3DPO: A Three-Dimensional Part Orientation System

Robert C. Bolles
SRI International, EK290
333 Ravenswood Avenue
Menlo Park, California 94025

Patrice Horaud, Radu
Laboratoire d'Automatique de Grenoble
BP 46 38402
Saint-Martin d'Heres, France

Abstract

In this paper, we present a system that recognizes objects in a jumble, verifies them, and then determines some essential configurational information, such as which ones are on top. The approach is to use three-dimensional models of the objects to find them in range data. The matching strategy starts with a distinctive edge feature, such as the edge at the end of a cylindrical part, and then "grows" a match by adding compatible features one at a time. (The order of features to be considered is predetermined by an interactive, off-line, feature-selection process.) Once a sufficient number of compatible features has been detected to allow a hypothesis to be formed, the verification procedure evaluates it by comparing the measured range data with data predicted according to the hypothesis. When all the objects in the scene have been hypothesized and verified in this manner, a configuration-understanding procedure determines which objects are on top of others by analyzing the patterns of range data predicted from all the hypotheses. We also present experimental results of the system's performance in recognizing and locating castings in a bin.

1. Introduction

How do you find a spatula in a drawer of kitchen utensils? If it is partially covered by other objects, as it probably is, you recognize one of its features, such as the handle, and then use it to hypothesize the position and orientation of the whole object. The more distinctive the visible features are, the easier it is to find the object.

We are interested in everyday household tasks not only because we want to learn more about how people perform them, but also because we want to develop techniques for performing similar industrial tasks. In this paper, we are particularly concerned with the problem of recognizing and locating identical objects jumbled together in a bin (see Fig. 1). By working on this class of difficult tasks, we plan to develop general-purpose techniques for recognizing and locating partially visible objects. Our approach is to use 3-D models of the objects to find them in range data. Our rationale for this approach is that, first of all, range data simplify the locational analysis since the geometric information is encoded directly in the data. Secondly, it will soon be economical to use range sensors in industrial tasks. Finally, familiarity with the model of a part will add enough new constraints to make it practical to locate relatively complex parts jumbled together in a bin.

Speed is extremely important when performing industrial tasks. One way to achieve speed is to locate a minimum of object features and extract as much information as possible from them. For example, after a feature has been found, its identity can be used to suggest the next feature to be located, and its position can constrain the region to be searched. A typical strategy of this type follows:

1. Locate a distinctive feature of the object to be found.
2. Use that feature's position to suggest where to look for a second feature to verify the first.
3. Use the two features to predict a third feature that completely constrains the position and orientation of the object.

The research reported here was supported by the National Science Foundation under grant DAR-7922248.

The International Journal of Robotics Research,
Vol. 5, No. 3, Fall 1986,

Fig. 1. Bin of castings.

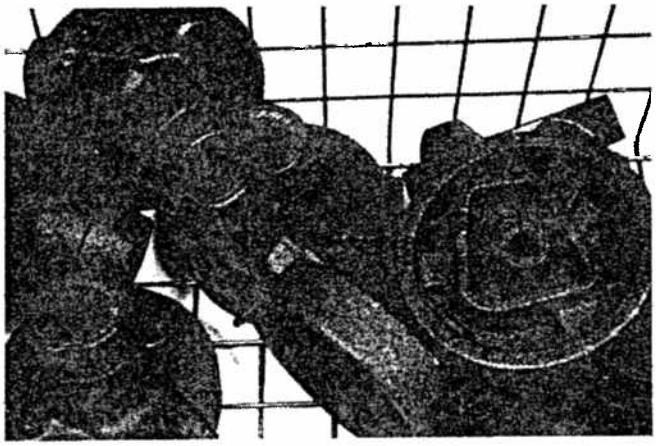
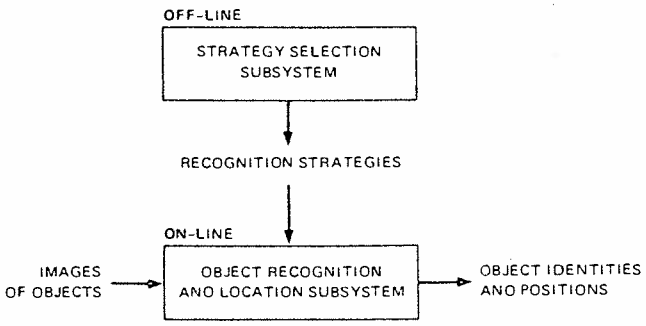


Fig. 2. Top-level block diagram of the 3DPO System.



Since some of the predicted features may not be visible or the feature detectors may miss some visible ones, alternatives have to be provided. Therefore, a complete strategy is an ordered tree of features, and the recognition process is a tree search.

As with all tree searches, it is important to order the alternatives according to their expected utility. This may be done in advance to minimize execution time. Thus, the system is naturally divided into two stages (see Fig. 2). The first stage selects the recognition strategies and the features to be used in them, while the second stage applies the strategies to locate objects in range data. Since the selection process is done only once for each task, it can be performed off-line and pass its results to the on-line system to be used repeatedly.

One of our goals for the 3DPO (three-dimensional part orientation) system is to explore ways to select strategies and features automatically. In this paper, we concentrate on the development of an aggressive recognition system that tries to use as much of the available information as possible at each step of the process. For example, in addition to the constraints derived from previously located features, the system uses (1) the geometry of the sensor to classify range discontinuities; (2) the intrinsic properties of the object features to reduce the list of candidate matches; and (3) the fact that the objects are opaque to predict range images, which in turn are used to verify hypothesized objects.

In this paper we describe a system that recognizes objects in a jumble, verifies them, and determines some essential configuration information, such as which ones are on top. In Section 2, we present the approach and show how it relates to other approaches. In Section 3, we describe the object representation scheme, which is designed to support both the off-line and on-line portions of the system. In Section 4, we discuss the rationale behind the selection of the strategies and features to recognize a particular object. In Sections 5–9, we describe the steps of the on-line recognition procedure. In each section we include a discussion of the strengths and weaknesses of the techniques employed. We conclude with a brief outline of work projected for the future.

Throughout the paper we illustrate the techniques being described by applying them to the task of recognizing the casting in Fig. 1. We chose this object because it is a moderately complex part that is typical of a large class of industrial parts.

