Using the translation assumption the area to be checked for possible confusions would be the one

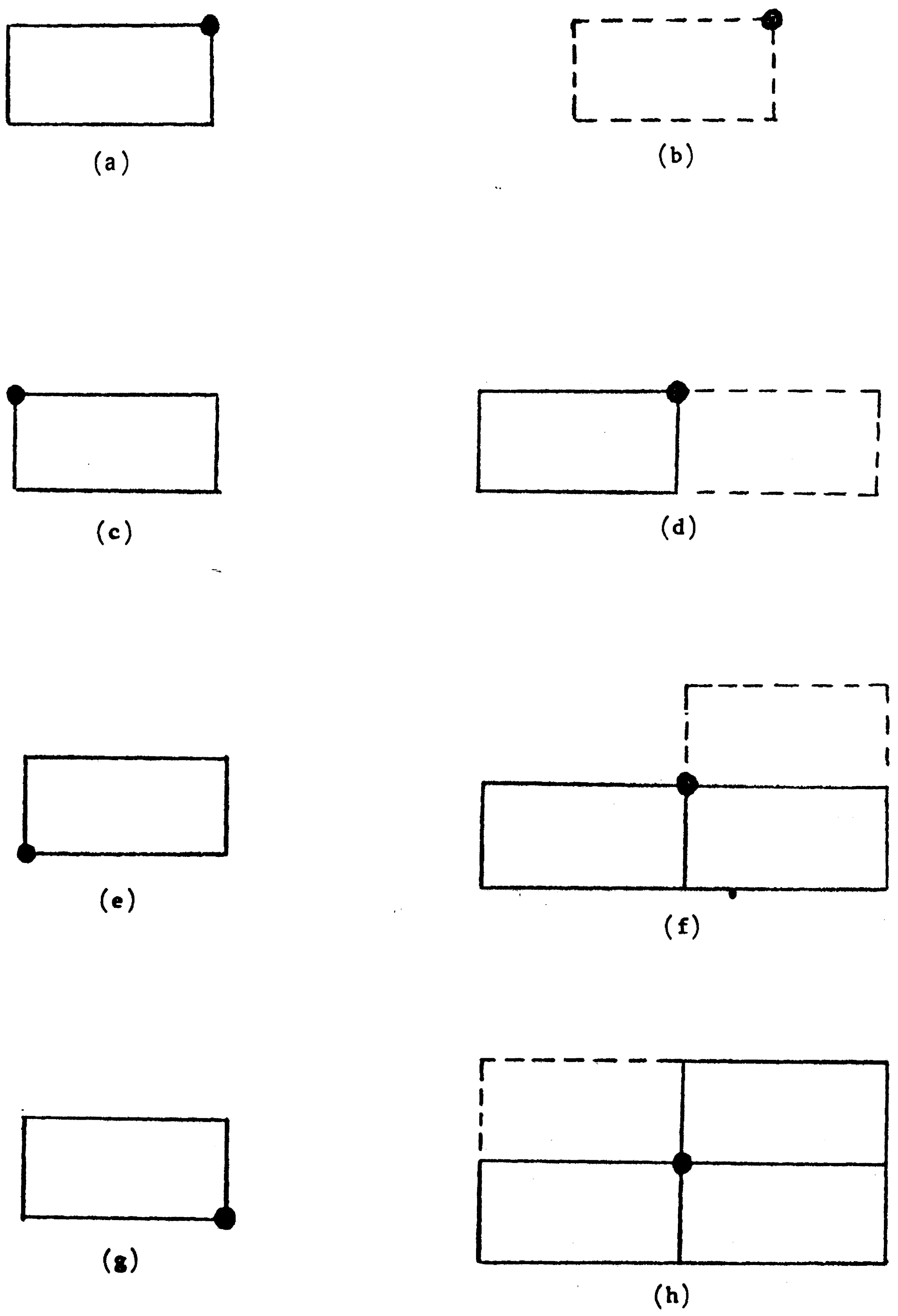


Figure 5.

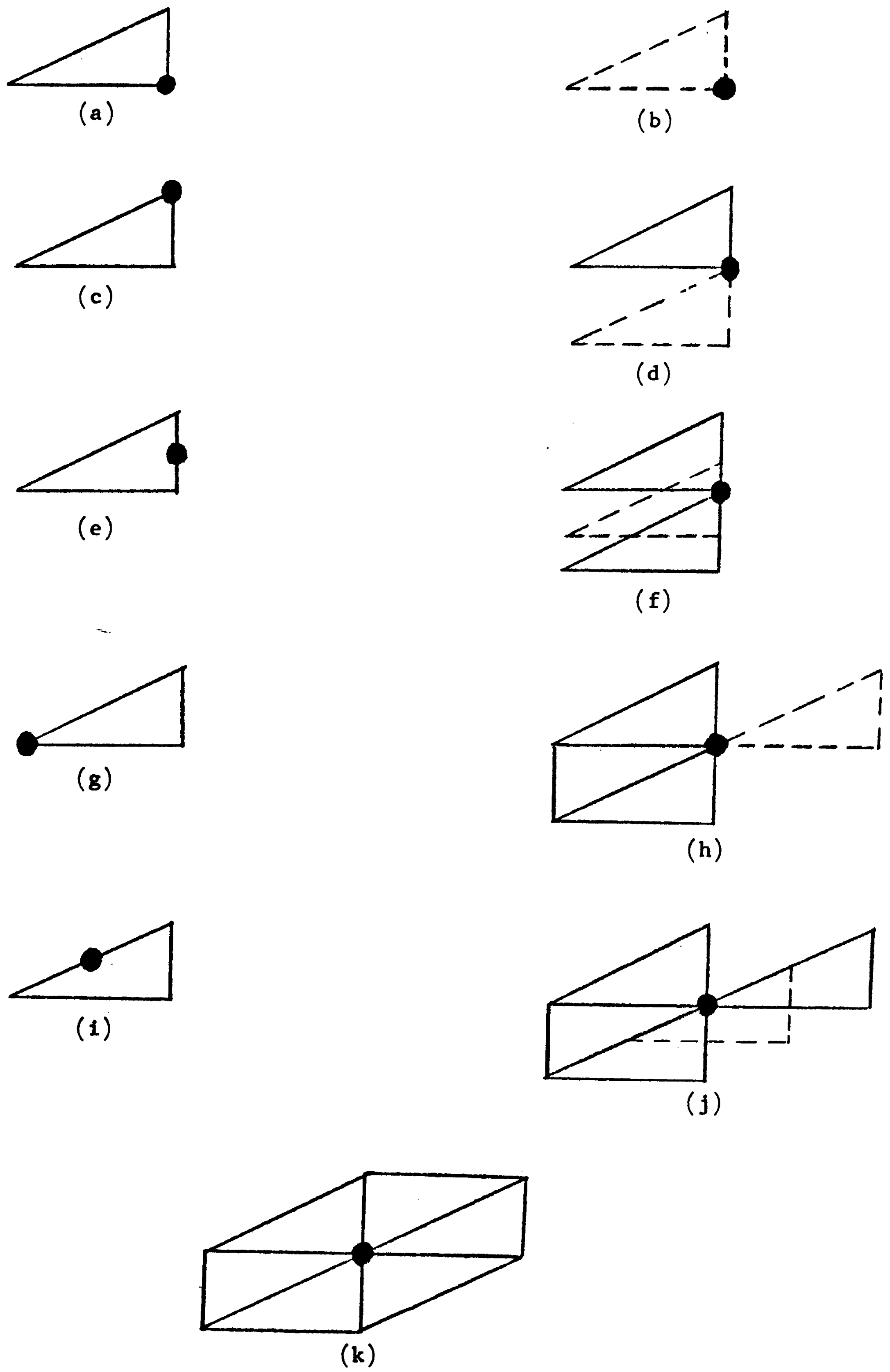


Figure 6.

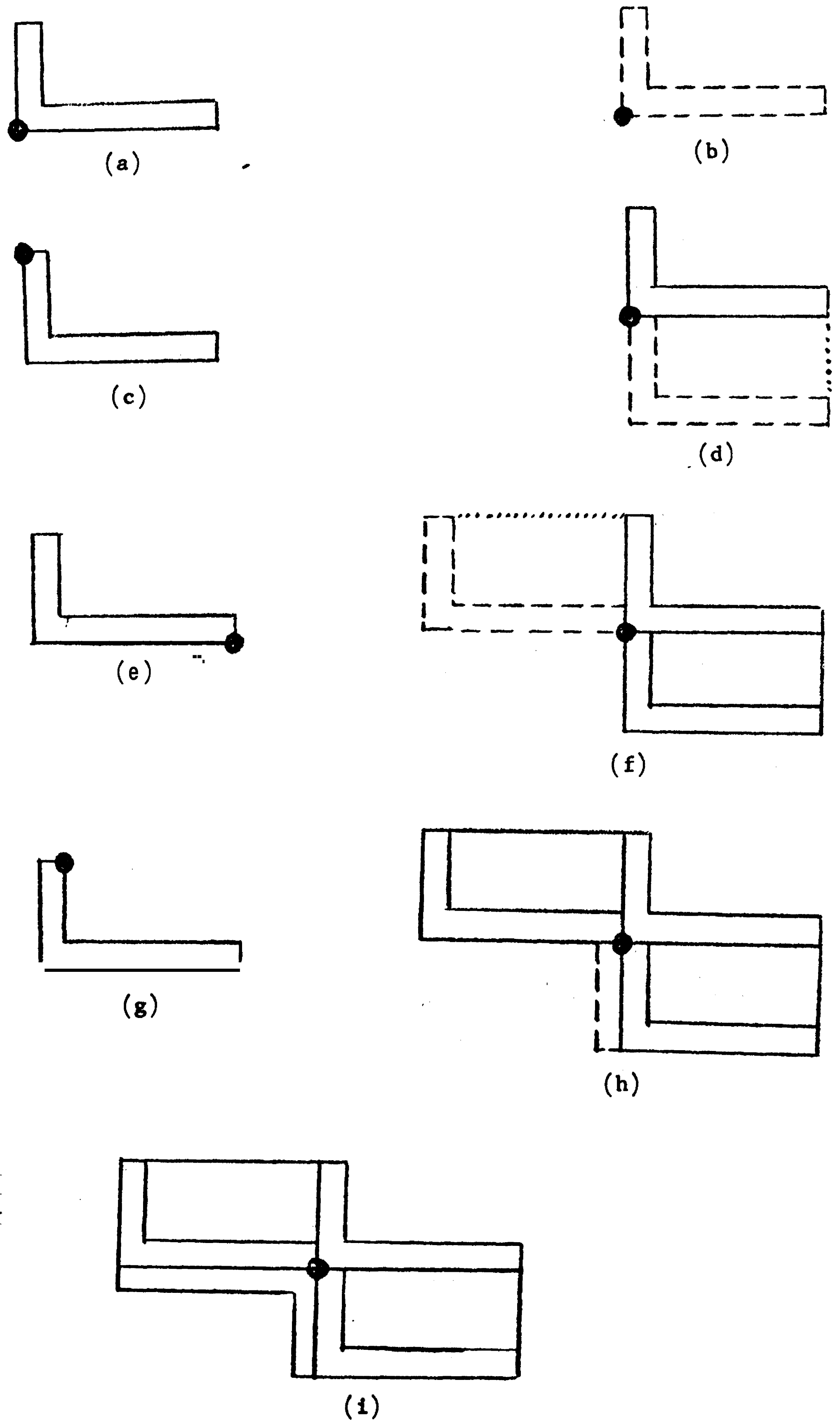


Figure 7 .

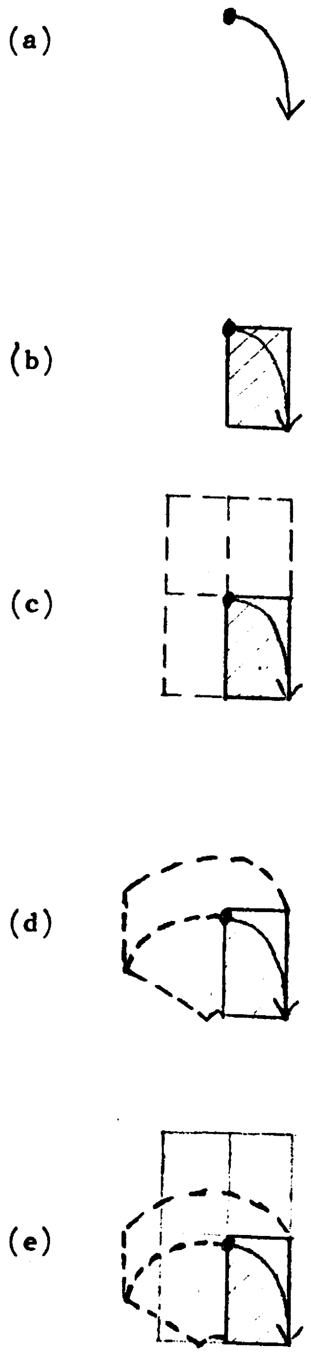


Figure 8.

shown in figure 8c. However, in fact, anything in the dotted region of figure 8d might appear in the tolerance region. Fortunately, the translation assumption usually holds If not, it is always possible to use the first algorithm mentioned above.

If the hole is found, what is the precision (in 3-D) of the result?

There are two keys to answering this: (1) a calibration of the camera with respect to the part and (2) an estimate of the precision of the hole-finding operator in terms of pixels (ie. picture units). The (planned) distance to the hole can be computed from the calibration. From this distance it is possible to compute the resolution of one pixel in a plane parallel to the image plane passing thru the center of the goal feature (eg. the hole). This resolution can be converted into a combination of equivalent resolutions along the axes of any other coordinate system. In the task mentioned above the desired coordinate system is the table. These new resolutions for one pixel can then be combined with the precision of the hole-finding operator to give the desired result.

If the goal tolerances are in a plane (as they are for this example) it is possible to compute the precision along the two coordinates of that plane even if the calibration only consists of a collineation matrix between the plane of the goal and the image plane. A collineation matrix is a one-to-one mapping between the image plane and some other plane. It does not indicate where the camera's lens center is or the distance between matching points. However, since the precision of the operator defines a region about the feature it matches, the collineation matrix can be used to map the extreme points of this region (eg. the corners of a rectangle) onto the goal plane. A region in the goal plane with these extreme points forms the basis for deciding the expected precision in that plane.

If the hole is found, how can useful 3-D information be determined? For example, what is the XY correction required by the arm to accommodate to the actual position of the Me?

If the object with the hole is constrained in some way so that the hole must lie within a plane (eg. the part is sitting upright on the table or held in the plane of a vise) the hole's position in the image can be directly converted into a point on that plane. The equation of the plane and the point on the plane determine a unique point in S-space. Since this planar assumption is true for the example task, the hole's position in the image can be easily converted into a useful quantity such as "the hole is displaced .2cm in X and 1.0cm in Y from its. planned position."

If the planar assumption is false (eg. because the object is being held by an arm), one possibility is to use stereo vision. Stereo vision involves locating features in the images of two calibrated cameras and computing their 3-D location by triangularization. If stereo is used, there is also a method for computing the expected precision of the result.

A third way of determining the 3-D information required by an arm is to use a 3-D model of the object to locate several feature points on the object. The model indicates the points on the object that match the visual features being located in the image. Given this model and the 2-D image locations of the feature points it is possible to compute a new 3-D position for the whole object. This is essentiaiiy the same problem as calibrating a camera. A variation on this idea is to use stereo to locate several features in the two views, compute their 3-D locations, and then do a least-squares fit on these new 3-D positions to determine the best estimate for the object's position.

There are several other ways of determining the 3-D location of a point; such as motion parallax, direct range finding, and laser tracking.

The suggestion which uses several feature points requires several different operators, Is there an easy way of setting up several operators?

Cross-correlation is one of the easiest and most flexible. It is generally easy to set up: interactively point out a promising patch in a training picture and let the system check its distinctness. Correlation offers normalization to compensate for an overall brightness change and it is easy to design special shapes and even add weights. It requires a previous picture of the scene. In programmable assembly this can easily be provided by taking a picture of an example assembly (ie. during a training session). The main limitation on correlation is that it does not work well when the new picture includes *a* rotation with respect to the training picture. It would be possible to use several operators, each designed to handle a part of the rotation range, but any one of the operators is limited to a small angular range. Quam has carried out some analysis to determine the effects of non-translational differences between the two pictures (see [QUAM1971]), but the limits are still not well determined. Functionally it seems possible to set the acceptance thresholds so that reasonably sized correlation patches (eg. 15x 15 pixels) correctly match whenever the rotation is less than ten degrees. More analysis (both theoretical and practical) needs to be done.