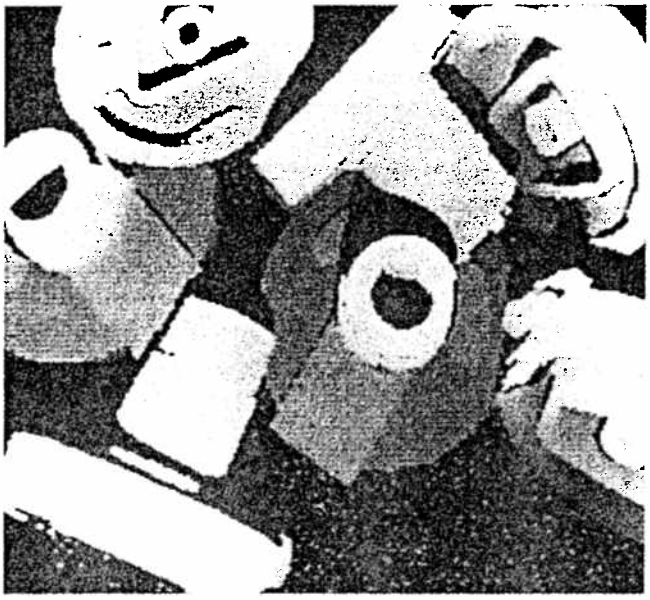
2. Approach

There are five steps in the on-line recognition process:

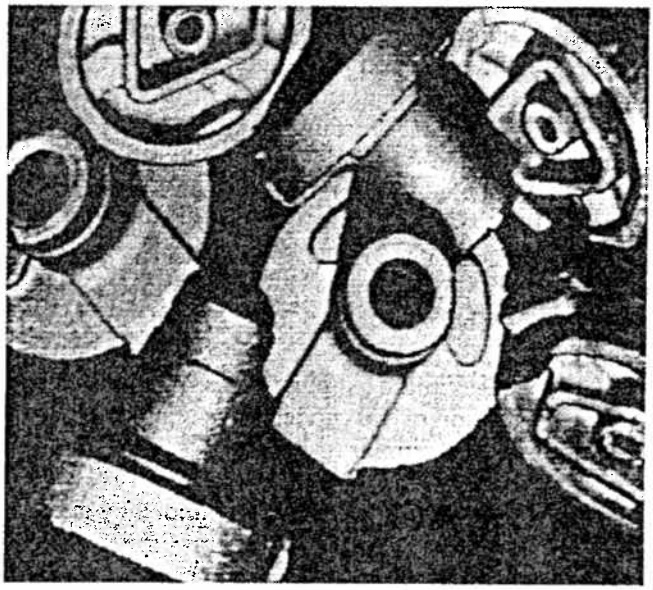
1. data acquisition,
2. feature detection,
3. hypothesis generation,
4. hypothesis verification,
5. configuration understanding.

First, we gather a "dense" range image of the scene (see Fig. 3). In Fig. 3A, the image points closer to the sensor are lighter than those farther away. Second, we locate features in the data, such as an arc at the end of

Fig. 3. Registered range and intensity images of a jumble. A. Height image of a jumble. B. Intensity image of jumble.



A



B

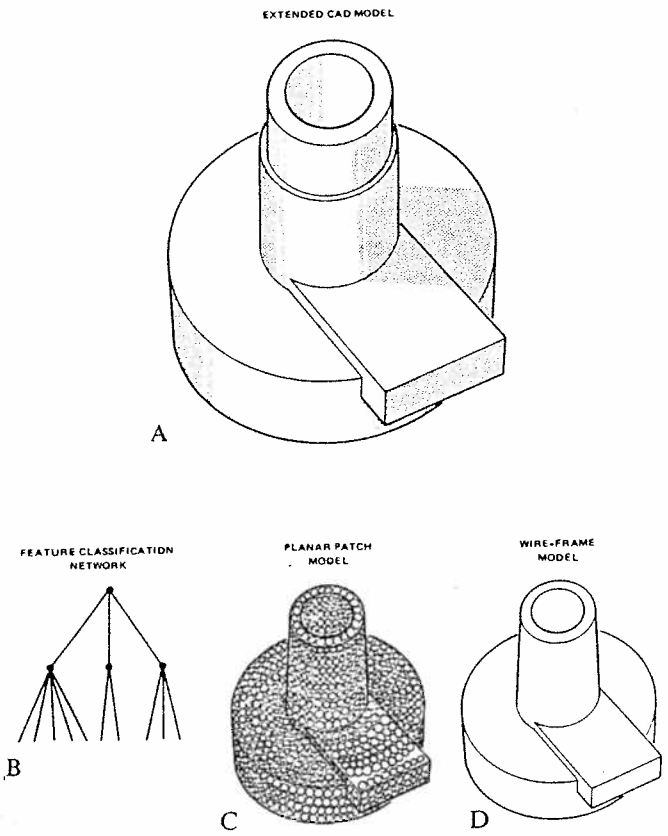
a cylinder or a straight edge formed by the intersection of two planes. Third, we "grow" a match around a distinct feature by adding compatible features, one at a time. The order of features to be considered is predetermined by the off-line feature selection process. As each new addition is made to the set of mutually consistent features, the system refines its estimate of the object's pose. (*Pose* is the object's position and orientation.) When a sufficient number of features has been added to determine the pose, the matching system forms a hypothesis and passes it to the verification procedure for assessment. The verification procedure uses the hypothesis to predict the range data that should have been measured if the object were in fact at the indicated pose and compares these predictions to the measured data. If the predictions match the data, the verifier accepts the hypothesis. If not, it rejects it. When all the objects have been thus hypothesized and verified, the configuration-understanding procedure determines which objects are on top of other objects by analyzing the patterns of range data predicted from the hypotheses.

We see industrial vision researchers reaching a point of convergence on a matching strategy that is a tree search (e.g., Oshima and Shirai 1978; Faugeraus and Hebert 1983; and Grimson and Lozano-Perez 1984). These groups have concentrated on different types of low-level features and applied somewhat different compatibility checks, but the matching methods are essentially the same. Our approach is of this type. It requires more preliminary analysis of the objects than some of the earlier approaches because it emphasizes key features, but the gain in efficiency is often worth the effort.

Range analysis programs have generally concentrated on one type of feature. Some have worked with surface patches (e.g., Oshima and Shirai 1978; Duda, Nitzan, and Barrett 1979; and Faugeraus and Hebert 1983). Others have worked with edges (e.g., Nevatia and Binford 1977). And some others have worked with simple shapes (e.g., Shirai and Suwa 1971; Popplestone et al. 1975; and Nitzan, Brain, and Duda 1977). The 3DPO system starts with edges, but quickly expands its analysis to include the adjoining surfaces. This approach is particularly well-suited to industrial parts that have distinct edges. Dave Smith at Carnegie-Mellon University has recently developed a similar system that builds descriptions of objects, such as pans and shovels, from a combination of surface and edge features (described in a yet unpublished manuscript).

Most object recognition research has concentrated on making hypotheses, not verifying them. In this

Fig. 4. Four-part object model. A. Extended CAD model. B. Feature classification network. C. Planar-patch model. D. Wire-frame model.



paper we include techniques for such verification, as well as for analyzing sets of hypotheses to ascertain which parts are on top of others.