The use of several features means that each feature must be checked for possible confusing matches. As mentioned earlier the setting up of tolerance regions and checking could be done manually, but what is required to do it automatically?

To answer this there has to be a system for describing the tolerances and constraints which apply to the various objects in a scene. Typical constraints are: plane P of the object contacts the XY plane of the table, the angle of the shaft is known to within ± 15 degrees, and point T lies within the rectangular box B. To state constraints of this sort, the S-D point modelling system would at least have to be enriched to include some form of a surface patch (eg. a polygon) and a volume (eg. a rectangular box) plus predicates for saying that a point "lies-in" a polygon; etc. Then there would have to be a method to take a list of constraints and produce the appropriate volume within which the goal point must lie. The camera model could then be used. to project that 3-D range onto the image. This projection could even take into account the precision of the camera calibration by making the projection of a point be a small region. Thus, the constraint model, the constraint solver, and the projector form a complete system for automating the determination of tolerance regions.

Taylor (see [TAYLOR]) has investigated a few types of constraints and various ways of representing them. He also has a system for producing the resulting constraints on the positions of features of interest.

There is one more thing required. to check for possible erroneous matches automatically: a method to produce the region of possible confusions from the feature's planned position and tolerance region. The complexity of this algorithm depends upon the generality of the representation for tolerance regions and the model of changes from one view of the scene to the next. If tolerance regions are represented by rectangles and the changes are assumed to be translational, the algorithm mentioned earlier would be sufficient.

This completes the facilities which make up the "basic" verification vision system. In fact, the automatic tolerance checking capability should probably be considered optional for, the most basic system. The semantic mechanisms required by these facilities are given below as a review.

CAMERAS AND A METHOD FOR CALIBRATING THEM WITH RESPECT TO THE TABLE (OR OTHER OBJECTS)

A REPRESENTATION FOR 2-D TOLERANCE REGIONS

A METHOD OF SEARCHING A 2-D TOLERANCE REGION

A METHOD TO COMPUTE A 3-D POSITION FOR A FEATURE GIVEN TWO SETS OF COORDINATES FROM STEREO VIEWS

METHODS TO DETERMINE THE EXPECTED

PRECISION OF A MONOCULAR OR STEREO LOCALIZATION

A SYSTEM FOR 3-D POINT MODELS OF OBJECTS

METHODS TO DETERMINE THE BEST ESTIMATE FOR THE NEW POSITION OF AN OBJECT GIVEN THE IMAGE COORDINATES FOR SEVERAL FEATURES (BOTH 2-D AND 3-D)

AN INTERACTIVE SYSTEM FOR SETTING UP RELIABLE CORRELATION OPERATORS AND INDICATING THE MATCHING FEATURE ON THE 3-D POINT MODEL OF THE OBJECT (THE CORRELATION SYSTEM MIGHT INCLUDE AN AUTOMATIC WAY OF SETTING THE THRESHOLDS REQUIRED TO DECIDE IF THERE IS A MATCH OR NOT)

A SYSTEM FOR DESCRIBING CONSTRAINTS

A REPRESENTATION FOR TOLERANCE VOLUMES

A METHOD FOR PRODUCING THE TOLERANCE VOLUME FROM A SET OF CONSTRAINTS

A METHOD FOR PRODUCING THE CORRESPONDING 2-D TOLERANCE REGION IN AN IMAGE FOR A TOLERANCE VOLUME

A METHOD FOR PRODUCING THE 2-D REGION TO BE SCANNED FOR POSSIBLE CONFUSIONS

In order to present a better idea of how a system with these capabilities might function, protocols are given below showing how a user might "program" solutions for a few tasks, including the two example tasks.

(1) CHECK 'FOR THE SCREW ON THE END OF THE SCREWDRIVER

(a) Position the arm, screwdriver, and screw at the expected location.
(b) Aim the camera so that the screw is visible.
(c) Take a reference picture.
(d) Manual ly move the arm so that the screw covers its range of uncertainty and mark the extremes.
(e) Produce a 2-D tolerance region for the screw.
(f) Visually check the background for homogeneity over thisregion.
(g) Assume that one correlation operator is sufficient. Interactively define a correlation operator to locate the screw.
(h) Move the screw to another position within the al lowed tolerances.
(i) Take another picture and check the effectiveness of the correlation operator. Can it find the matching point in ths region of possibilities?
(j) Take a picture without the scrsu on the end.
(k) Apply the correlation operator and make sure that i t doesn't find any srronsous matches.
(l) The 'program' is essentially: take a picture, apply ths operator throughout the necessary region. If it finds a match, assume that the screw i s there, otherwise, assume that i t isn' t.
(m) If there are confusing points in the background, the user can try a new position for checking the screw, a new camera position, or increase the number of operators and check for consistency as mentioned in the next example.

(2) LOCATE THE HOLE IN THE ENGINE CASING

(a) Position the object at the expected position.
(b) Aim the camera so that the hole and several other features on the object are visible.
(c) Cal ibrate the camera.
(d) Check the potential precision at that camera location.
(e) Take a reference picture.
(f) Interactively choose reliable correlation operators.
(g) Set up a 3-D point model which includes the points that correspond to the features being matched by the operators.

(h) Extend the point model to include the plane of the base and a plane for the table top. Add a palygon, P, to the representation of the table'8 surface so that it can be used to state the X and Y constraints on the uncertainty of the casing.

(i) State the constraints on. the casing:

  (1) The plane of the base contacts the plane of the table top.

  (2) The Z of 'the base points in the same direction as the Z of the table.

  (3) The main reference point on the casing lies within the polygon, P, on the table,

and (4) The rotation about the Z axis is limited to plus or minus 10 degrees.

(j) Have the constraint solver produce the 3-D volume that represents the range of possibilities for feature F1. Since the casing is known to be sitting upright, this volume will only be a 2-D patch.

(k) Produce the corresponding 2-D tolerance region,

(l) Produce the region to be scanned for other matches.

(m) Scan that region with the appropriats operator to 8ee if there are any possible ambiguities. If there are, throw that operator away or change it so that it is unique.

(n) Do this for all of the operators.

(o) The 'program' would then be: take a picture of the object, search for the operators within their regions, for each one that is found use the feature's known height and the collineation matrix to determine the corresponding 3-D position, map the new 3-D positions onto the plane of the table top, compute the best estimate for the new position of the object (throwing out inconsistent points), and finally check the precision of the result to make sure it is within the desired tolerances

(3) DETERMINE THE RELATIVE DISPLACEMENT OF A SCREW FROM A KNOWN HOLE

The program would be essentially the same, but stereo image8 would be used to determine the new 3-D positions. The best estimate for the new position would be carried out in 3-D.

A SIMPLE STRUCTURE SYSTEM

The 'basic system' described in the last subsection has several problems and limitations. For example, it assumes that correlation works. That is, it assumes that one correlation operator can be set up to locate each feature in a new picture. This is not always a good assumption, especially if the objects in the scene can rotate more than 15 or 20 degrees. The features change their appearances too much. It may be possible to locate a feature by setting up several correlation operators, each of which is tuned to a certain portion of the total range of angles. Sometimes even this is not possible because small rotations and translations can cause large changes in appearances when one part of an object is occluding another.

There is no attempt made in the basic system to try to use the location of one correlation point to help find other points. Intuitively it seems possible to predict more precisely where a feature point might appear after a few others have been located. For example, if the object being looked at is rigid and if the main effect of the uncertainties is an unknown translation in the picture, once one point has been found the observed translation for that point can be used as an estimate for the displacement of the other points.

The basic system does not make use of any 'extended' features which may be easier to find and can be used to limit the amount of effort required to find correlation points. Since a correlation feature is essentially a point, the whole tolerance region has to be scanned to locate a match. A search for a line segment, on the other hand, might consist of only a few linear searches across the tolerance region.

Most of these limitations are concerned with the use of structure: the structure of the objects which are being looked at. This subsection investigates what is necessary to take advantage of some of this structure. Again the method is to state a task and enumerate the basic requirements needed to accomplish the task.

TASK: LOCATE A SHAFT WHICH HAS BEEN PLACED IN A VISE --- ASSUME THAT THE SHAFT LIES IN THE PLANE OF THE VISE, THAT ITS ORIENTATION IS KNOWN TO WITHIN ±20 DEGREES, AND THAT THE POSITION OF ITS END IS KNOWN TO WITHIN ±1CM. THE GOAL IS TO DETERMINE THE POSITION OF THE END OF THE SHAFT TO WITHIN ±.15 CM ALONG EACH AXIS AND DETERMINE THE ANGLE OF THE SHAFT (IN THE PLANE OF THE VISE) TO

'WITHIN ±2 DEGREES.

SOLUTION: SINCE A SINGLE CORRELATION OPERATOR DOES NOT WORK RELIABLY OVER A 40 DEGREE RANGE, SET UP THREE CORRELATION OPERATORS FOR EACH FEATURE. APPLY ALL OF THEM AND USE ANY OF THEM THAT MATCH IN THE COMPUTATION OF THE OBJECT'S LOCATION.

This solution does not take full advantage of the object's structure to reduce the amount of work required or to insure a consistent set of matching features. The structure is only used to check consistency and to compute a new estimate for the object's position after all of the features have been located. Are there incremental approaches for locating an object? What other types of features besides correlation are there and what can they contribute toward the localization of an object?

There are several other types of features, such as line segments, curve segments, homogeneous regions, and textured regions. They are all 'extended' features, but they have quite different functional characteristics. For example, a rotation changes the orientation of a line segment, but it still appears as a line segment. One of the standard edge operators can be used to locate a point on such a segment. And in addition to returning the position of the point, it can produce an estimate for the orientation of the line. Since line segments are extended, they should be easier to find than a point. The longer the better. Instead of scanning a whole region, a few linear scans across the region are generally sufficient. These characteristics would be very useful for the shaft location example. Consider the following strategy for locating the shaft:

(1) locate a couple of points on the side of the shaft,

(2) use these to determine the shaft's orientation,

and (3) use that to choose between three training pictures and the associated correlation operators (which now only have to cover 13 to 14 degree ranges).

In addition to choosing the right correlation operators, a point or two on a line segment can reduce the region the operators have to cover.

Notice that this strategy is an ordered set of steps (ie. a program). The basic system did not provide for a user-defined program. There was only a fixed control structure: locate as many of the correlation features as possible and use them to compute a new estimate for the object's position. The 'simple structure' system, on the other hand, needs some way of representing a user-defined program. The idea is that a much larger range of tasks can be handled by a system which provides a way for the user to take advantage of a few pieces of structural information (as in the example of the shaft). The programs are expected to be simple and straightforward. Hence the name for this type of verification system.

A trace of one of these programs will take the following form:

(1) try to locate a feature

(2) make an inference about the position of the next feature and/or the position of the object

(3) try to locate another feature

(4) make an inference

(n) compute the final estimate for the object's position

There are several forms that the program itself may take: a set of routines which can be called from some general-purpose language, a set of processes that communicate with each other, or a graphstructure of features that an interpreter looks at and decides what to do next. No matter what the form actually is there are a few capabilities which should be included. There should be some way of continuing a search if one 'location' for a feature is later determined to be inconsistent. There should be a direct way of incorporating the fact that a feature has been missed. Misses are important. Knowing that some feature is NOT in some region helps restrict the possible positions for the object in much the same way that knowing a point is in a region does.

Each extended feature has its own 'structure' (eg. straight edges appear to be straight lines in a picture). But there is also a structure which relates one feature to another (eg. the screw hole is 2 cm from the edge). In the basic system this structural inter-relationship of the features is only used at the very end of the process when it computes a new estimate for the position of the object. That is, the basic system does NOT use one feature's position to help locate another one. The simple structure system should have some way of doing that. For example, consider figure 9. If two points have been located on the side of the shaft not only can they be used to choose which set of correlation operators to use, they can reduce the tolerance region about the end of the shaft. Figure 9a shows the planned position of the shaft, its side, and its end. Figure 9b shows two points that have been located on the side of the shaft. The uncertainty of their position is represented by the small rectangles about the points. This uncertainty carries over to the computation of the angle of the side of the shaft. Since both points are known to be on the line segment which is the bottom of the shaft, they restrict the linear motion of the shaft as shown in figure 9c. The combination of these two uncertainties (ie. the angular and the linear) generates the small region shown in figure 9d which represents the total range of possibilities for the position of the end-of the shaft. This

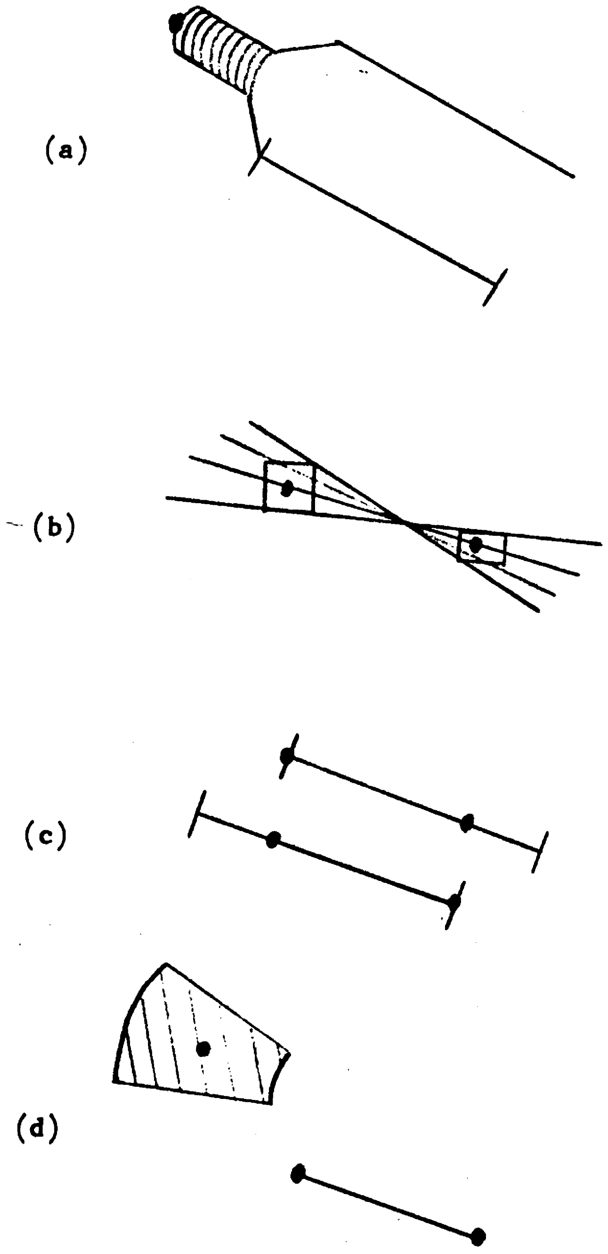


Figure 9.

region is considerably smaller than the tolerance region which would have been used within the basic system.

Notice that the reasoning done above assumes that the relative position of the end of the shaft with respect to the side is fixed. This is certainly true in 3-D, but in a 2-D picture this may not be the case. Some camera angles are worse than others. Thus the 'correct' way of making this implication is to work with a 3-D model. Unfortunately, that is considerably harder than a 2-D model. Therefore, the simple structure verification vision system only deals with 2-D models which approximate the 3-D situation. The open question is "when are 2-D models sufficient?"

Notice that the use of 2-D models for the tolerance reduction implications does not mean that everything is 2-D. After the features have been found, the final computation of the object's position is still carried out in 3-D (if necessary).

The use of extended features demonstrates an interesting trade-off between the ease of finding a featureand the amount of information provided by the feature. The difficulty in finding a feature is defined to be the amount of searching involved to locate it. A point feature such as a correlation operator is the hardest to find, but produces the most information (a point to point match). It is easier to find a point on a line segment, but less information is gained (one point is restricted to a line segment). It is easier still to locate a point in a region, but the larger the region the less information is gained about the location of the object. This trade-off doesn't mean that it is useless to find extended features. It just means that one of these features may not pin down the location of the object as well. Two or three may. And as shown in the example strategy for finding the shaft, extended features may be important stepping stones toward a final location.

So far this discussion assumes that there are operators which can locate a part of an extended feature. What operators are there and what is involved in using them?

The standard edge operator (eg. the Hueckel operator) can be used to locate a point on a line. Edge operators often return the angle of the line in addition to the coordinates of the point. This angle is important because it can be used to filter out bad matches (ie. the edge point is not within the expected 40 degree range) and it can help locate the line (ie. it is an estimate of the shaft's orientation).

The edge operator can also be used to locate points on a curve. Curves are particularly useful when they are known to be invariant (ie. their shape does not change throughout the range of possible images) or almost invariant. For example, the curve (ie. the ellipse) which is the image of' a large machined hole appears invariant if the only rotation is in the plane of the hole. For an invariant curve the angle returned by the edge operator can be used to locate a particular point (or set of points) on the curve with the matching slope. This means that an invariant curve is almost as good as a point operator even though it is extended, and hence easier to find. Unfortunately invariant curves are not as common as they might be.

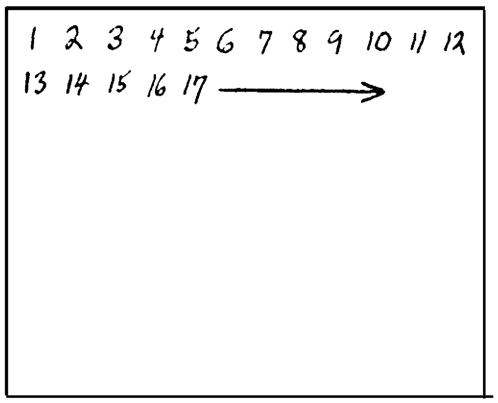
The standard point properties of regions (eg. intensity and color) can be used to locate a point in a homogeneous region and the standard texture operators can be used to locate points within textured regions. These are especially useful for constraining searches,

Any feature can be found by scanning the appropriate operator over the tolerance region for the feature. But one scanning technique may be better than others when looking for an extended feature. What types of searches are there and when should they be used?

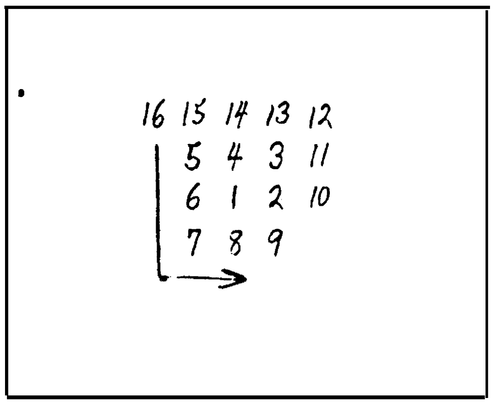
There are several types of scanning techniques: raster scan, spiral scan, linear scan, alternating linear scan (ie. start at one point on a line, try a point on one side, then a point on the other, etc.), random scan, etc. (see figure 10). The choice of scan depends upon the type of feature and how much is known about where it is expected to be. For example, if one is searching for a region, a random scan may be the technique to try. If the feature is a line, one might use a series of alternating linear scans that are perpendicular to the expected line. If there is an estimate of where the feature is and it is likely that the feature is close to this estimate, a spiral scan is probably the best choice. The upshot of this discussion is that the system should provide several different types of scans and a way of evaluating how effective they are.

Remember that the user is expected to write the program to do the verification. This means that he has to decide which feature to look for, what operator to use, where it should be tried, and what to do if nothing is found. In particular, this means that the user has to choose the scanning technique and fill in the details of where to start, which way to go first, and how far to go. In order to do this there should be an interactive subsystem for designing searches. The term 'system' may sound too impressive for such a seemingly small task, but the task isn't as small as dne might think. Such a system is essentially a graphics drawing program which can talk about tolerance regions, lines, angles, and all the parameters for the various techniques. It should provide a way of overlaying a proposed search on top of a picture and moving it around to see what might be encountered. There are two reasons for this overlaying: (1) to make sure that a scan is guaranteed to find one point on the feature and (2) to check for possible confusing matches.

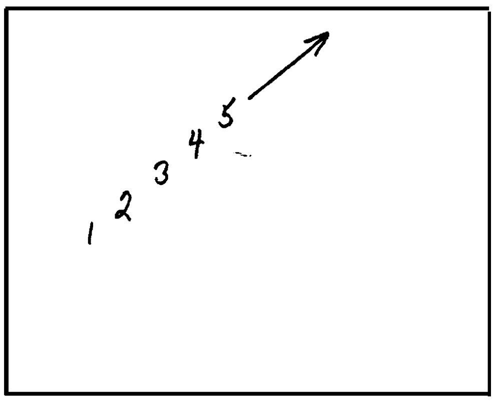
Consider figure 11. Figure 1 la shows a line segment feature and the tolerance region about its center. Figure 1 lb shows various positions of the line for different- apparent



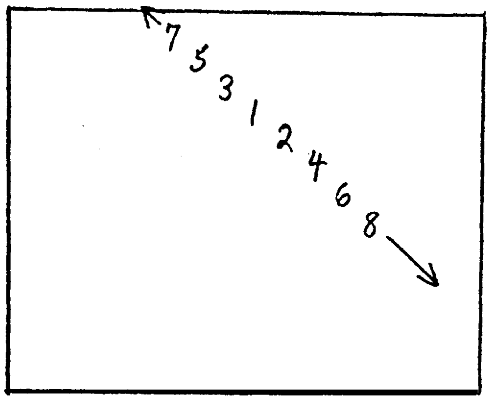
(a)



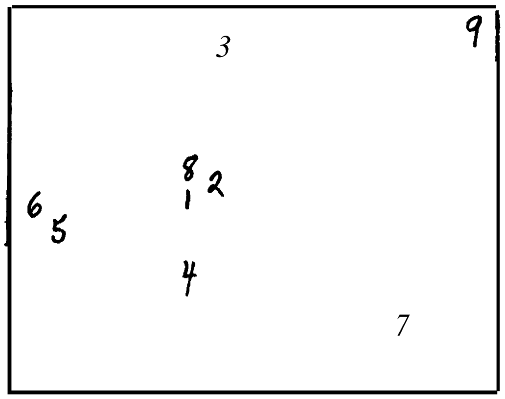
(b)



(c)



(d)



(e)

Figure 10.

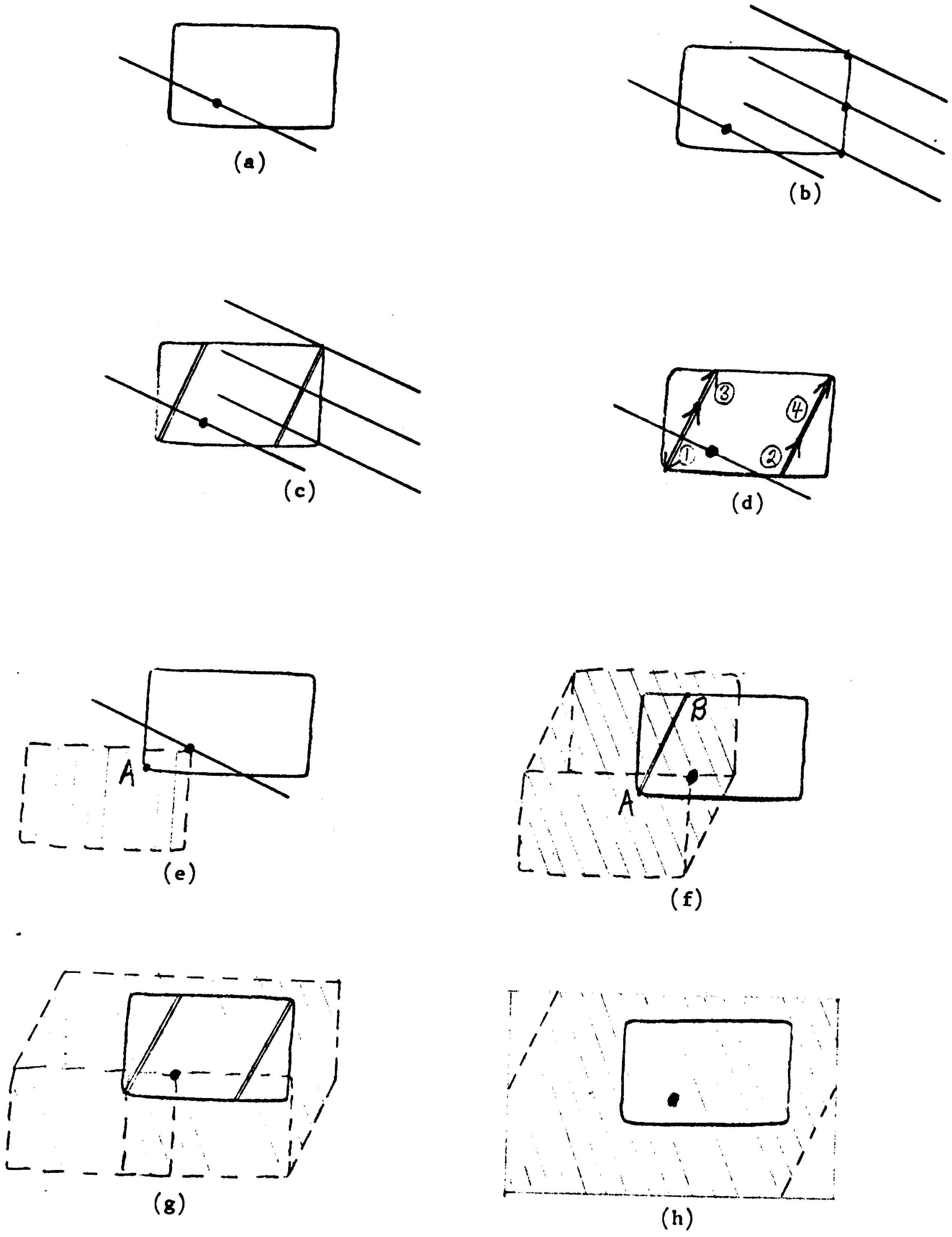


Figure 11.

positions for the center (remember that the center can wander around inside the tolerance region). Since the line segment is an extended feature a few linear scans are sufficient to guarantee one intersection with the line. The whole region does NOT have to be scanned. In this example the two scans shown in figure 1 lc are all that is needed. If the line is expected to be close to its planned position, it would be more efficient to break these lines up into an ordered set of smaller scans. One possibility is shown in figure 1 ld.

Intuitively it appears that there is a much smaller chance of matching an erroneous point if the operator is only scanned along these two lines than if it scans the whole region. But that is not true. The area which might contain erroneous matches is almost as large for the two linear scans as it is for the whole region. Figure 1 le shows the region of the picture which would be encountered at point A if the center of the line segment wanders over the whole region. Notice that A's region is sort of a left-to-right and top-to-bottom mirror image of the original region. Figure 1 lf shows the region of possible points encountered if the operator is scanned along the segment AB. And finally, figure 1 lg shows 'the total area which might be encountered along either linear scan. Notice in figure 1 lh that this area is almost the same size as the region used in the basic system.

Even after careful planning there may be ambiguous matches or the operators may find some small piece of the picture that they like even though it is not the 'correct' match. What can be done to insure that the correct matches are being made?

There are two different levels at which a feature can be checked: local and global. Local checking means that the portion of the picture near the possible match is checked for a structure which is consistent with the initial match. For example, if a line is being searched for and an edge operator has located one point on the line, the line can be followed (by the edge operator) to make sure that there really is a line there with the correct contrast across it and at the right angle. Similarly correlation patches can be increased in size or surrounded by several other small patches that match. Texture operators can grow larger regions about a possible point. Thus the confidence in a match can be increased by increasing the sire of the local match.

Global checking involves the use of the S-dimensional structure of the object being looked at and the constraints on that object to make sure that the features being matched are consistent with respect to each other. This 3-D checking can often be approximated by checking the 2-D consistency. For example, when trying to match a point on the lower side of a shaft it is possible to check a point by locating an edge point on the upper side. The position and angle of the upper can be predicted from the thickness of the shaft. If such a point is found one can be reasonably sure that the first operator is correctly matching a point on the lower side. In a fancier verification vision system these ideas-about confidence may be formalized into an automatic system of confidences, but keeping in line with the design of the 'simple structure' this type of checking has to be explicitly stated in the conditional statements of the program.

There are several thresholds associated with the various operators; such as the range of contrast, the confidence of the edge, and the range of colors. The operators also produce an answer within some precision. How can all of these parameters be determined?

The easiest way of deciding what the value should be for a certain threshold is to look at several training pictures (which hopefully 'cover the range of possibilities). The operators can be interactively applied on important portions of the picture and the range of contrasts, angles, etc. can be directly computed. For example, the precision of the edge operator's estimate for the slope of the line greatly depends upon the type of edge being looked at. The edge operator can be used to follow example edges in two or three pictures and its precision can be measured.

There is also a-theory about how to set the thresholds for certain operators such as edge operators (see [BINFORD]) and correlation operators (see [QUAM]). These should certainly be used when available.

Training sessions like these can also be used to determine how the location of one feature can help to locate another. After the two features have been located in several training pictures it is possible to set up a tolerance region about the implied position of one with respect to the other. Consider figure 12. Figures 12a through 12d show four different training pictures and the'locations of two features A and B. Having found A, these four examples imply that B would be at one of four places (as indicated in figure 12e). If we assume that these represent four extreme points in a connected region of possibilities, we can surround them with such a region (see figure 12f). The claim is that this region is the conditional tolerance region for B having found A. How correct this is depends upon whether or not the training 'pictures actually cover the range of possibilities.

Notice that programmable assembly provides an opportunity to have this type of training session. Other application areas may be able to provide training sessions, but not with such accurate details. For example, a training session for the task of navigating down a road may be conducted on one road, but not on all of the roads that the vehicle is supposed travel on. This restriction means that training sessions for such tasks can not possibly produce as specific results as in programmable assembly.

TASK: LOCATE THE SCREW ON THE END OF THE

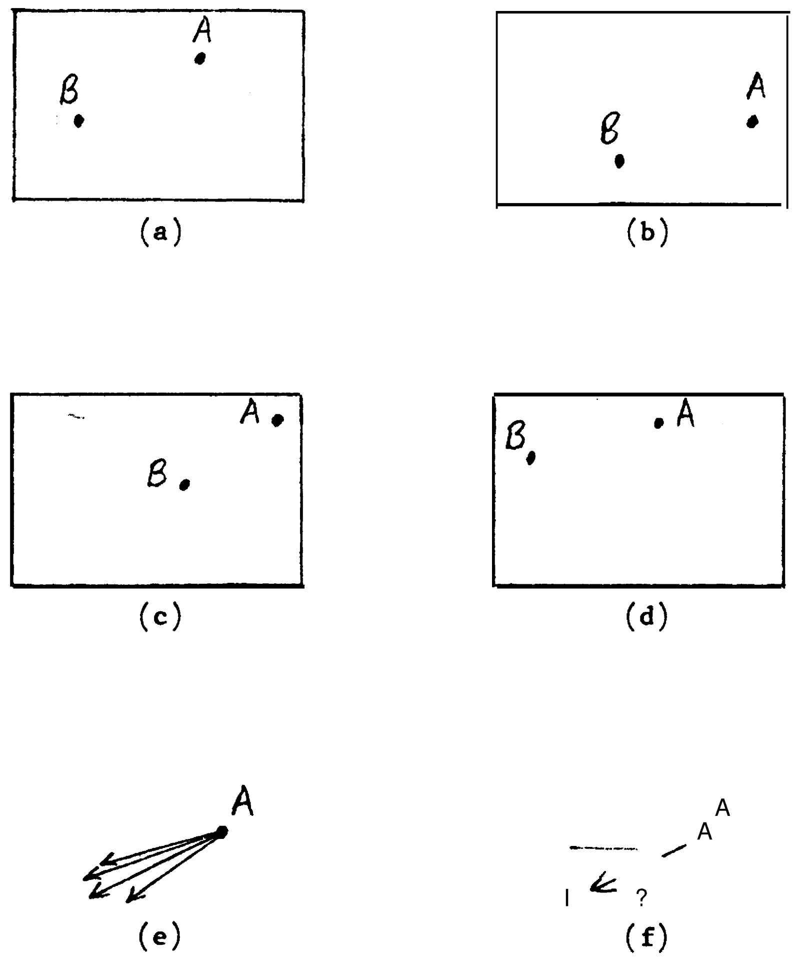


Figure 12.

SCREWDRIVER AND VISUALLY SERVO IT INTO THE HOLE --- ASSUME THAT THE HOLE HAS ALREADY BEEN LOCATED, AND THAT PROCESS PROCEEDS AS FOLLOWS: THE ARM MOVES AND STOPS, PICTURES ARE TAKEN, THE SCREW IS LOCATED, A CORRECTION IS DETERMINED, AND THE ARM MOVES AGAIN.