3. Object Modeling

Computer-aided-design (CAD) systems and their representations are intended for constructing and displaying objects, not recognizing them. Even though a CAD model may be complete in the sense that it contains all the 3-D information about an object, there are more convenient representations for a recognition system. For example, a CAD model might state the size and position of a hole, but a recognition system needs a list of all the holes of that size. In general, a recognition system must be able to provide answers to such questions as follow:

How many features are there of a given type and size?
Which surfaces intersect to form this edge?
What other features lie in this plane?
What neighboring feature could be used to distinguish this feature from others like it?

To answer these types of questions efficiently, each object feature should be listed under several different classifications. For example, a dihedral edge should be in the list of edges bounding the two surfaces that meet and form the edge; it should be in the list of all dihedral edges (classified according to their included angles); and it may also be in other lists, such as the list of features in a specific plane. This redundancy is the key to efficient processing.

In the 3DPO system, the model of a part consists of four components: an extended CAD model, a feature classification network, a planar patch model, and a wire-frame model (see Fig. 4). The extended CAD model is the primary model of the part from which the other models are derived. Each is designed for a specific function. The feature network supports the off-line feature selection procedure; the planar patch model is used to predict range data for the verification and configuration-understanding procedures; and the wire-frame model is used to display hypotheses.

The extended CAD model consists of a standard volume-surface-edge-vertex description and pointers linking topologically connected features. For example, the representation of an edge includes pointers to the two surfaces that intersect to form it, while the representation of a surface contains an ordered list of its boundary edges and a list of its holes. We have implemented a version of this modeling system that uses a pointer structure similar to Baumgart's winged-edge representation (Baumgart 1972). The primitives are cylindrical and planar surfaces surrounded by circular arcs and straight lines. We selected this limited set of primitives because they are common components of machined and cast objects and can be modeled mathematically with ease.

Figure 5 shows two views of the casting model built in terms of these primitives. It contains 7 full cylinders, 8 partial cylinders, 25 planar patches, all of which are bounded by 32 circular arcs and 28 straight lines. These numbers are large enough to exclude the straightforward matching strategy that compares each

Fig. 5. Model of casting (bottom and top).

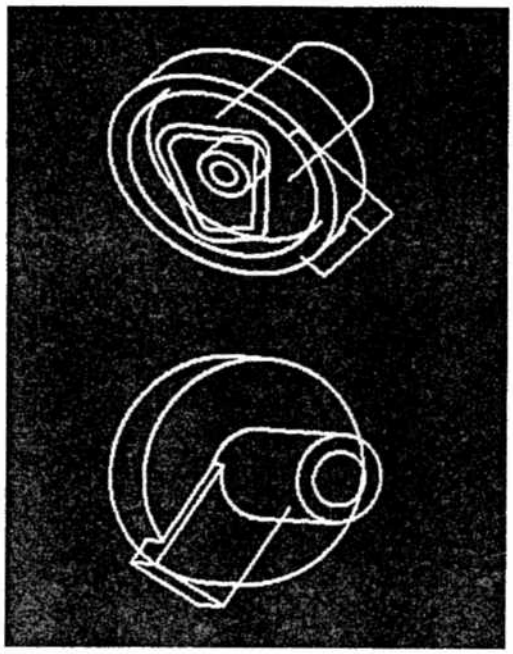
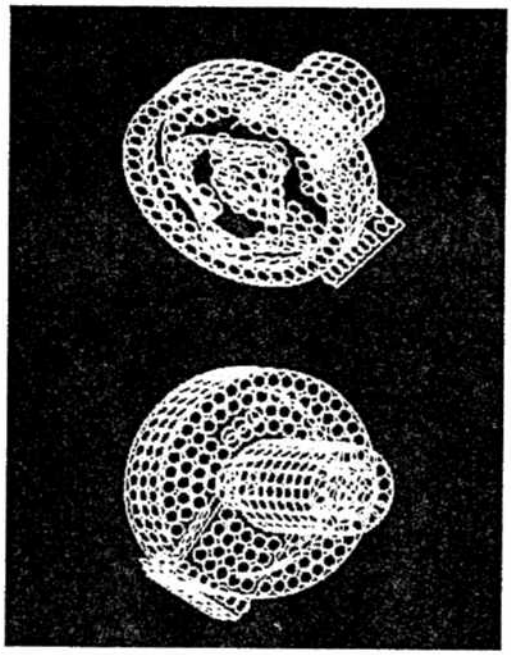


Fig. 6. Planar patch model of casting (bottom and top).



observed feature (i.e., line, arc, plane, or cylinder) with each model feature and tries to find the largest set of consistent matches. The combinatorial explosion inherent in this search, however, can be reduced dramatically by carefully selecting the order of the features to be considered, measuring certain properties of the observed features, and then restricting the matches to those between features with similar properties.

The second component of an object model in the 3DPO system is a feature network, which is primarily designed to support the off-line feature selection portion of the system. In the current implementation, the network classifies features according to their types and sizes. There are several different lists, including a list of circular arcs (sorted by radius) and a list of straight edges (sorted by length). Given these ordered lists as a data base, we have implemented a set of routines that can answer some of the questions mentioned earlier. These routines extract entries from these lists and then analyze the topology of the object in the vicinity of each extracted feature. For example, these routines can quickly produce a list of all circular edges that are concave dihedral angles and have radii of between .8 and 1.0 inch.

We plan to include other feature groupings, such as lists of surface elements that have a common normal or lists of cylinders that have a common axis. Each representation will be designed for a set of special-purpose procedures that analyze the data in terms of a single property—yet all the representations will contain pointers back to the extended CAD model, which serves as the core representation.

The third component of a 3DPO object model is a planar patch description of the object. It is composed of lists of small planar patches on the object's surface. Figure 6 shows two views of the model of the casting, which consists of approximately 500 patches. The patches are grouped into sublists of those lying on individual object features, such as a cylindrical or planar surface. In the current system, each planar patch is the same size and is represented by a 3-D location for its center and a surface normal. The surface normals are used by the range prediction routines to eliminate quickly those surfaces that face away from the sensor. This simple surface model was used instead of a more complete CAD model because it provided an easy way to predict range values.

The fourth and final component of an object model

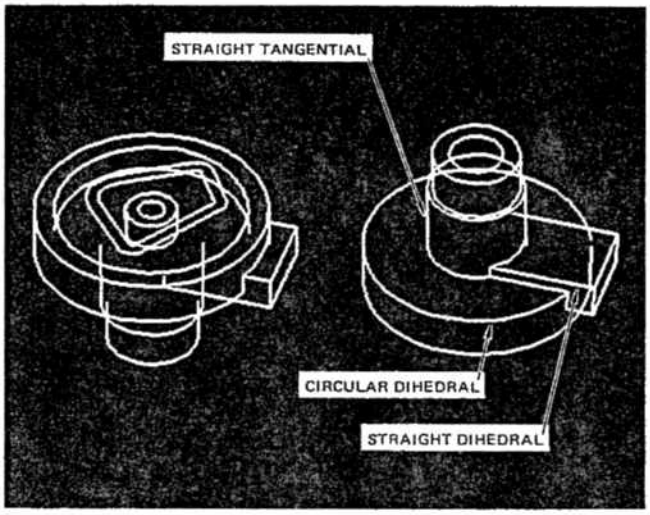
Fig. 7. Three types of edges.

is the wire-frame model. It is a list of object features, such as cylinders and planes, to be displayed for hypothesized objects. Figure 5 is actually a view not of the CAD model, but of the wire-frame model of the casting. The CAD model includes significantly more features than the ones shown. Given a hypothesis to display, the current display routine applies a few simple rules to determine which lines to draw for each feature. A more complete graphics system would produce a more accurate rendering of the object. With our relatively crude graphics system we have found it helpful to simplify the display by showing only a few key features.

In the 3DPO modeling system we differentiate between view-independent relationships, such as topological connectivity, and those that are view-dependent, such as image proximity (in pixels). The *view-independent relationships* are functions of the inherent geometric characteristics of the part, such as its size and topology, and can easily be enumerated by an analysis of the part model. The *view-dependent relationships,* on the other hand, are functions of the sensor, its type, and its location; they are significantly more difficult to list. Even a simple part can exhibit 50 or more structurally different views (Chakravarty 1982). Also see Koenderink and Van Doorn (1976) for a discussion of the changes in appearance of an object as it is rotated in front of a viewer. We plan to investigate techniques similar to those used in ACRONYM (Brooks 1981) for representing classes of similar views.

4. Feature Selection

The recognition strategy, as outlined above, is to locate a key feature and then add one feature at a time until the object can be located reliably and precisely. We refer to the first feature to be located as the *focus feature,* which is consistent with the terminology used in the two-dimensional local-feature-focus method (Bolles and Cain 1982). The selection of the focus feature is a function of several factors. Some of these factors are the uniqueness of the feature, its expected contribution, the cost of detecting it, and the likelihood of detection. Some features are inherently easier to find than others. If two features provide information that is essentially of equal value, but one feature

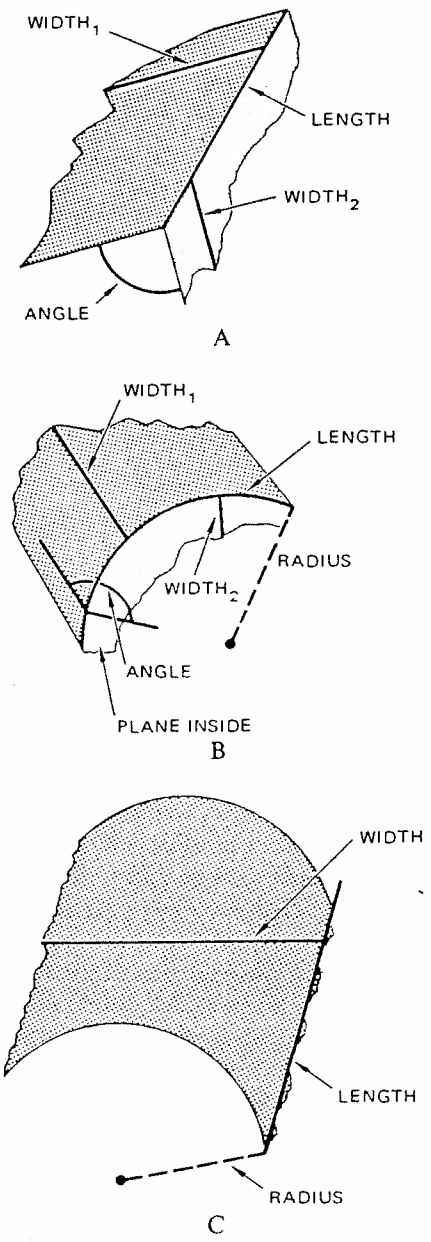


is easier to find, it should be ranked ahead of the other one. If a feature is often missed by the feature detectors, its rank should be lower than for one that is consistently found.

Selection of the second, third, or fourth features is even more complex than the first because it is a function of the set of features already detected as well as its own characteristics. For example, if one or more features have been found, some may be eliminated from the list to be considered because they are on the side of the object away from the sensor. Faugeras et al. (1983) have used this type of reasoning to reduce their tree searches.

In the 3DPO system, we concentrate on edges because they contain more information than surface patches and are relatively easy to detect in range data. We have worked with three types of edges: straight dihedrals, circular dihedrals, and straight tangentials (see Fig. 7). Each type has a set of intrinsic properties that can be used to identify matching model edges. For example, a straight dihedral has its length, the size of the included angle (if both surfaces are detected), and the properties of the adjacent surfaces, such as their widths and areas (see Fig. 8A). Circular dihedrals have the same properties as straight dihedrals plus two additional ones: the radius of the circle and the binary property of whether or not the planar surface is inside or outside the circle (see Fig. 8B). Note that it is possible to establish these last two properties even if only

Fig. 8. Properties of the three edge types. A. Straight dihedral. B. Circular dihedral. C. Straight tangential.



one surface is detected. Straight tangential edges (see Fig. 8C) have fewer intrinsic properties than circular dihedrals. Since range data are noisiest on surfaces curving away from the sensor, straight tangential edges are difficult to locate. As a general rule, circular dihedrals are more distinct and easier to find than the two types of straight edges.

There are four contributions a new feature can make in the "grow-a-match approach" that we are pursuing. It can

1. reduce the number of interpretations for a cluster of located features,
2. verify an interpretation,
3. determine some unknown degrees of freedom associated with an interpretation,
4. increase the precision with which the degrees of freedom can be computed.

A single feature can make more than one contribution. For example, a new feature could verify an interpretation and increase the precision.

An unconstrained object in a jumble has six degrees of freedom associated with its position and orientation, three displacements, and three rotations. Different types of object features constrain different degrees of freedom. For example, a circular dihedral determines all but one of the object's six degrees of freedom. The only unknown is the rotation about the axis of the cylinder. Locating a straight dihedral on an object also determines five degrees of freedom, except that there is a binary choice of the orientation along the edge. The one undetermined degree of freedom is the position of the edge along the matching line. A straight tangential determines only four degrees of freedom because the object can rotate about the cylinder and slide up and down the matching edge. Thus, circular dihedrals, in addition to being more distinct than the straight edges, also provide slightly more constraints on the object's position and orientation.

Given the task of recognizing and locating an object or set of objects, the 3DPO system divides the set of object features into subsets with similar intrinsic properties and applies a different matching strategy for each subset. Thus, if a 2-in. circular dihedral is found as the focus feature, one strategy is adopted; if a 1-in. circular dihedral is found, a different strategy is used. The strategies in the current system are determined interactively and are implemented as procedures. As mentioned earlier, we plan to develop an automatic selection procedure that builds trees of features for subsequent interpretation by a general-purpose procedure.