SOLUTION: SET UP ONE SPECIALLY SHAPED CORRELATION OPERATOR TO LOCATE THE SCREW FOR EACH OF THE STEREO CAMERAS. APPLY THESE AND USE THE STANDARD TRIANGULARIZATION TO COMPUTE THE SCREW'S 3-D LOCATION.

If the bachground is relatively complex, the correlation operator is restricted to the internal portion of the screw. Any part of the operator that stuck out might make the position of the match dependent upon what is in the background. This restriction is fine as long as the screw has enough internal information to produce a crisp match. If not, other information has to be used. Picture differencfng may help accentuate the change, but what other types of information are there?

There are two types of additional information: internal features of other objects rigidly affixed to the object of interest (eg. the screwdriver or hand) and boundary features which are formed by the interaction (or occlusion) of some part of the object which is moving and a part of the background.

The system described so far is powerful enough to take advantage of the other internal features, but what about the boundary features? A match of a boundary feature depends upon what is in the background next to the screw. Thus if a boundary feature is missed, the system should NOT assume that the screw is not there, but rather that the screw is currently in front of something that makes the boundary hard to see. The idea is that a boundary feature should be believed when it is located, but totally ignored if not. In some sense' it is an optional feature; it only contributes information if found. The simple structure system can certainly handle this type of feature. The programmers just need to be aware of it.

When stereo is being used, is there some way of using the locations of the features in one image to help locate them in the other image?

There is. Quam and Hannah have made extensive use of the well-known- idea that once a point has been located in one stereo view, the corresponding ray **can** be back-projected into the second view, and the feature must be on (or close to) this line (see [SOBEL], [QUAM] and [HANNAH]). The back-projected line, or actually a narrow region about the line, can be intersected with the normal tolerance range for the feature to produce **a** smaller region to be searched.

A similar benefit can be derived **from the** motion of the arm. After making the latest move along **the** path to the hole, the arm can be interrogated to find out where it thinks it moved. This positidn in S-space can be projected onto an image and a region **about that** point can be formed from an estimate of how precise the arm measurements are. The errors due to an inaccurate camera-to-arm calibration can be easily eliminated **by** considering the RELATIVE motion made by the arm from one point in the image to another.

The following is a summary of the semantic mechanisms required for the **'simple** structure' verification vision system.

A VARIETY OF "EXTENDED" FEATURES: LINES, CURVES, & REGIONS --- 2-D REPRESENTATIONS FOR THEM (<u>NOT</u> 3-D CURVED SURFACE MODELS . . . REMEMBER THAT THE BASIC ASSUMPTION OF THE SIMPLE STRUCTURE SYSTEM IS THAT 2-D FEATURES AND TOLERANCE IMPLICATIONS ARE SUFFICIENT . . . 3-D IS ONLY USED TO COMPUTE THE ACTUAL LOCATION OF AN OBJECT)

OPERATORS TO LOCATE PARTS OF THESE FEATURES ... **EG.** EDGE OPERATORS WHICH CAN LOCATE A POINT ON A LINE OR A CURVE, TEXTURE OPERATORS, ETC.

AN INTERACTIVE WAY OF DETERMINING THE VARIOUS THRESHOLDS AND LIMITS ASSOCIATED WITH THESE OPERATORS

SEVERAL SEARCH STRATEGIES TO CHOOSE FROM . . . **EG.** SPIRAL, LINEAR, & RANDOM

AN INTERACTIVE WAY OF SETTING UP AND EVALUATING SEARCH STRATEGIES TO LOCATE A

PARTICULAR FEATURE

METHODS TO DO LOCAL CHECKING ABOUT EDGE POINTS, CORRELATIONS, AND REGION POINTS

A 2-D SYSTEM FOR PREDICTING THE RANGE OF POSITIONS FOR A FEATURE ONCE ANOTHER FEATURE HAS BEEN FOUND

A FORM FOR VERIFICATION VISION PROGRAMS'

Notice that the simple structure system is designed around an interactive training session. In one session the user can do everything necessary to program a verification vision task:

(1) set up example assemblies
(2) take pictures
(3) define features by interactively drawing them on top of an example picture
(4) decide what operators to use and interactively set their thresholds
(5) determine the tolerance region about a feature point
(6) design a search to locate a point on a feature
(7) check for undesirable matches
(8) decide upon the amount of local checking to be used
(9) set up the 2-D conditional implications from one feature to another

and (10) write the program which uses all of these pieces.

This type of interactive system is demonstrated in the protocols that follow.

(1) LOCATE THE SHAFT (ITS END AND ITS ORIENTATION)

(a) The user decides that the 40 degree range can NOT be handled directly by correlation. How much of 'this range can be reliably handled by correlation?
(b) Take several picture8 uith the shaft at different angles. The glare on the shaft and the shadows increase the change in appearance from one position to the next.
(c) Try several correlation operators to determine the size of the subranges. Assume that the range can be safely divided up into three

slightly overlapping ranges.

(d) The user decides to try to locate the lower edge of the shaft and use that to decide which sub-range is appropriate. What is required to determine the angle of the shaft well enough to choose the right sub-range? Is one point on the side enough? two points?

(e) Define the line feature which is associated with the bottom of the shaft. This can be done by pointing out the two ends of the segment in one of the training pictures.

(f) Set the thresholds for the edge operator so that it accepts almost all edges and use it to follow the line. Possibly follow the side in three or four training pictures.

(g) Gather statistics on the actual values for the contrast, confidence, etc. and use them to set tighter (ie. more discriminating) thresholds for the operator. Fit a line through all of the points found on the segment. Compare the slope of this line with the estimates from the operator and compute the precision of the edge operator's estimate of the slope.

(h) Project the tolerance region for the midpoint of the line onto the picture of the shaft at its planned position. Set up a search technique which guarantees one point on the side.

(i) Check for possible confusing points and plan for disambiguation (possibly by following the line, or by using a second edge operator to find a point on the other side of the shaft). Assume that one point on the side is enough, ie. the operator's estimate of the angle is good enough to decide which of the three situations the shaft is in. Notice that shaft's orientation may still not be determined well enough to meet the goal of ±2 degrees.

(j) Each of the three situation8 is a straightforward problem of applying the correlation operators and determining the best estimate for the shaft's position and orientation, But there is more information that could be used. When the edge operator locates a point on the side of the shaft (or two such points are combined to determine the apparent angle of the shaft) there is eome precision associated with that computation, That precision may indicate that the angle is known to within ±3 degrees; If that is the case, the tolerance region surrounding the end of the shaft could be determined with a total angle uncertainty of 6 degrees instead of 14. It is not clear whether or not that is a significant reduction, but it might be.

(k) The program would be: Apply the edge operator along the predetermined search' path. When it locates an appropriate edge point, check it by following the edge or whatever was decided. If it isn't the correct one, continue along the search path. If no edge point is located, complain to the human operator. When a good edge point is located, use its estimate for the angle to choose one of the three subproblems. Use the precision of the angle and a 2-D model of the line segment to produce a region of possible location8 of the end of the shaft. Locate the matching correlation points and compute the shaft's position and orientation. flake sure that the values are within the desired tolerances. If not, complain to the human operator.

(1) LOCATE THE SCREW ON THE END OF THE SCREWDRIVER AND VISUALLY SERVO IT INTO THE HOLE

(a) Assume that stereo is going to be used to determine the relative displacement of the screw tip from the hole. Stereo has already located the hole and told the arm to correct accordingly. Where do you look to find the screw? This can be treated I i'ke a 2-D conditional tolerance implication. Set up an example assembly, locate the hole, and locate the tip of the screw. Do this for a few different situations and combine the relative positions of the tip from the hole into a 2-D region which covers the range of possibilities, , In order to do this, however, there has to be a way of locating the screw tip.

(b) Assume that the screw is not very distinct. That is, the correlation operator has narrow tolerances and even if it finds a match, the resulting precision is low. If the screw is in front of a background which makes the outline quite distinct, the user can set up a few correlation operators_ to key off of the outline. During execution, if they are successful, their results are used. Otherwise, the correlation on the screw is used as the last resort.

(c) Another possibility is to use an edge operator to locate a point on the side of the screwdriver. Assume that the boundary correlations are sufficient so that this is not necessary.

(d) After locating the screw once, use the portions of the picture matched by the correlation operators as the basis for future correlations. These new operators should be even better correlation operators than the ones set up during the training phase because they are based upon the way the scene actually appears during this particular assembly. Each assembly may have slightly different objects, object positions, lighting and camera calibrations. Extracting information for future correlations assumes, of course, that you are sure that you know what you have matched. It would be unfortunate to locate the screw incorrectly and then extract 'good' correlation patches based on that match.

(e) Another point: since you are tracking the screw (ie. looking at it every .2 cm or something) it should not be disastrous to miss it once in a while. Sometimes the background is going to be bad and sometimes the operators are going to miss things.

(f) There should also be a special check for termination. Often the background and local changes in the appearance of a screw (or any part) are most pronounced when it is approaching the goal, Therefore, with the screw, when the tracking indicates that the screw is close to the hole, the key feature should be shifted away from the tip to some point near the top of the screw.

(g) The program might be:

(A) LOCATE THE HOLE . . . usual stereo training etc.

(B) LOCATE THE SCREW THE FIRST TINE . . . plan to locate a point on each side of the screwdriver and then correlate on the screw and the boundaries of the screw . . , this location is especially important because the rest of the tracking will use correlation patches derived from this picture.

(C) TRACK THE SCREW UNTIL IT IS ONE CM OVER THE HOLE . . . use the arm's estimate of how far it has traveled to predict the position of the screw. Try the boundary correlations first. If they are found check for global consistency . . . ie. that they are in the correct relative positions (within 'tolerances). If the screw is found in one , of the stereo views, backproject its position into the other view and use that to compute the prediction of where it is. If it is found in both views compute its relative 3-D position with respect to the hole and decide the next move of the arm. If 'the screw is not found, continue to move the arm in the same direction etc. as last time. If the screw is lost for more than two successive times or it is getting too close to the hole, stop. It would also be possible to stop the arm and concentrate on re-acquiring the screw.

(D) TRACK THE TOP OF THE SCREW UNTIL THE TIP CAN BE IMPLIED TO BE IN THE HOLE . . . this just means start the location process by looking for features that are near the top and ars less likely to be altered by being near the hole.

A FANCIER SYSTEM

The aim of this 'fancier' verification vision system is to (1) reduce the amount of work required of the user to accomplish a task and (2) increase the precision and reliability of the final result. The simple structure system provides interactive tools so the user can conveniently try out different operators and approaches. However, all of the decisions about how good an operator is or what operator to try next are left up to the user. The fancier system tries to automate some of these decisions. For example, instead of requiring the user to point out the good features, the system tries to suggest and locate good features on its own. This section presents a list of potentially automated subtasks and discusses some of the key implementation issues.

SUGGEST GOOD FEATURES

Probably the easiest way of automatically determining 'good' operators is to scan an 'interest' operator over a training picture. This is often done to find good correlation points. The interest operator tries to determine how distinct the local region is and estimate how well a correlation patch would work there. Quam, Hannah, and Moravec all have their favorite interest operators for correlation (see [QUAM] and [HANNAH]). They range from variance operators to simple corner operators and from the analysis of autocorrelation characteristics to an analysis of the directional information. But they all produce the same result: a list of 'good' correlation points to be used to locate corresponding, points in a new picture of the Scene.

Notice that these correlation points are NOT necessarily associated with parts of an object or points in, a model of an object. They are simply visually distinct portions of the picture. The rest of the system has to know what to do with the matches after they have been found. If the task is to navigate down a road, the matches could be used to determine how far the vehicle has moved from one picture to the next (assuming that the world is static and that the apparent change in the position of the points is due to the' vehicle's motion). Within the programmable assembly environment the user may want to identify each 'good' correlation operator with the corresponding point on the model of the object, ie. the point on the object that appears in the picture at the center of the correlation patch. In this way, after the matching correlation points have been found, the system would know what parts of the object have been located and thus be able to compute a new estimate for the object's position.

It would be possible to do something similar to find good 'extended' features such as lines or regions, but it might require much too much work to find a 'good' long line by checking every possible match that an edge operator might find in a picture. It makes much better sense to start with some idea of what feature will-appear and where. Then it is a matter of locating the feature and checking it out. But there is a catch: where do the predictions of good features come from? They come from the model of the objects. Thus, instead of scanning a training picture, the interest operator could scan a synthetic picture of the expected scene and suggest features to be considered. The synthetic picture could be simply a hidden-line view of the scene or a complete, synthetic color picture. Thus, to find good line features an operator might scan the line drawing for lines of a certain length and then check the expected contrast across the edge by looking at the corresponding point in the synthetic grey-scale picture. Similarly, corners in the line drawing could be suggested as good points for correlation.

The process of finding good features to be used in scanning real scenes can be characterized as follows:

(a) Build a model of the objects
(b) Place the real objects at their planned position on the table
(c) Take a training picture
(d) Symbolically place the models at their planned position with respect to the camera
(e) Produce the expected hidden-line view of the scene and the complete, synthetic picture
(f) Have the interest operators wander around the line drawing and synthetic picture picking out potentially good features
(g) Locate the feature in the training picture
(h) Determine the thresholds for the operator (from the actual picture data)

and (i) Decide whether or not the feature is good enough

The success and generality of this approach depend upon several capabilities: the modelling system, the hidden-line procedure, the synthetic picture generator, the ability to locate a suggested feature, and the method of describing interesting features. Each one of these' tasks is a formidable task indeed. There are partial solutions to all of them. As better solutions are found they can be incorporated into the system. Until then the user can take up the slack. The user will have to be around for a while anyway to make sure that the process is proceeding as planned. In particular she may have to make sure that the right features are being located to match the ones suggested by the automatic system.

Even though the objects are supposed to be in a planned position there are several reasons why the synthetic picture may be incorrect: an incorrect calibration of the-camera to the table, inadequacies in the light model which produces the expected brightnesses, an incorrect placement of the object, slight variations in the object with respect to model, and noise. Thus, step (g) is a verification problem itself. The only difference between it and the original problem is that the positions of the objects should be better known (since the object is at its planned position). The result of steps (g) and (h) can be thought of as a secondary calibration of the camera and the synthetic picture generator. These steps determine the final corrections for the position and appearance of an object.

Many of the objects which appear in programmable assembly tasks are composed of machined or cast parts. Cylindrical components (eg. shafts and holes) are common. Cylindrical components. are important because the angular uncertainties of an object are often aligned with the axis of one its cylinders and this means that the image of the cylinder will contain an invariant curve (ie. an ellipse). Recall that invariant curves are convenient features for verification vision. The point is that in order to predict curves as features the modelling system has to be able to model curved surfaces.

There are various systems for representing curved surfaces (see computer-aided design articles), but they are probably too complex for this type of system. There are, however, a few simpler ways of including curves. One way is to extend the model to allow cylindrical surfaces in addition to the usual planar surfaces. Unfortunately the hidden-line algorithms do not handle cylindrical parts directly. A possible way around this is to have the system maintain a symbolic model of an object which associates a type with each component. Whenever the hidden,-line algorithm is needed, the cylindrical parts can be approximated by several planar facets. If the algorithm keeps track of where the various points and lines in the predicted image come from, it might end up with a series of points that all belong to the end of a cylinder. An ellipse can be fitted through these points to produce a reasonably accurate 2-D image of the end of the cylinder. The resulting ellipse can be used as a feature. Notice that this approximation process is NOT limited to cylinders and ellipses. As long as the hidden-line algorithm can identify a series of points that belong on a smooth, connected curve, it would be possible to spline them together to produce a reasonably accurate estimate of how the real curve would appear in the picture.

The upshot of this section is that it is possible for the system to predict and locate features itself.

SEARCH PATTERNS

The basic system included a subsystem which could produce the tolerance region about a feature point. That is, it could outline the portion of the screen where the feature might appear. In order to find the feature this region would be searched. As mentioned earlier there are several techniques for searching such a region. The choice of **which technique or** combination of **techniques** to use in any particular situation is relatively complex. It **depends upon** the type of feature **being** looked for, the size of the feature, the expected distribution of appearances in the region, the cost of generating the next trial position, and the size **and shape** of the region. This choice is especially important for extended features **because their main** potential advantage is that they are larger and supposedly easier to find.