The circular dihedrals are the best focus features for the casting in Fig. 1 because their intrinsic properties

Table 1. Table of Circular Arcs

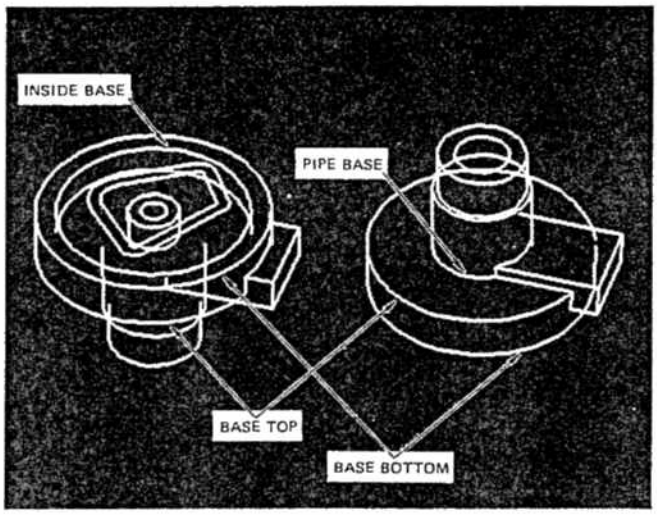
Name	Radius	Plane	Dihedral Angle	Arc Angle	Length	Plane Width	Cylinder Width
Base-bottom	2.30	inside	90	360	14.4	0.4	1.0
Base-top	2.30	inside	90	317	12.7	1.3	1.0
Shelf-bottom	2.30	outside	−90	43	1.7	0.4	1.2
Inside-base	1.90	outside	90	360	11.9	0.4	0.7
Pipe-shoulder	1.00	inside	90	360	6.3	0.1	1.8
Pipe-top	0.90	inside	90	360	5.7	0.3	0.9
Pipe-base	1.00	outside	−90	243	4.2	1.3	1.8
Pipe-on-shelf	1.00	outside	−90	116	2.0	1.6	1.5
Pipe-joint	0.90	outside	−90	360	5.7	0.1	0.9
Corner-1	0.68	inside	90	100	1.2	0.2	0.7
Small-cylinder	0.50	inside	90	360	3.1	0.2	0.7
Corner-2	0.38	inside	90	89	0.6	0.2	0.7
Corner-3	0.33	inside	90	103	0.6	0.2	0.7
Inside-pipe	0.55	outside	90	360	3.5	0.3	2.6
Corner-4	0.43	outside	90	100	0.7	0.2	0.7

generally reduce the number of possible interpretations to one or two, even though the casting has more circular edges than straight edges. Table 1 lists 15 of the circular arcs on the casting and their intrinsic properties. (The other 17 arcs are not included because they are either too small to detect or are not visible.) The horizontal rules in the table delimit subsets of features that have similar values for the first two properties listed across the top of the table. Note, for example, that there are four circles with radii of approximately 2 in., but they are divided into two subsets because two of them have a planar surface inside the circle and two of them have a planar surface on the outside. The first two properties are used because they can be computed even if only one of the surfaces adjacent to the edge is visible. Table 1 illustrates the importance of knowing the sizes of model features and the ability of the feature detection procedures to classify features according to their sizes.

The 3DPO system employs a different strategy for each of the six subsets of features in Table 1. Each strategy depends on the number of features in the subset and the structure of the object in the vicinity of the feature. The strategy for the first subset is to try to ascertain which one of the two features has been found. If the surfaces adjacent to the edge have been located, their respective properties suffice to differentiate the two. If not, the system tries to find either the *pipe-base arc* or the *inside-base arc* (shown in Fig. 9), either of which would identify the focus feature. Once this has been done, the next subgoal of the strategy is to compute the rotation about the axis of the cylinder. This is done typically by locating some of the straight dihedral edges shown in Fig. 9. The strategy starts with a focus feature that is relatively distinct; next it locates a second feature to identify the focus feature; then it finds a feature or two to determine the final degree of freedom.

These edge-based strategies are made possible by the fact that we can reliably locate edges and measure properties of the adjacent surfaces. Since our data are gathered by a triangulation system, they rarely contain more than two surfaces that are connected directly. Usually a "missing data" area intervenes between surfaces. Therefore, any attempt to grow matches topologically beyond a couple of surfaces is likely to fail.

Fig. 9. Key features of the casting.



In the past, most vision researchers have avoided objects with several features because they wanted to keep the images as simple as possible. We, on the other hand, have found that an abundance of features is generally helpful because it means that there is almost always a nearby feature to help identify one for which there are many interpretations. If all the objects are parallelepipeds, all the features look the same. The nearby features are not much help either, since they all look the same too. However, if an object has 30 different features, finding one is often enough to form a hypothesis—while finding two generally suffices to both form a hypothesis and verify it. The fact that the average depth of a strategy tree is about three or four features indicates that the features on the casting provide significant information.

5. Data Acquisition

We use the White Scanner, built by Technical Arts, Inc. of Seattle, Washington, to gather our range data. It is a triangulation device that projects a plane of light onto the scene, as shown in Fig. 10. It measures the x, y, z components and intensity of points along the intersection of the light plane and the objects in the scene. To gather a "dense" range image, the plane is scanned across the scene and the image built up one line at a time. Figure 11 is an overview of the work station showing the laser and the camera used to gather raw data for processing by the White Scanner computer.

Figure 12 shows the data obtained by this sensor. Figure 12A contains the intensity data. Figure 12B is the z component of the range data. Figure 12C is the x component. (The y component is not shown.) Figure 12D is the code image. The coordinate system of the data has its origin on the table top and the z axis pointing up toward the sensor. Thus, z values correspond to heights above the table. The data are encoded in these images so that larger values are lighter, which means that the z component is encoded so that higher points are lighter. The black regions in the images are areas of "no data" that are usually due to occlusion; the camera cannot see the intersection of the light plane and the objects in the scene. The code image specifies the sensor's behavior at each point. It indicates such things as no visible stripe, multiple visible stripes, and blooming (which we will explain shortly).

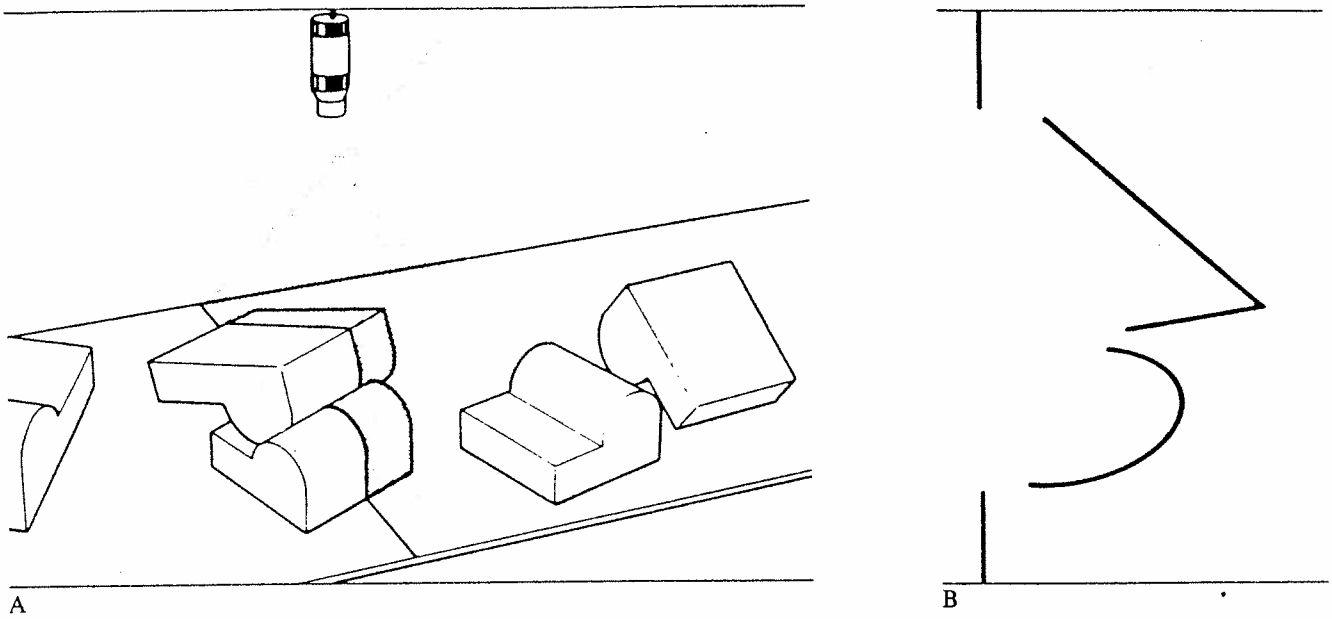
It takes approximately five minutes to gather a 240×240 dense range image, such as the one whose components are shown in Figure 12. The field of view of the sensor is approximately 12 in $\times$ 12 in. The relative precision of the height measurements is about .020 of an inch. At the time we took these images we had a problem with the laser light plane drifting relative to the camera, which had the effect of slightly reducing the absolute precision.

Since the surfaces of the castings are dull (i.e., they have a large Lambertian component in their reflectivity function), the sensor is able to measure data off surface patches whose normals point as much as 45 or 50 degrees away from the line of sight. A shinier surface would reflect less energy back to the camera, which would shrink the range of orientations for measurable surfaces.

Shiny surfaces cause two additional problems: multiple reflections and blooming. *Multiple reflections* occur when the light plane intersects a shiny surface, bounces off it, and intersects another surface. If the camera sees light reflected from both surfaces it cannot distinguish the primary reflection from the secondary one; consequently, the sensor is unable to compute a unique range value.

The second problem, *blooming,* arises when the camera's dynamic range is insufficient to handle the amount of light reflected from the surface. This is

Fig. 10. Diagram of a triangulation-based range sensor. A. Range sensor based on a camera and a plane of light. B. Image of the intersection of the light plane and the objects.



more of a problem for shiny surfaces because they have a large specular component in their reflectivity function, which means they reflect a larger portion of the incoming light at the specular angle. If the camera happens to view a shiny surface patch from the specular direction, it receives significantly more light than at other angles, which may cause it to bloom. The effect of blooming on the range data depends on how the camera blooms. Some cameras, such as a Fairchild CCD, fill a whole column of the image with white if they bloom, causing multiple reflections on all lines. Other cameras, such as a GE CID, just spill white into adjacent pixels. Unfortunately, they don't always do it symmetrically, a condition that would be relatively easy to correct.

6. Feature Detection

We locate local features, such as the arc at the end of a cylinder, by performing the following sequence of operations:

Detect discontinuities in the range data.
Classify them as jump discontinuities, convex discontinuities, shadows, etc.
Discard artifacts, such as shadows.
Link the discontinuities into edge chains.
Classify the chains into subchains lying in planes.
Segment the planar subchains into arcs and lines.
Analyze the surfaces adjacent to the arcs and lines.
Refine the locations of the arcs and lines by intersecting the adjacent surfaces.

We locate discontinuities by analyzing the data one row at a time and then one column at a time. The row analysis detects vertical edges, while the column analysis detects horizontal edges. In each direction we mark jump and slope discontinuities. We locate slope discontinuities by recursively partitioning a sequence of contiguous points into line segments and then evaluating each corner that is introduced. To evaluate a corner, we fit portions of the data on both sides of it with lines and mark it as a slope discontinuity if the angle between the two lines is less than some threshold, such as 150 degrees. In effect, we use the recursive line-fitting procedure to suggest possible discontinuities, which we than evaluate further.

Figure 13 illustrates this edge detection process. Figure 13A shows a typical slice of range data in the coordinate system of the light plane. The ordered list of points is first divided into six sublists of contiguous points, and then line segments are fitted recursively to each sublist. This fitting process introduces the circled

Fig. 11. Overview of the work station.

Fig. 12. Raw data produced by the White Scanner. A. Intensity image. B. Z image. C. X image. D. Code image.

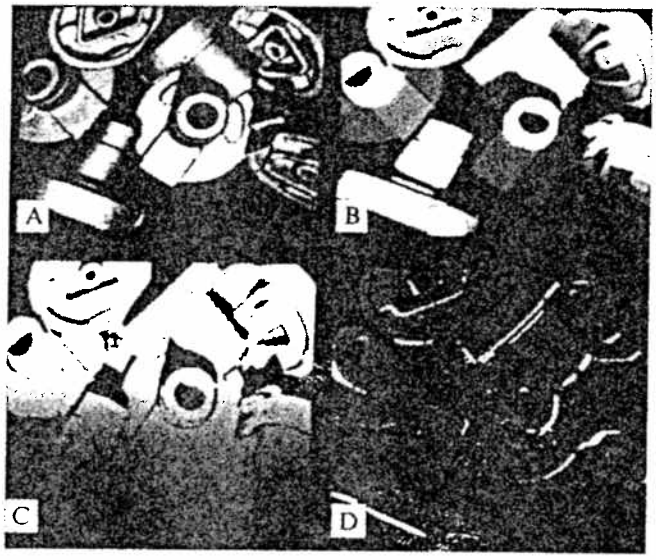
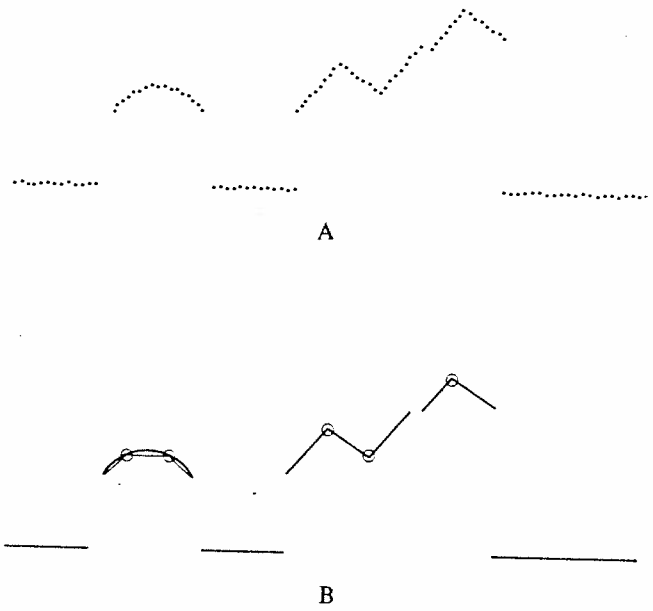