Consider the case that the tolerance regions are rectangular (as shown in figure 13). Figure **13a** shows a line segment and the tolerance region about its center. The goal is to design an efficient search strategy to find a point on the segment. First notice that a search that is restricted to the rectangle must include two of the corners (see figure **13b)** because they are **the only** points on the segment that intersect the rectangle. Also notice **that the** 'extendedness' of line segment is maximized when the search is perpendicular to the **segment.** Keeping these two ideas in mind a reasonable start might be the linear search shown in figure **13c.** The dashed region indicates the portion of the screen where the center of the segment. could be and-still have this search intersect the segment. Figure **13d** shows the **results** after adding a similar search from the other critical corner. Figure **13e** includes a third search to cover most of the middle. Unfortunately there are several **small areas which are still** not covered. That is, if the center of the segment happens to be in one of them, the three searches suggested so far will NOT find a point on the segment. One solution is to **add** several short searches as shown in figure **13f.** Another solution is to forget about the restriction of staying within the rectangle and extend the existing three searches to cover the' **small** areas. This is shown in figure **13g.** Notice, however, that the region of **possible** confusions should be based upon the larger, dashed region.

Figure 14 **shows a** very **simple method** for automatically generating a **reasonable** search. The expected orientation of the segment **is used** to decide whether horizontal or vertical **scans** are more efficient and then **a** series of these are pieced together to cover the **whole** region, If one **assumes that** the closer a point is to the expected position of the segment **the, higher** the probability is that the segment is there, the searches can the ordered by their distance from the expected position of the center of the segment (see figure **14f).**

Some curve segments can be treated in a similar manner. Figure **15a shows such a** segment. The maximum chord of the segment and its perpendicular bisector **are shown in** figure **15b.** The tolerance region is about point A. Figure **15c** shows the portion **of the** screen that is covered by the vertical search. Figure **15d** shows the suggested search.

There are similar, crude **methods** for deciding where **one should look** to find a point in a region. Figure 16 shows one possibility. Figure **16b** shows the largest inscribed rectangle

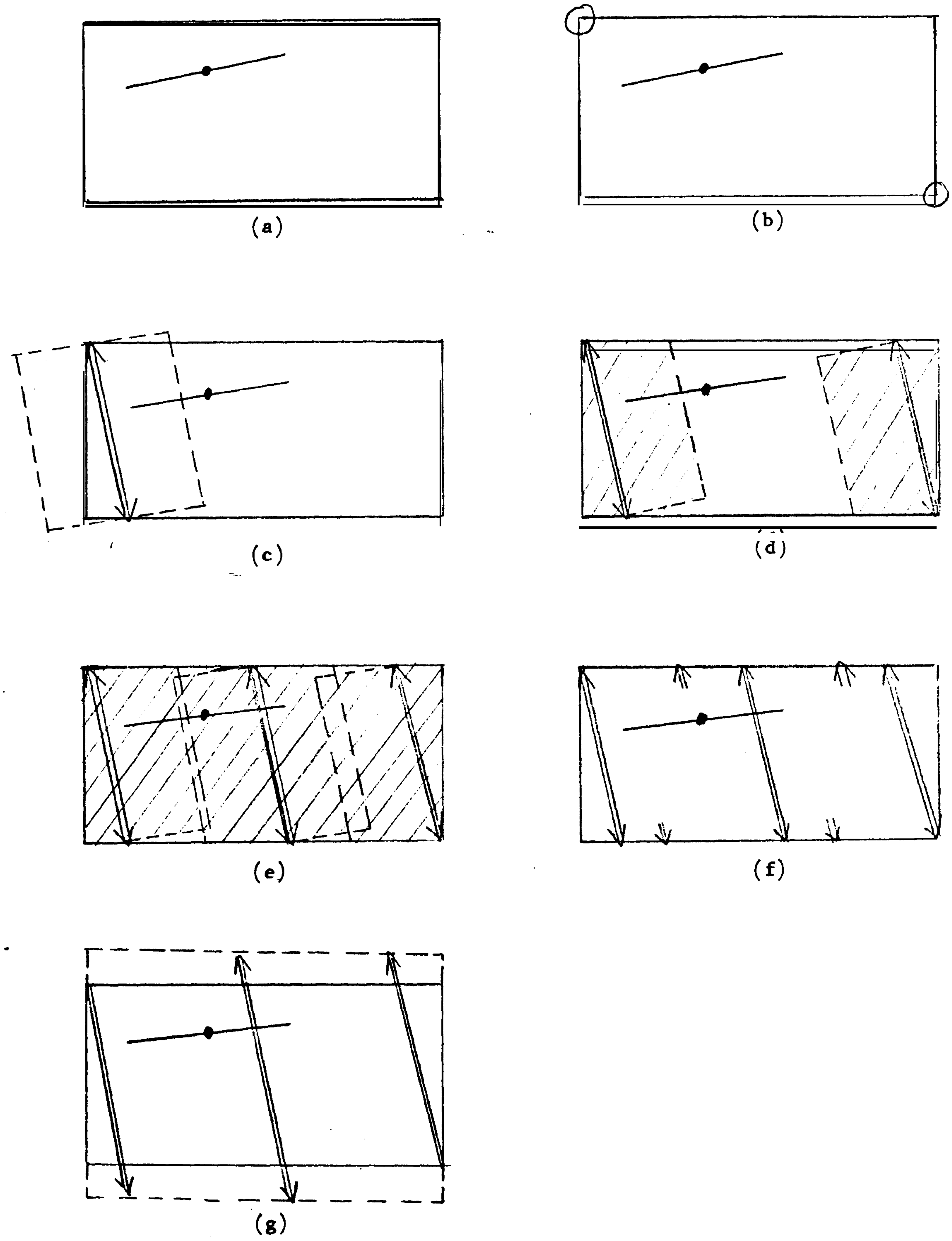


Figure 13.

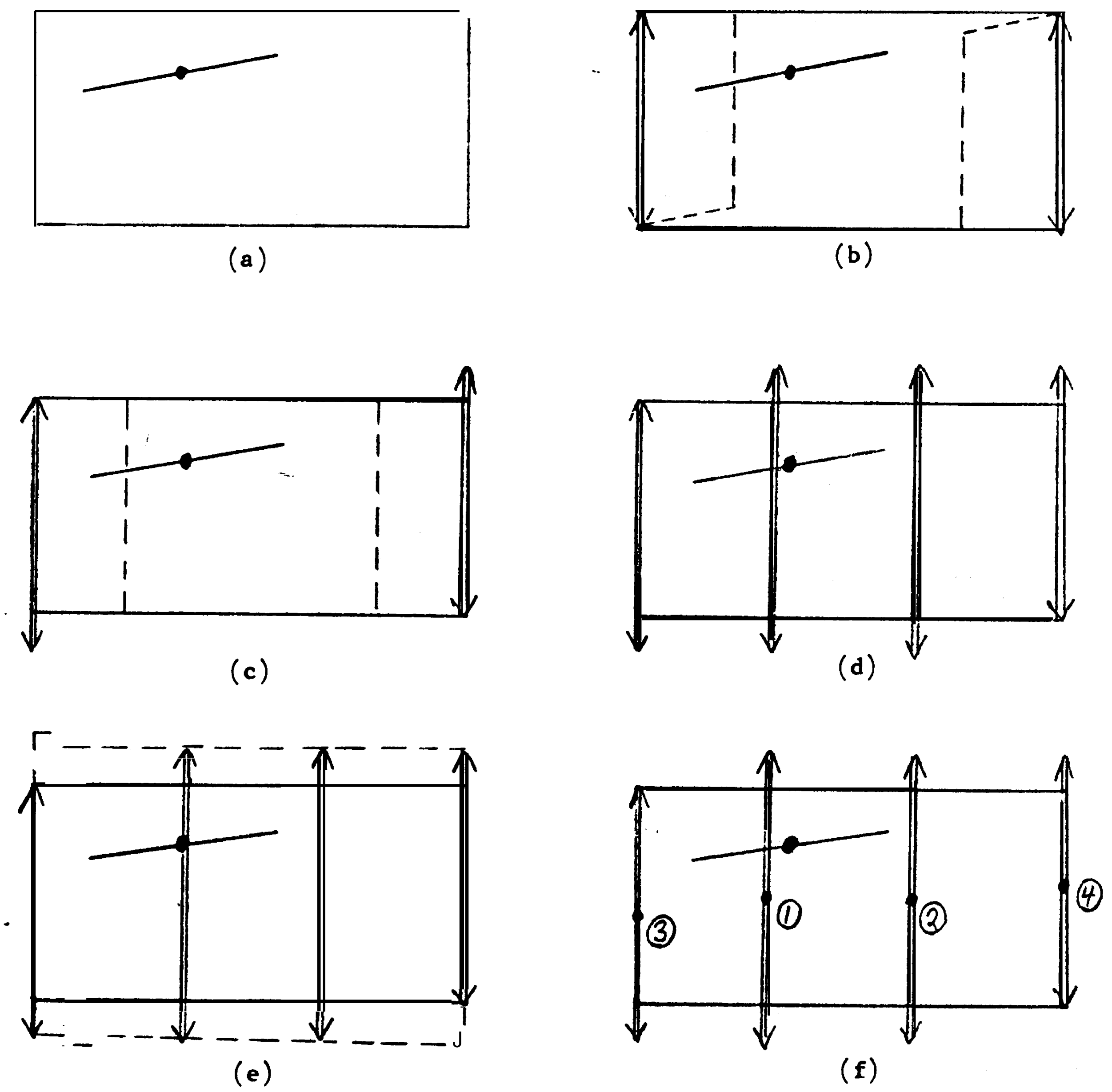
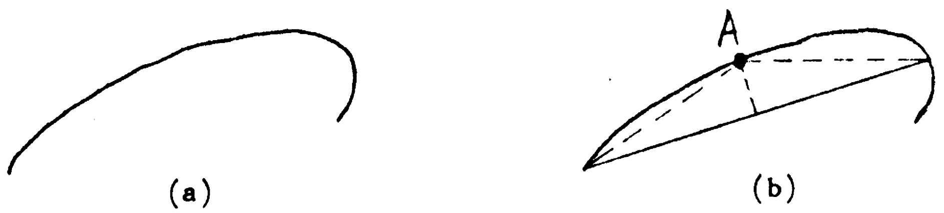
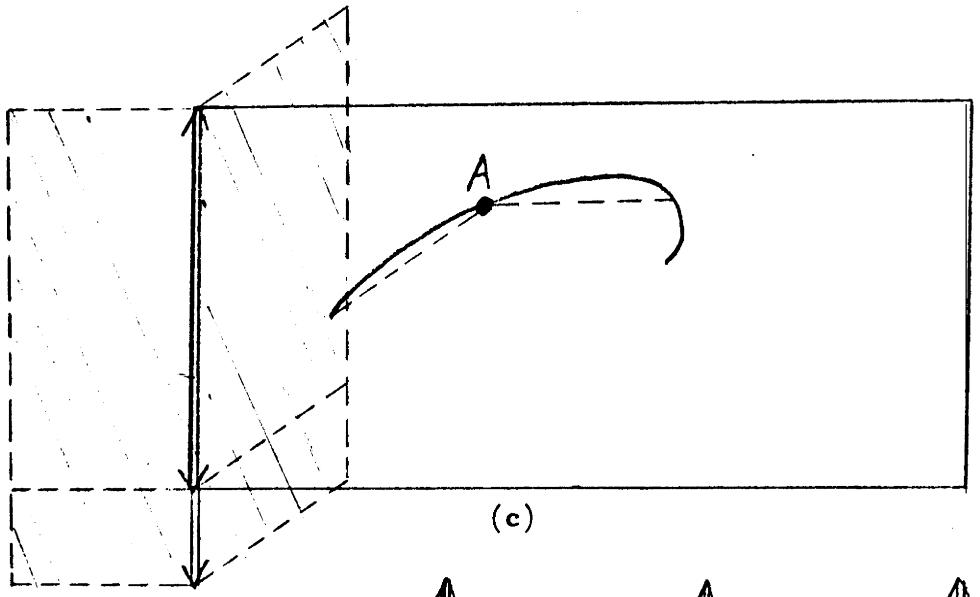


Figure 14.





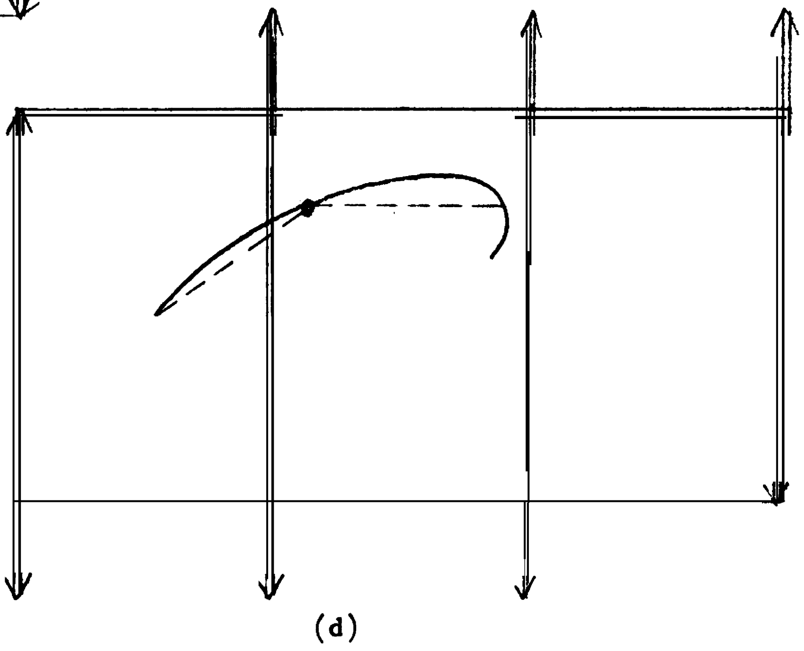


Figure 15.

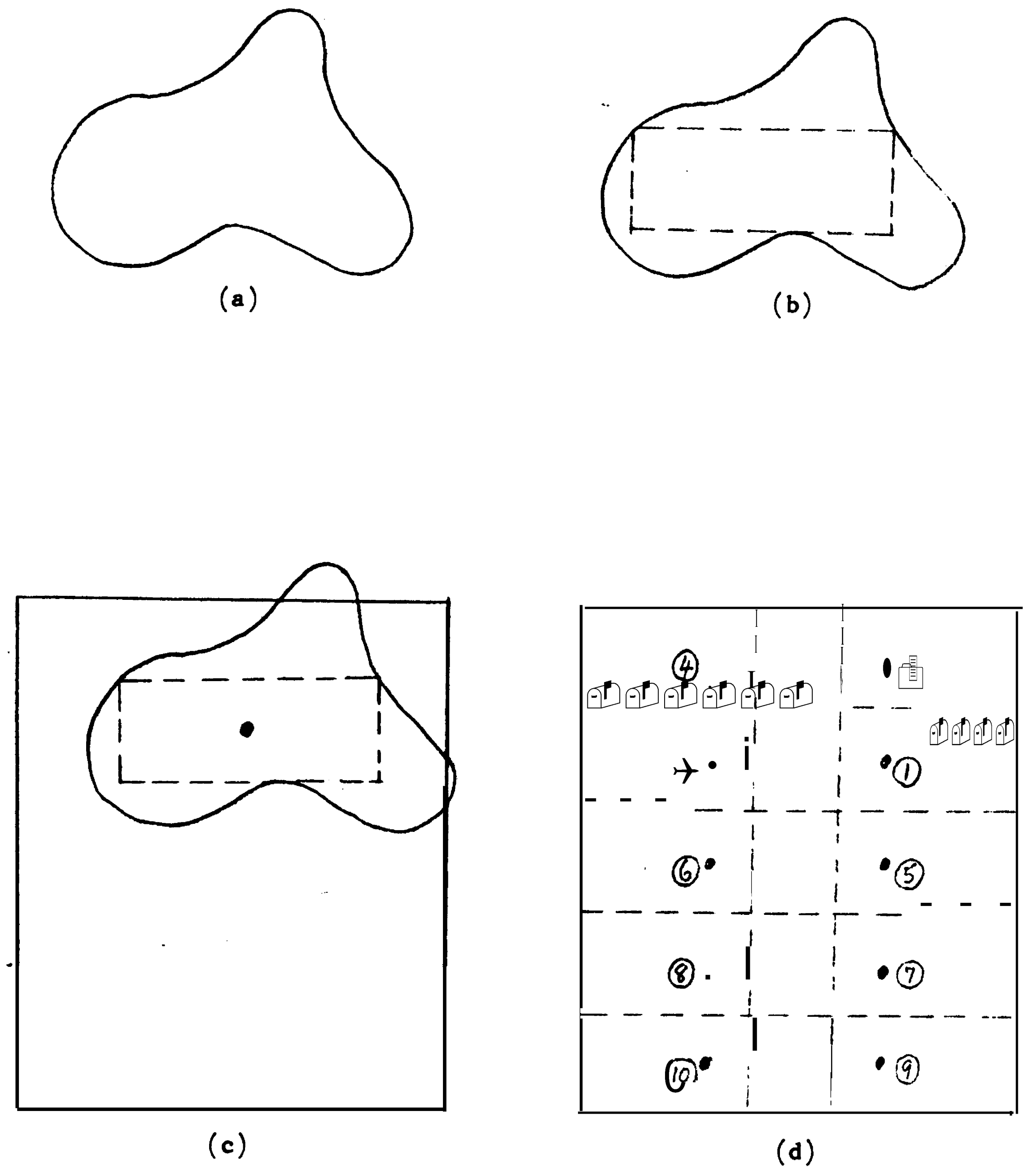


Figure 16.

within the region. The center of the rectangle is used as the feature about which a tolerance region is constructed (see figure 16c). The tolerance region is simply 'tiled over' with these rectangles and their centers are ordered to form a search (see figure 16d).