Fig. 13. Edge detection process. A. Slice of range data. B. Corners inserted by the recursive line-fitting routine.



corners shown in Fig. 13B. After these are evaluated, the two corners on the arc are discarded because they are not sharp enough.

The next step is to classify each discontinuity. Figure 14 shows the types of classifications used by the 3DPO system. Figure 15 shows the classifications of the discontinuities illustrated in Fig. 13A. A tangential discontinuity is formed by a surface curving away from the line of sight (i.e., along the side of a cylinder). A shadow discontinuity occurs when one surface occludes another. In Fig. 15 the two shadows furthest to the right are easily detected because they lie directly under a jump discontinuity. The other two shadows are more difficult to detect because they are produced by a surface curving away from the sensor. To do so, the program uses a heuristic for tangential discontinuities that in effect allows for a wider gap between light rays than at jump discontinuities. To detect them, the program looks for discontinuities close to, but not directly under, the tangential discontinuities.

Once the discontinuities have been classified, the artifacts, including shadows and picture edges, are removed from further consideration. This elimination of artifacts typically reduces the number of edge points by about 30 percent. It is, therefore, an important step in the process of locating valid object features.

Fig. 14. Discontinuity classifications along a slice.

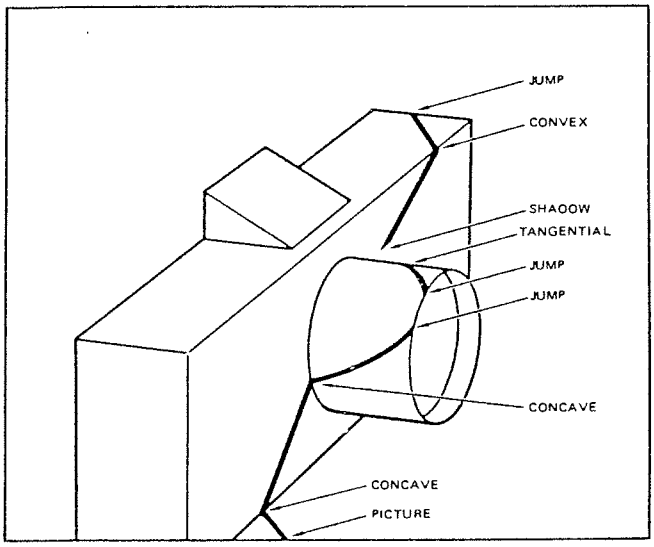
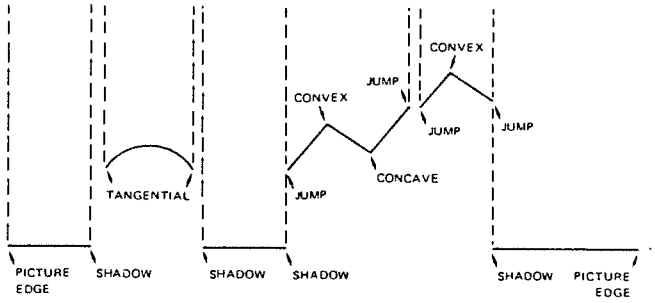


Fig. 15. Classifications of the discontinuities along the slice in Fig. 13A.



After the individual edge points have been detected, classified, and filtered, they are linked together to form edge chains (*edges*). The linking procedure starts with an unattached edge point and tries to form a string of connected edge points. One point is added at a time to the current list, with preference given to points of the same type, points in line with the current edge, and points close to the preceding one. Figure 16 shows the edges located in the data in Fig. 12. Notice that the top edge of the shadow cast by the vertical pipe near the center of the picture has been eliminated by the shadow filter. A small portion at the right of that shadow edge was missed by the filter.

As we mentioned before, the 3DPO system concentrates on three types of object features: straight dihedrals, circular dihedrals, and straight tangentials. Each of these features lies in a plane. A straight dihedral lies in both the planes that intersect to form the edge, a circular dihedral lies in the plane that intersects the cylinder at right angles, and a tangential edge of a cylinder lies in a plane tangent to the cylinder. To locate such features, therefore, the program partitions edges into planar subedges by fitting planes recursively to smaller and smaller portions of the edges until the points along each subedge lie in a plane.

The next step in the feature detection process is to partition the planar subedges into circular arcs and straight lines. To do this, the program maps the 3-D points along a subedge into the two-dimensional coordinates of the plane passing through them, and then partitions the two-dimensional curve into circular arcs and straight lines. It applies a technique similar to Pavlidis's (1982) to accomplish this final segmentation. Figure 17 shows the straight lines and the points along the circular arcs found for the edges in Fig. 16.

The last two steps of the feature detection process are to analyze the surfaces on either side of an edge (if there are data on them) and intersect them to improve the estimates of the edge's location. Each of the surfaces is classified according to its type and a few simple properties are measured, such as the maximum excursion of the surface from the edge. A circular edge is expected to have one planar surface and one cylindrical surface adjacent to it. Straight segments may be the tangential edge of a cylinder, the intersection of two planes, or the intersection of a plane and a cylinder. After the surfaces have been classified, they are refitted with as much data as possible and the updated surface equations are used to improve the estimates of the parameters of the lines and circles. This completes the feature detection process.

The most time-consuming part of the feature detection process is the analysis of the surfaces adjacent to the edges. While it is important to determine surface type, it is less essential that exact fits be produced—rough estimates of the surface parameters may be sufficient. Therefore, it appears that it would be more efficient to develop some techniques that characterize surface types quickly than to spend the time required to perform costly iterative fitting.

Extra features can slow the matching process down a little because they force the program to explore hypotheses that it cannot verify. We have not found this

Fig. 16. Nonshadow range edges.