These techniques assume that the major effect of the uncertainties on the object is translational. Any effects due to angular uncertainties can be covered by checking for the least beneficial orientation of the segment and using an appropriately conservative estimate for the portion of the screen covered by one linear scan.

The important point of this section is that there are ways for the system to automatically set up its own search techniques.

CHARACTERIZE THE BENEFIT OF LOCATING A FEATURE

There are two main benefits of locating a feature: (1) a decrease in the uncertainty about the object's position and (2) an increase in the confidence that the correct features are being located. The basic system and the simple structure system concentrated on the first. The user was responsible for the second. The earlier systems provided a unified system of tolerances and tools for acquiring the necessary information. There was no similar system for confidences. The user had to decide for himself whether the features were consistent or not and whether another feature should be located just to make sure.

Even though the earlier systems provided tools for gathering tolerance information, they did NOT automatically determine the parameters required by the tools. For example, the simple structure system did not automatically decide how much tolerance information is gained about one feature by locating another feature. The user had to decide what the extreme cases were and then combine the range of possibilities into an implied tolerance region for feature two from feature one. This process is a candidate for automation. It essentially requires a method of representing a range of scenes, in particular, the range of scenes which are possible, given a set of constraints on the objects in a scene. This is rather difficult. It can be approximated by a method which decides the values of the constraints which determine the extremes of a tolerance region and an assumption that the scenes change smoothly from one extreme to the next. The synthetic scenes which correspond to the extremes could be generated and analyzed to produce the implication tolerances from one feature to the next.

Notice, however, that this is still an approximation. It is quite different from the following 'optimum' process:

(1) Combine the current constraints on the position of the object to

produce the expected tolerance region about the next feature to be looked for.

(2) Locate the feature or part of the feature.

(3) Use the location information to produce another constraint on the position of the object. For example, an edge point on a line should produce a constraint which says something like: edge such-and-such of the object must intersect the 3-D ray which starts at the lens center and passes through the appropriate point in the image plane, and the edge must project into a line with an orientation of X ± y. In fact, instead of intersecting a ray, the constraint should really be an intersection with a narrow cone centered about the ray and whose width is determined by the position uncertainty of the edge operator.

(4) Use the expanded list of constraints to produce the tolerance region about the next feature, etc.

Unfortunately, this requires a very sophisticated constraint system.

In order to automate the concept of confidence a unified system of confidences would have to be set up in such a way that each operation on a picture would be accompanied by an appropriate confidence computation. Each attempt at locating a feature would cause a reaction within the tolerance system and a reaction within the confidence system. Such a confidence system would require each operator to report its degree of certainty that it found what it was looking for. This information could be integrated with the position information to decide the consistency of a set of features and even possibly indicate which feature is the least consistent if the whole set appears to be inconsistent.

A NETWORK OF FEATURES INSTEAD OF AN EXPLICIT PROGRAM

So far the system has been provided with tools for automatically choosing potential features, setting the operators' thresholds, determining the expected reduction in tolerances, and increasing the confidence in the location process. There is one major area left which needs to be incorporated before the system can automatically decide which feature to look for next, This is the cost information. If the system could predict the expected cost of a search, it could carry out a complete cost/benefit analysis to determine what to do next.

One simple approach to cost is to equate the cost of an operation with the amount of computer time required to do the operation. Thus, in order to decide the expected cost of a search for a feature the system would have to be able to determine the expected number of tries and the cost per try. This is relatively straightforward.

A more complete strategist would have to take into account the amount of core required by the various operators, the amount of time spent in the strategy module, the expected amount of real time (for focusing or changing lenses), etc. Feldman and Sproull have recently made an interesting formulation of this problem (see [FELDMAN]).

Notice that once the system can decide what to do next, there is no longer any need for an explicit program. The verification vision program reduces to a network of features and the system takes the form of an interpreter which looks at the network of features and decides what to do. For example, the interpreter might decide that it needs more position information and so it suggests locating a point on the bottom of the shaft, or it may decide that it needs to boost the overall confidence, so it suggests locating a point on the other side of the shaft. Another possibility would be to invoke the strategist in such a way that it 'compiles' a program from one of these networks. The program would be set up to handle explicitly the various situations which might arise, just like the user's program was supposed to do within the simple structure system. The strategist would have to be able to simulate different situation% and construct a plan which covered a range of possibilities.

A SYSTEM FOR DESCRIBING FEATURES

Ideally there should be language for describing new operators, their costs, weaknesses, what types of features they find, etc. In this way whenever a new operator has been perfected it could be easily added to the system. A similar facility should exist for all parts of the system, including features and searches. This requires a higher level of understanding, It is one thing to be able to use various operators. It is something else to be able to systematize their properties in such a way that new operators can be completely described within the system.

A SUMMARY OF THE FACILITIES NEEDED TO IMPLEMENT THESE IDEAS:

, A 3-D MODELLING SYSTEM WHICH INCLUDES SURFACE INFORMATION SUCH AS REFLECTANCE . . . IT SHOULD ALSO BE ABLE TO MODEL SOME CURVED SURFACES, EVEN IF THEY HAVE TO BE HANDLED INDIRECTLY

A LIGHT MODEL . . . IE. A POSITION AND INTENSITY OF THE LIGHT SOURCE

A HIDDEN-LINE ELIMINATION METHO

A

A CURVE FITTING ROUTINE . . . EG. SPLINE PACKAGE

A SYNTHETIC GREY-SCALED PICTURE GENERATION METHOD

A SET OF 'INTEREST' OPERATORS TO SCAN THE WIRE-DIAGRAM PICTURES AND SYNTHETIC PICTURES IN ORDER TO LOCATE **POTENTIALLY** USEFUL FEATURES

A METHOD FOR AUTOMATICALLY **SETTING UP A** SEARCH PATTERN

A REPRESENTATION FOR A RANGE OF SCENES

A METHOD FOR AUTOMATICALLY DETERMINING 'IMPLICATION REGIONS' FROM ONE FEATURE TO ANOTHER

A METHOD TO DETERMINE THE CONSTRAINTS THAT APPLY AT THE EXTREMES **OF A TOLERANCE** REGION

A SOPHISTICATED CONSTRAINT LANGUAGE AND RESOLVING SYSTEM

A SYSTEM OF CONFIDENCES

A SYSTEM OF COSTS

A NETWORK' OF FEATURES (INSTEAD OF AN EXPLICIT PROGRAM)

AN INTERPRETER WHICH CAN DO **A COST/BENEFIT ANALYSIS TO DETERMINE WHAT** SHOULD BE DONE NEXT

A METHOD TO CONVERT A NETWORK OF FEATURES INTO A COMPILED PROGRAM WHICH HANDLES THE NECESSARY RANGE OF POSSIBILITIES

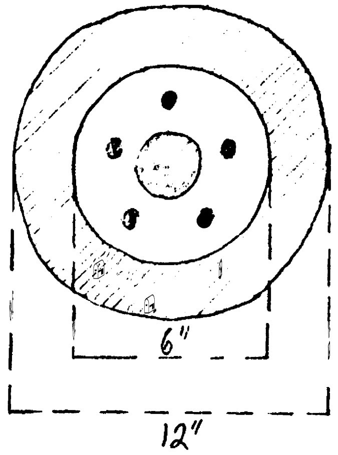
A DESCRIPTIVE SYSTEM FOR OPERATORS, FEATURES, SEARCHES, ETC.

An example protocol:

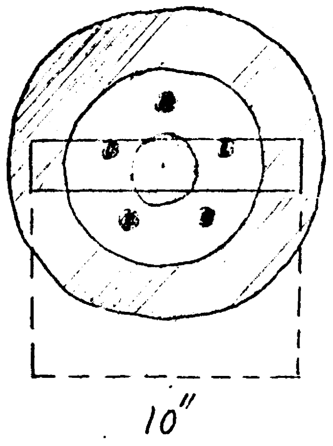
TASK: LOCATE A WHEEL HUB (SEE FIGURE 17A) --- ASSUME THAT THE HUB IS THE REAR WHEEL HUB ON A CAR MOVING DOWN AN ASSEMBLY LINE. THERE IS A TRIP SWITCH THAT TRIGGERS THE CAMERA FOR EACH CAR ON THE LINE. HOWEVER, THE SWITCH IS ONLY ACCURATE TO WITHIN ±5 INCHES (IE. THE POSITION OF THE HUB ALONG THE ASSEMBLY LINE IS KNOWN ONLY TO WITHIN ±5 INCHES WHEN THE PICTURE IS TAKEN). THE PLANE OF THE HUB IS KNOWN BECAUSE THE CARS ARE ALL POSITIONED ON THE LINE THE SAME.

GOAL: LOCATE THE CENTER OF THE HUB TO WITHIN ±1/10th INCH AND DETERMINE THE ROTATION ABOUT THE CENTER TO WITHIN ±2 DEGREES --- ASSUME THAT THESE ARE THE REQUIREMENTS NEEDED TO ASSEMBLE THE WHEEL ONTO THE HUB. GIVEN THE TIME THAT THE PICTURE WAS TAKEN, THE SPEED OF THE LINE, AND THE POSITION OF THE HUB IN THE PICTURE, THE SYSTEM CAN FIGURE OUT WHERE THE ARM MUST GO TO TRACK THE HUB AND ASSEMBLE THE WHEEL.

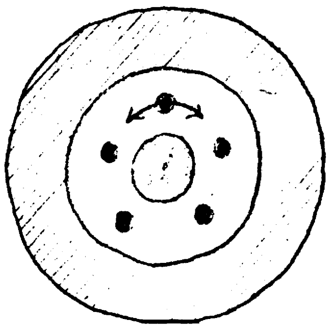
The first subtask is to determine the position of the camera and check the potential resolution. The camera must have a wide enough view of the scene to see several features no matter where the hub may be (within its constraints) and yet the resolution of the individual



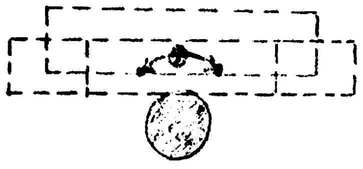
(a)



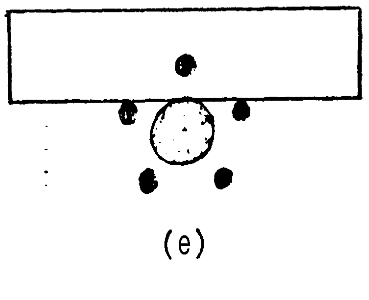
(b)



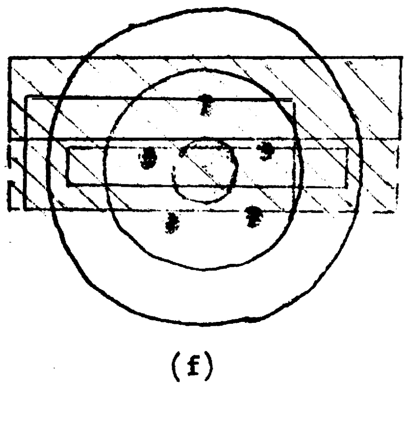
(c)



(d)



(e)



(f)

Figure 17.

pixels must be great enough to produce the desired precision of ±1/ 10th inch. If we assume that the operators and computations are precise enough to locate a point in an image to within 1/2 a pixel, the resolution of one pixel must be at least 1/5th inch.

The next question is how much of the scene should be in view. There are two steps involved in answering this: (1) what features should be in view and (2) what is the portion of the scene that includes the 'union' of their tolerance regions. To answer these questions a model of the object should be built (eg. see figure 18) and the constraints on the object should be stated (eg. the plane of the hub is parallel to the XZ plane of the work station, the rotation axis of the object is parallel to the Y axis of the work station, the center of the shaft may, fluctuate along the X-axis by ±5 inches and along the Z-axis (of the work station) by ±1 inch). The rotation constraint can be reduced to ±36 degrees without lose of generality because of the symmetry of the five bolts. The user can then point out portions of the model that should be' seen (eg. the center of the shaft, a couple of lug bolts, and a part of the medium-sized curve). This' is just a rough indication of what should be in the picture. The automatic system will later decide which features are actually needed.

After the features have been pointed out the system can produce the tolerance regions about them. The tolerance region for the center of the shaft is shown in figure 17b. Figures 17c through 17e develop the tolerance region about the top bolt. All of the tolerance regions can be combined to produce the total region which should be in view (see figure 17f). In order to cover this region which is approximately 16" by 4" and still achieve the necessary precision, the image. must be at least 320 pixels by 80 pixels. If such a camera is available, everything is fine. The position of the camera can be computed from this information.

However, if the only cameras available have 200 by 200 images, two of them could be used to take slightly overlapping pictures which could be patched together to form a 360 by 160 picture (see figure 19a). The alignment between the two cameras presents an interesting verification vision problem in itself. If the user positions the two cameras so they are approximately aligned, the system could automatically refine the alignment as follows:

(1) Take a picture with each camera
(2) Scan the correlation interest operator over the portion of one picture which is expected to overlap the other picture. This will produce a list of interesting correlation patches (see figure 19b).
(3) Locate these correlation patches in the second picture (see figure 19c).
(4) The differences between the two pictures may appear as an XY displacement, a rotation, and/or a scaling. Use the matching pairs to determine these values.

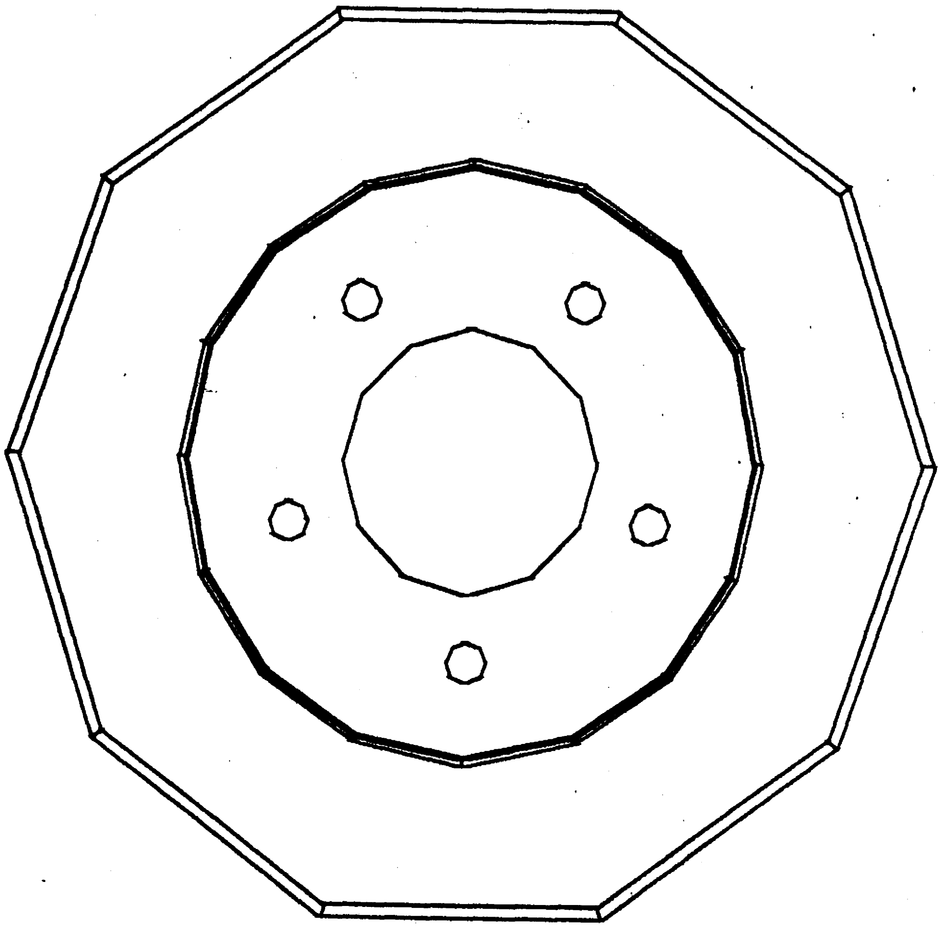
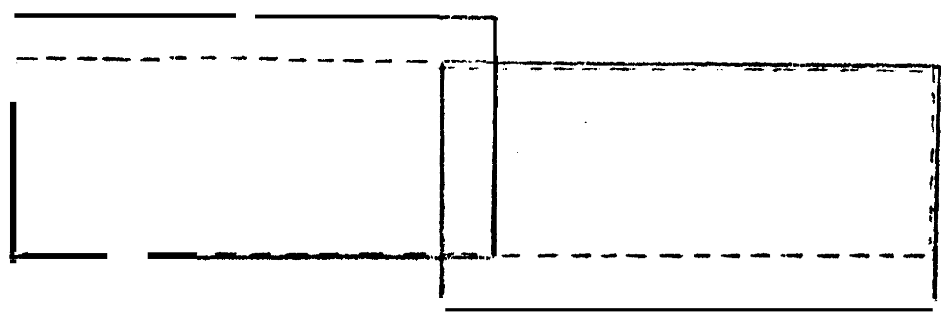
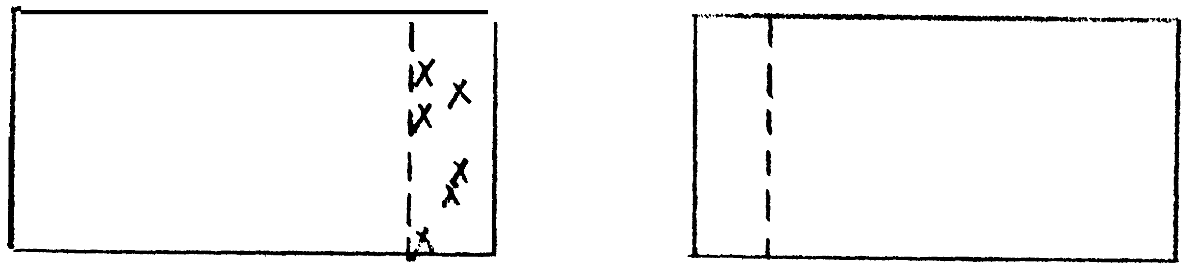


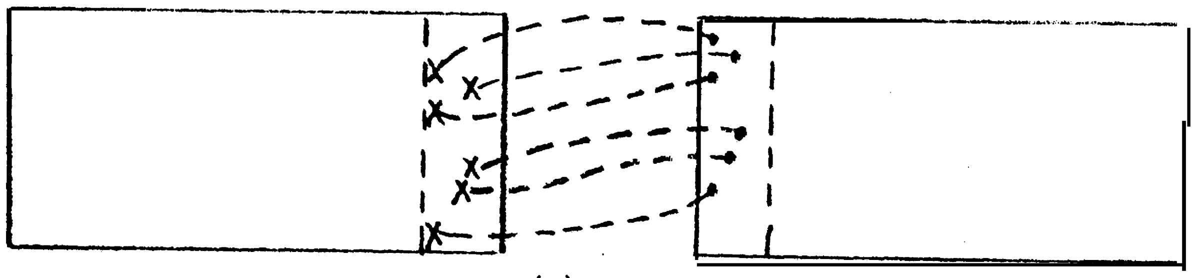
Figure 18.



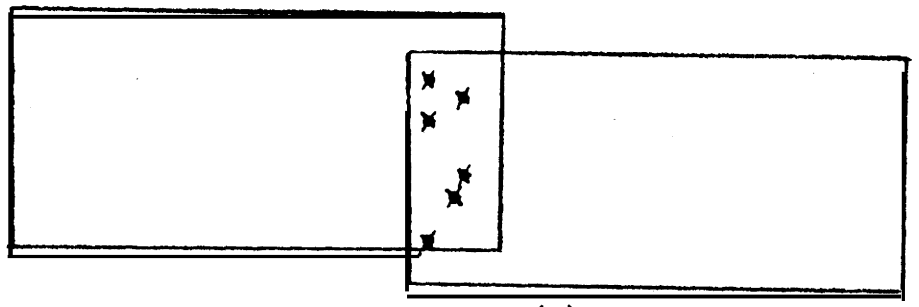
(a)



(b)



(c)



(d)

Figure 19.

(5) If the rotation or scaling is significant, the user should try to improve the position of one camera and try the process again.

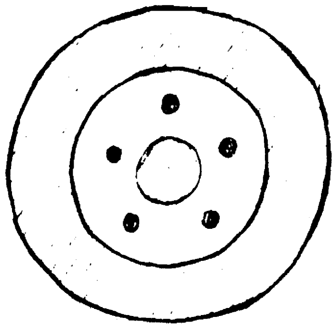
(6) When the rotation and scaling are almost the same, the two images can be logically combined into one, larger image (see figure 19d).

This relatively simple procedure would align the two cameras well enough to produce one picture which can be searched for features. The final calibrations of the cameras and the position computations should be done separately to maximize the precision.

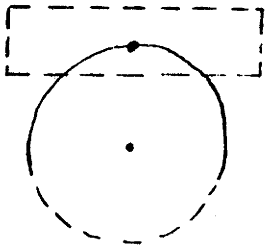
At this point assume that there is one large image available. The system could then AUTOMATICALLY generate synthetic images, pick out potential features, produce their tolerance regions, set up searches, and check for possible confusions. For example, the model would predict, a set of invariant curves (ie. the small, medium, and large ones shown in the figure 20a). Figure 20b shows the tolerance region about one point on the medium-sized curve. Notice that its tolerance region is smaller than might be expected because the curve is invariant, which means that the rotation uncertainties do not affect the size of the region. The suggested search --is shown in figure 20c. Figures 20d through 20f develop the implication region about the center of the shaft which can be made by finding a point on the curve. Figure 20d shows a linear search which has located an edge point with a certain slope. The arrow marks the point on the curve with that slope. Since the edge operator only returns approximate slopes (eg. ±5 degrees) the actual matching point may be any place within the range shown in figure 20e. There is a similar position uncertainty. These combine to produce the implication region for the center of the shaft shown in figure 20f.

A similar calculation produces the implication region for one of the bolts (see figure 20g). If the system were very smart it would notice that the uncertainty of the edge operator and the unknown rotation of the hub could be combined in a more compact way as shown in figure 20h. This would mean that locating one point on the circle could essentially eliminate the linear uncertainties in X and Y.

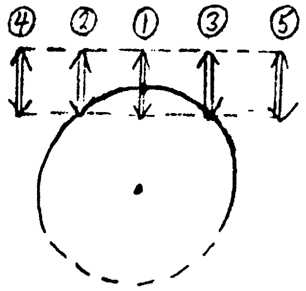
After analyzing the potential features the system would have a small network of features, the operators to use, their thresholds, the searches to use, and the implications to be made. For this problem the network would probably include curve segments and correlation operators to find the bolts. The system could then simulate the complete location process and doublecheck to make sure that the desired tolerances can be produced from the available features.



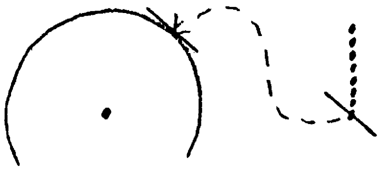
(a)



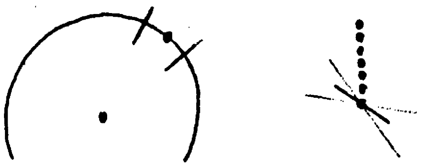
(b)



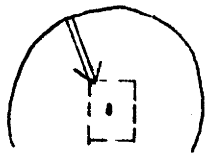
(c3



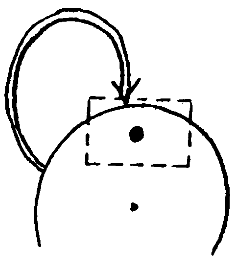
(d)



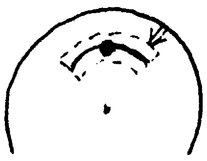
(e)



(f)



(g)



(h)

Figure 20.

THE IDEAL SYSTEM

The ideal system would not contain any new types of capabilities. But all of its subsystems would be 'complete.' For example, it would have a full range of features, such as textured regions and color. It would have a complete language for describing 2-D and 3-D constraints and the constraint solver to go with it. Its object modelling system would be able to handle any complex, textured surface and a synthetic picture routine to produce the corresponding pictures.

Such a system could play a major role within a general assembly strategist. If a user wanted to program the work station to place a bearing and seal on a shaft, he would have to do two things: (1) design the object models and (2) describe the task. Hopefully there would be a direct way of using the computer-aided design information as the basis for a model. After the designer has completed a design he could ship the information to the manufacturing engineer. In the case that there isn't any CAD information for a part there should be a descriptive vision system which builds a 3-D model from several views of the object. In effect this would do most of the geometry required and possibly some of the light reflectance calculations. The user would have to associate names and symbolic descriptions to any particular features he wanted to use in the task description.

The task description would be given in a "strategist's" language. The amount of detail required would be determined by the smartness of the strategist. For example, if the strategist were very smart it would know about bearings, seals, shafts, etc. and know that it would be useful to slip a sleeve over the end of the shaft. The sleeve would protect the seal and make the assembly easier for the arm. If the system didn't know this much, the user would have to suggest the use of the sleeve.

Assuming that the system knows about slipping things over other things, it would know that one of the most critical parameters is the relative tolerance between the parts. Therefore, it would check the diameter of the shaft and the inside diameter of the sleeve and combine them with the precision of the arm to decide if dead-reckoning is sufficient to make the alignment. Assume it is not. Then the strategist needs to decide on a type of feedback. Since there is nothing to touch or push as the sleeve is being positioned off the end of the shaft, visual feedback is probably the best alternative.

The strategist could even decide where the cameras should be. To do this it would have to take into account the necessary resolution, the other objects in the work station, and the room needed by the arms to perform the assembly. Once that has been done the verification vision system could be called to make sure that there were enough features visible to locate the necessary objects.

The ideal system minimizes the work required of a user and maximizes the reliability of the result. It "understands" assembly operations, tools, parts, tolerances and feedback. **It** knows about costs, mistakes, and confidences. And finally it can act as a part of an **overall** strategist that provides the user with a high-level task ***description*** language. **The ideal system** is, indeed, ideal.

LIST AND DISCUSS THE SEMANTIC SYSTEMS

The purpose of this section is to present a complete list of the desired capabilities and briefly discuss some of the unsolved problems. The complete list is presented as a comprehensive collection of the capabilities required by verification vision and to impress the reader with the magnitude of the problem. The list is reordered and regrouped by topic for the short discussions about the current status of work on some of the harder problems.

THE COMPLETE LIST OF CAPABILITIES

(from the basic system)

CAMERAS AND A METHOD FOR CALIBRATING THEM WITH RESPECT TO THE TABLE (OR OTHER OBJECTS)

A REPRESENTATION FOR 2-D TOLERANCE REGIONS

A METHOD OF SEARCHING A 2-D TOLERANCE REGION

A METHOD TO COMPUTE A 3-D POSITION FOR A FEATURE GIVEN TWO SETS OF COORDINATES FROM STEREO VIEWS

METHODS TO DETERMINE THE EXPECTED PRECISION OF A MONOCULAR OR STEREO LOCALIZATION

A SYSTEM FOR 3-D POINT MODELS OF OBJECTS

METHODS TO DETERMINE THE BEST ESTIMATE FOR THE NEW POSITION OF AN OBJECT GIVEN THE IMAGE COORDINATES FOR SEVERAL FEATURES (BOTH 2-D AND 3-D)

AN INTERACTIVE SYSTEM FOR SETTING UP RELIABLE CORRELATION OPERATORS AND INDICATING THE MATCHING FEATURE ON THE 3-D POINT MODEL OF THE OBJECT (THE CORRELATION SYSTEM MIGHT INCLUDE AN AUTOMATIC WAY OF SETTING THE THRESHOLDS REQUIRED TO DECIDE IF THERE IS A MATCH OR NOT)

A SYSTEM FOR DESCRIBING CONSTRAINTS

A REPRESENTATION FOR TOLERANCE VOLUMES

A METHOD FOR PRODUCING THE TOLERANCE VOLUME FROM A SET OF CONSTRAINTS

A METHOD FOR PRODUCING THE CORRESPONDING 2-D TOLERANCE REGION IN AN IMAGE FOR A TOLERANCE VOLUME

A METHOD FOR PRODUCING THE 2-D REGION TO BE **SCANNED** FOR POSSIBLE CONFUSIONS

(from the simple structure system)

A VARIETY--. OF "EXTENDED" FEATURES: LINES, CURVES, & REGIONS --- **2-D** REPRESENTATIONS FOR THEM (NOT 3-D CURVED SURFACE MODELS . . . REMEMBER THAT THE BASIC ASSUMPTION OF THE SIMPLE STRUCTURE SYSTEM IS THAT 2-D FEATURES AND TOLERANCE IMPLICATIONS ARE SUFFICIENT . . . 3-D IS ONLY USED TO COMPUTE THE ACTUAL LOCATION OF AN OBJECT)

OPERATORS TO LOCATE PARTS OF THESE FEATURES . . . **EG.** EDGE OPERATORS WHICH CAN LOCATE A POINT ON A LINE OR A CURVES, TEXTURE OPERATORS, ETC.

AN INTERACTIVE WAY OF DETERMINING THE VARIOUS THRESHOLDS AND LIMITS ASSOCIATED WITH THESE OPERATORS

SEVERAL SEARCH STRATEGIES TO CHOOSE FROM . . . **EG.** SPIRAL, LINEAR, & RANDOM

AN INTERACTIVE WAY OF SETTING UP AND EVALUATING SEARCH **STRATEGIES** TO LOCATE A PARTICULAR FEATURE

METHODS TO DO LOCAL CHECKING ABOUT EDGE POINTS, CORRELATIONS, AND REGION POINTS

A 2-D SYSTEM FOR PREDICTING THE RANGE OF POSITIONS FOR A FEATURE ONCE ANOTHER FEATURE HAS BEEN FOUND

A FORM FOR VERIFICATION VISION PROGRAMS

(from **the** fancier **system)**

A 3-D MODELLING SYSTEM WHICH INCLUDES SURFACE INFORMATION SUCH AS REFLECTANCE . . . IT SHOULD ALSO BE ABLE TO MODEL SOME CURVED SURFACES, EVEN IF THEY HAVE TO BE HANDLED INDIRECTLY

A LIGHT MODEL . . . IE. A POSITION AND INTENSITY OF THE LIGHT SOURCE

A HIDDEN-LINE ELIMINATION METHOD

A CURVE FITTING ROUTINE . . . EC. A SPLINE PACKAGE

A SYNTHETIC GREY-SCALED PICTURE GENERATION METHOD

A SET OF 'INTEREST' OPERATORS TO SCAN THE WIRE-DIAGRAM PICTURES AND SYNTHETIC PICTURES IN ORDER TO LOCATE POTENTIALLY USEFUL FEATURES

A METHOD FOR AUTOMATICALLY SETTING UP A SEARCH PATTERN

A REPRESENTATION FOR A RANGE OF SCENES

A METHOD FOR' AUTOMATICALLY DETERMINING 'IMPLICATION REGIONS' FROM ONE FEATURE TO ANOTHER

A METHOD TO DETERMINE THE CONSTRAINTS THAT APPLY AT THE EXTREMES OF A TOLERANCE REGION

A SOPHISTICATED CONSTRAINT LANGUAGE AND RESOLVING SYSTEM

A SYSTEM OF CONFIDENCES

A SYSTEM OF COSTS

A NETWORK OF FEATURES (INSTEAD OF AN EXPLICIT PROGRAM)

AN INTERPRETER WHICH CAN DO A COST/BENEFIT ANALYSIS TO DETERMINE WHAT SHOULD BE DONE NEXT

A METHOD TO CONVERT A NETWORK OF FEATURES INTO A COMPILED PROGRAM WHICH HANDLES THE NECESSARY RANGE OF POSSIBILITIES

A DESCRIPTIVE SYSTEM FOR OPERATORS, FEATURES, SEARCHES, ETC.

REORDERED BY TOPIC

MODELLING

(from the basic system)

A SYSTEM FOR 3-D POINT MODELS OF OBJECTS

AN INTERACTIVE SYSTEM FOR SETTING UP RELIABLE CORRELATION OPERATORS AND INDICATING THE MATCHING FEATURE ON THE 3-D POINT MODEL OF THE OBJECT (THE CORRELATION SYSTEM MIGHT INCLUDE AN AUTOMATIC WAY OF SETTING THE THRESHOLDS REQUIRED TO DECIDE IF THERE IS A MATCH OR NOT)

(from the simple structure system)

A VARIETY OF "EXTENDED" FEATURES: LINES, CURVES, & REGIONS --- 2-D REPRESENTATIONS FOR THEM (NOT 3-D CURVED SURFACE MODELS ... REMEMBER THAT THE BASIC ASSUMPTION OF THE SIMPLE STRUCTURE SYSTEM IS THAT 2-D FEATURES AND TOLERANCE IMPLICATIONS ARE SUFFICIENT ... 3-D IS ONLY USED TO COMPUTE

THE ACTUAL LOCATION OF AN OBJECT)

(from the fancier system)

A 3-D MODELLING SYSTEM WHICH INCLUDES SURFACE INFORMATION SUCH AS REFLECTANCE ... IT SHOULD ALSO BE ABLE TO MODEL SOME CURVED SURFACES, EVEN IF THEY HAVE TO BE HANDLED INDIRECTLY

A LIGHT MODEL . . . IE. A POSITION AND INTENSITY OF THE LIGHT SOURCE

A HIDDEN-LINE ELIMINATION METHOD

. A CURVE FITTING ROUTINE . . . **EG.** A SPLINE PACKAGE

A SYNTHETIC GREY-SCALED PICTURE GENERATION METHOD

A SET OF 'INTEREST' OPERATORS TO SCAN THE WIRE-DIAGRAM PICTURES

A REPRESENTATION FOR A RANGE OF SCENES

A NETWORK OF FEATURES (INSTEAD OF **AN** EXPLICIT PROGRAM)

This list contains several capabilities which are only partially understood: 3-D **modelling,** light models, visual features, and ranges of scenes. The general idea is that the verification vision system will be based upon the currently available techniques and will be expanded to incorporate new techniques as they are perfected. Three-dimensional **modelling is a typical example.** The basic system and the simple structure system only use 3-D point **models** of the objects in the scene. When some of the ideas about 'affix structures' and curved surfaces **have been** better developed they will be included. There are several **people** working on these ideas: (see **[FINKEL],** [TAYLOR], **[LIEBERMAN], [AGIN], [NEVATIA], [MIYAMOTO], [BAUMGART], [COONS],** [GORDON], and [GOULD]).

Light modelling and synthetic picture generation techniques are currently **being developed** to produce high quality pictures of scenes containing curved **objects (see**

[GOURAUD] and [RIESENFELD]). The resulting pictures look good to people, but there are a number of reasons why such pictures are NOT accurate predictions of actual images. The techniques either do not handle or only partly handle the following: (I) several light sources, (2) indirect, lighting, (3) shadows, or (4) textured surfaces. Horn has recently published a collection of the more theoretical ideas concerning light intensities and how they should be treated (see [HORN1975]).

There is currently no way to represent "all possible views of a scene" given the set of objects in the scene and a set of constraints on those objects. The idea is to produce the "range of pictures" and scan it for interesting features, possible confusions, and abrupt changes caused by occlusions. A linear movie is not enough. The constraints often produce a multi-dimensional set of possible images. It may be possible to approximate such a range with a set of linear sub-ranges.

<u>VISUAL OPERATORS</u>

(from the basic system)

AN INTERACTIVE SYSTEM FOR SETTING UP RELIABLE CORRELATION OPERATORS AND INDICATING THE MATCHING FEATURE ON THE 3-D POINT MODEL OF THE OBJECT (THE CORRELATION SYSTEM MIGHT INCLUDE AN AUTOMATIC WAY OF SETTING THE THRESHOLDS REQUIRED TO DECIDE IF THERE Is A MATCH OR NOT)

(from the simple structure system)

A VARIETY OF "EXTENDED" FEATURES: LINES, CURVES, & REGIONS --- 2-D REPRESENTATIONS FOR THEM (NOT 3-D CURVED SURFACE MODELS ... REMEMBER THAT THE BASIC ASSUMPTION OF THE SIMPLE STRUCTURE SYSTEM IS THAT 2-D FEATURES AND TOLERANCE IMPLICATIONS ARE SUFFICIENT ... 3-D IS ONLY USED TO COMPUTE THE ACTUAL LOCATION OF AN OBJECT)

OPERATORS TO LOCATE PARTS. OF THESE FEATURES . . . EG. EDGE OPERATORS WHICH CAN LOCATE A POINT ON A LINE OR A CURVES, TEXTURE OPERATORS, ETC.

AN INTERACTIVE WAY OF DETERMINING THE VARIOUS THRESHOLDS AND LIMITS ASSOCIATED WITH THESE OPERATORS

METHODS TO DO LOCAL CHECKING ABOUT EDGE POINTS, CORRELATIONS, AND REGION POINTS

(from the fancier system)

A DESCRIPTIVE SYSTEM FOR OPERATORS, FEATURES, SEARCHES, ETC.

There is a need for a wider variety of visual features and operators to find such features. Some of the most useful would be operators which could grow textured regions and/or locate boundaries between two textured regions. There are some promising techniques being explored (eg. see [BAJCSY], [LIEBERMAN], and [MARR]), but progress has been slow.

There should be a general system for describing how effective an operator is under certain conditions. Such a system could be used by a strategist to determine which operators should be used. The problem of determining the effectiveness of an operator is closely related to the automatic methods for setting thresholds for the operators. Such techniques are available for some of the more common operators (see [BINFORD] and [QUAM]), but better characterizations are needed.

CONSTRAINTS

(from the basic system)

A REPRESENTATION FOR 2-D. TOLERANCE REGIONS

A METHOD TO COMPUTE A 3-D POSITION FOR A FEATURE GIVEN TWO SETS OF COORDINATES FROM STEREO VIEWS

METHODS TO DETERMINE THE BEST ESTIMATE FOR THE NEW POSITION OF AN OBJECT GIVEN THE IMAGE COORDINATES FOR SEVERAL FEATURES (BOTH 2-D AND 3-D)

A SYSTEM FOR DESCRIBING CONSTRAINTS

A REPRESENTATION FOR TOLERANCE VOLUMES

A METHOD FOR PRODUCING THE TOLERANCE VOLUME FROM A SET OF CONSTRAINTS

A METHOD FOR PRODUCING THE CORRESPONDING 2-D TOLERANCE REGION IN AN IMAGE FOR A TOLERANCE VOLUME

A METHOD FOR PRODUCING THE 2-D REGION TO BE SCANNED FOR POSSIBLE CONFUSIONS

(from the simple structure system)

A-2-D SYSTEM FOR PREDICTING THE RANGE OF POSITIONS FOR A FEATURE ONCE ANOTHER FEATURE HAS BEEN FOUND

'(from the fancier system)

A METHOD FOR AUTOMATICALLY DETERMINING 'IMPLICATION REGIONS' FROM ONE FEATURE TO ANOTHER

A METHOD TO DETERMINE THE CONSTRAINTS THAT APPLY AT THE EXTREMES OF A TOLERANCE REGION

A SOPHISTICATED CONSTRAINT LANGUAGE AND RESOLVING SYSTEM

Two-dimensional constraints are relatively straightforward. The completely general three-dimensional constraint solver, on the other hand, is extremely difficult. Thus, one of the main concerns of this paper has been the approximation of 3-D constraints and their implications by a 2-D constraint system. There are several theoretical questions about how effective this can hope to be. The 2-D approximations are used to reduce the amount of work required to locate important features. The better the approximations are, the less work has to be done to find the features. The final positions are always calculated in 3-D.

There are a few people working on constraint systems for a limited class of constraints (see [TAYLOR] and [AMBLER]). They provide for constraints such as: plane P contacts plane Q, cylinder C is in V-slot X, and point Y is in box B.

STRATEGIES

(from the simple structure system)

SEVERAL SEARCH STRATEGIES TO CHOOSE FROM . . . EG. SPIRAL, LINEAR, & RANDOM

AN INTERACTIVE WAY OF SETTING UP AND EVALUATING SEARCH STRATEGIES TO LOCATE A 'PARTICULAR FEATURE

A 'FORM FOR VERIFICATION VISION PROGRAMS

(from the fancier system)

A METHOD FOR AUTOMATICALLY SETTING UP A SEARCH PATTERN

A SYSTEM OF CONFIDENCES

A SYSTEM OF COSTS

A NETWORK OF FEATURES. (INSTEAD OF AN EXPLICIT PROGRAM)

AN INTERPRETER WHICH CAN DO A COST/BENEFIT ANALYSIS TO DETERMINE WHAT SHOULD BE DONE NEXT

A METHOD TO CONVERT A NETWORK OF FEATURES INTO A COMPILED PROGRAM WHICH HANDLES THE NECESSARY RANGE OF POSSIBILITIES

A DESCRIPTIVE SYSTEM FOR OPERATORS, FEATURES, SEARCHES, ETC.

Another one of the basic questions about verification vision is "how can the system take advantage of all of the information that is available?" This requires several subsystems to handle various types of semantics, but it also requires some organizing principle which encompasses the whole process. In the basic, system there is only a "fixed" strategy: find as much as possible and solve for the new position. The simple structure placed the strategy problem in the user's lap. The user had to decide what to try to find, when, and what to do if something is found. Both of these systems are only temporary solutions to the strategy problem. The ultimate system will know about costs, constraints, and confidences and will be able to determine a cost-effective plan for locating the desired objects. Feldman and Sproull have developed one of the most comprehensive systems for this type of planning (see [FELDMAN]). Other systems which do their own planning for visual processing are [YAKIMOVSKY] and [GARVEY].

CONCLUSION

There were two main purposes for this paper: (1) distinguish **a sub-class** of **visual** feedback tasks (in particular, verification vision tasks) and (2) characterize a set of **general-purpose** capabilities which, if implemented, would provide a user with a system in which to write programs to perform such **tasks.** The example tasks and protocols motivated the various semantic capabilities which are needed within a verification vision system. The four different levels of verification systems showed how these capabilities could be incorporated into working systems. But there are several research questions which have to be answered before such systems can be implemented. For example, **object modelling and** constraint solving are particularly interesting and virtually open-ended **problems.** In addition there are several smaller problems whose solutions were only roughly sketched **out.** In general the intuitive ideas need to be formalized and the heuristics need to be **theoretically** analyzed and converted into algorithms (if possible).

The overall-goal of verification vision is to make visual feedback **a viable** alternative within programmable assembly. It is intended to complement touch and force **feedback which** are already reasonably well understood. Instead of writing a special-purpose program **from** scratch for each visual feedback task, verification vision will offer a structured system for programming visual feedback operations in a straight-forward way. The system **will** know about the costs for different approaches, about the increase in confidence from finding a feature, and about the reduction in tolerances as more and more information **is gathered. Visual** feedback should become a standard part of programmable assembly systems.

BIBLIOGRAPHY

Agin, G . J . [1972], "Representation and Description of Curved Objects," Stanford Artificial Intelligence Project Memo No. 173, October 1372.

Agin, G. J . and Binford, T. O. [1973], "Computer Oescr ipt ion of Curved Objects, " Proceedings of the Third International Joint Conference on Artificial Intel l igence, Stanford, August 1973, 629-640,

Ambler, A. P. and Popplestone, R.J. [1973], "Inferring the Positions of Bodies from Specified Spatial Relationships," Dept. of Machine Intel l igence, University of Edinburgh, Edinburgh, Scotland.

Ba jcsy, R. [1973], "Computer Description of Textured Surfaces," Proceedings of the Third International Joint Conference on Artificial Intel l igence, Stanford, Aug. 1973, 572-579.

Baumgar t, Bruce G. [1974a], "GEOMEO - A Geometric Editor," Stanford Artificial Intelligence Project Memo No. 232, May 1974.

Baumgart, Bruce G.[1974b], "Geometric Model ing for Computer Vision," Stanford Artificial Intelligence Project Memo No. 249, October 1974.

Binford, T. 0. [1975] "Optimizing the Huecke l Operator, " an i nternal memorandum at the Stanford Artificial Intel l igence Project, December 1375.

Bol les, R. C. and Paul, R. [1973], "The Use of Sensory Feedback in a Programmable Assembly System," Stanford Artificial Intelligence Project Memo No. 220, October 1973.

Coons, S. A. [1967], "Surfaces for Computer-aided Design of Space Forms," MIT Project MAC, MAC-TR-41, June 1967.

Oeuar, R., Lewis, N. R., Rossol, L., and Olsztyn, J. T. [1973], "An Application of Computer Vision to Automatic Wheel Mounting," The First International Joint Conference on Pattern Recognition, October 1373.

Falk, Gilbert [1970], "Computer Interpretation of Imperfect Line Data as a Three-Dimensional Scene," Stanford Artificial Intel l igence Project Memo' No. 132, August 1970.

Feldman, J. 'A. and Sproul l, Robert [1974], "Decision Theory and Art ificial

Intel I igence: An Approach to Generating Efficient Plans," draft, July 1974.

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.

Garvey, Thomas D. [1975], "Perceptual Strategies for Locating Objects in Indoor Scenes, " Forthcoming Stanford PhD Thesis.

Gordon, W. J. and Riesenfeld, R. F. [1972], "Bernstein-Bezier Methods for the Computer-aided Design of Free-form Curves and Surfaces," General Moters 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, Henr i [1971], "Computer Display of Curved Surfaces," University of Utah Technical Report, UTEC-CSc-71-113, June 1971.

Hannah, Marsha Jo [1974], "Computer Hatching of Areas in Stereo Images," Stanford Artificial Intel l igsnce Project Memo No. 239, July 1974,

Horn, Berthold K. P. [1970], "Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View," MAC-TR-73, NIT, Cambr i dge, November, 1970.

Horn, Berthold K. P. [1975], "Image Intensity Understanding," Massachusetts Institute of Technology AID No. 335, August 1375.

Lieberman, Lawrence [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 5483, 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.

Harr, D. [1975], "ANALYZING NATURAL IMAGES: a computation theory of texture vision," HIT Artificial Intelligence Laboratory Memo No. 334, June 1975'.

Hiyamoto, Eiichi and Binford, T. B. E19751, "Display Generated by a Generalized Cone Representation,' Computer Graphics and Imege Processing Conference, Anaheim, Ca. Hay 1975.

Nevatia, R. and Binford, T. O. [1973], "Structural Description of Complex Dbjects," Proceed i ngs of the Third International Conference on Artificial Intelligence, Stanford, August, 1973, pp. 641-647.

Quam, Lynn H. [1971], "Computer Comparison of Pictures," Stanford Artificial Intelligence Project Memo No. 144, Hay 1971,

Quam, Lynn H., Sidney Liebes, Jr., Robert B. Tucker, Marsha Jo Hannah, and Botond G. Eross 119721, "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, December 1974.

Riesenfeld, Richard [1973], "Applications of B-spline Approximation to Geometric Problems of Computer-aided Design," University of Utah Technical Report UTEC-CSc-73-126, March 1973.

Shirai, Yoshiaki [1973], "A Heterarchical System for Recognition of Polyhedra," Artificial Intelligence, Vol. 4, No. 2, 1973.

Sobsl, Irwin [1970], "Camera Models and Machine Perception," Stanford Artificial Intelligence Project Memo No. 121, Hay 1370,

Taylor, Russell H. [1975], "Assembly Robot Program Automation," Forthcoming Stanford PhD Thesis.

Tenenbaum, Jay H. [1970], "Accommodation in Computer Vision," Stanford Artificial Intelligence Project Memo No. 134, September 1970.

Thomas, Arthur J. and Pingle, Karl [1974], "A Fast, Feature-Orliven Stereo Depth Program,' Stanford Artificial Intelligence Project Memo No. 248 July.1974, Yakimovsky, Y. [1973a], "Scene Analysis using a Semantic Base for Region Growing," Stanford Artificial Intelligence Project Memo No. 289,

Yakimovsky, Y. and Feldman, J. [1973b], 'A Semantics-based Decision Theory Region Analyzer," Proceedings of the Third International Joint Conference on Artificial Intel l igence, Stanford, August, 1973, pp. 580-588.