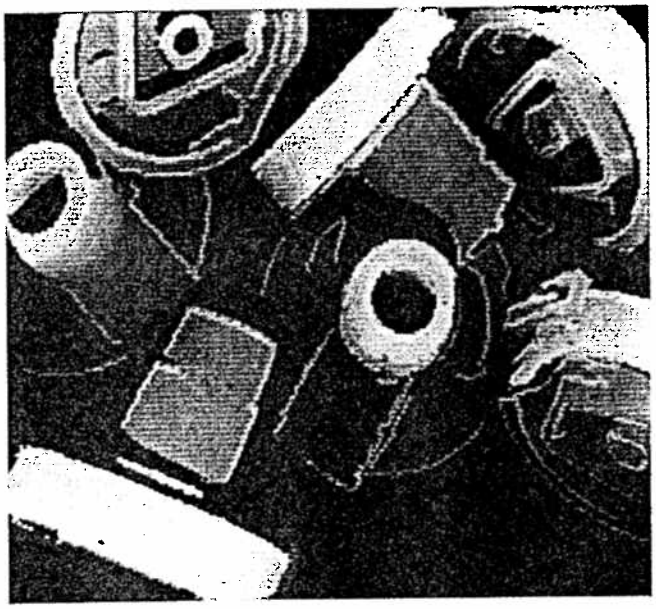
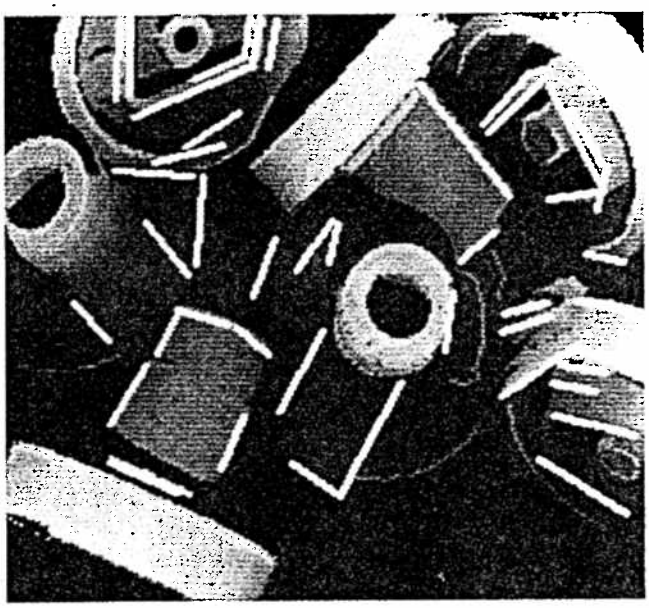


Fig. 17. Circular arcs and straight lines.



to be a significant problem, however. In fact, we usually set relatively low thresholds for accepting features and rely on the high-level system to filter out the extra ones. Low thresholds are beneficial because they let in features that otherwise would have been missed, and the false features are quickly rejected because they do not form clusters of features that are consistent with the objects.

7. Hypothesis Generation

The top-level strategy for hypothesizing object locations tries to grow a match from the most distinctive feature found by the feature detection procedure, then from the second most distinctive feature, and so on. When a hypothesized object is verified, its features are removed from the global list of features, which in effect simplifies the scene to be analyzed. This process of hypothesizing and verifying objects continues until everything in the scene is understood or no more matches can be formed.

The most distinctive features for the casting are the large to medium-sized circular arcs listed in Table 1, followed by the longer tangential edges. Therefore, the program starts its analysis of a scene, such as the one shown in Fig. 17, by looking for the larger circular arcs. When it finds one, it applies the strategy that was established for that type of focus feature by the off-line strategy selection process. Figure 18 shows one of these large arcs with a radius of approximately 2 in. and a planar surface on the outside. This indicates this arc must be either part of the *shelf-bottom* arc or the *inside-base* arc. Since its length is significantly longer than 1.7 in., the program eliminates the shelf-bottom interpretation.

To verify the inside-base interpretation, the strategy associated with this focus feature applies a routine to locate one of the concentric circles on the bottom of the casting. This is essentially a verification routine that tries to find a circle with a specific radius at a specific 3-D position and orientation. To do this, the strategy first looks to see if such a feature has already been detected. If so, the inside-base interpretation is verified. If not, the program analyzes the range data for evidence of the circle. Figure 19 shows two concentric circles located in this manner. The small one had already been detected by the initial feature detection process. The larger one was detected by reanalysis of the data. The system uses the data along these circles to refit the plane passing through them and to improve its estimate of the location of their common center.

Fig. 18. A circle fitted to an arc.

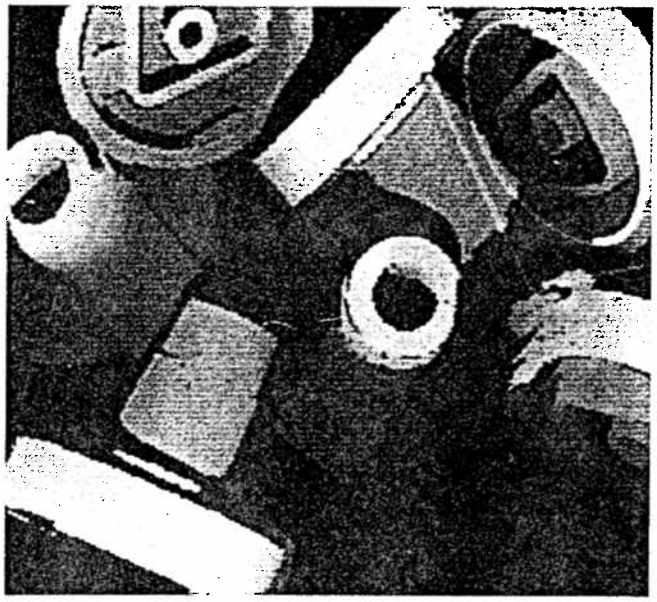


Fig. 19. Concentric circles.

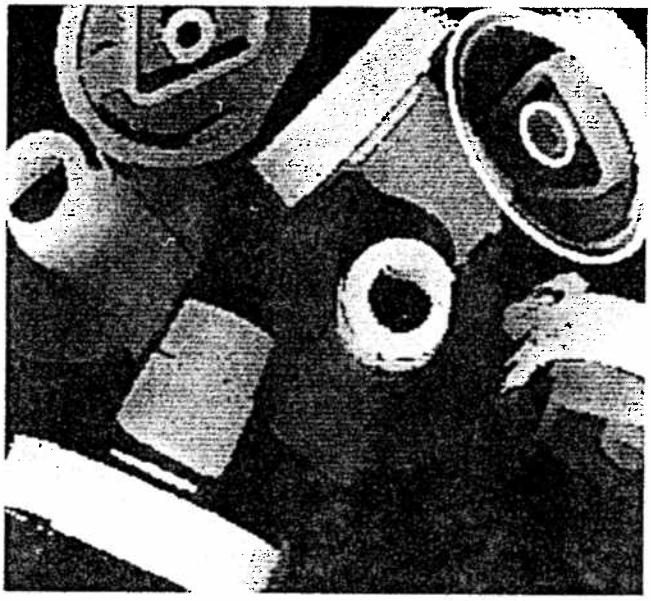
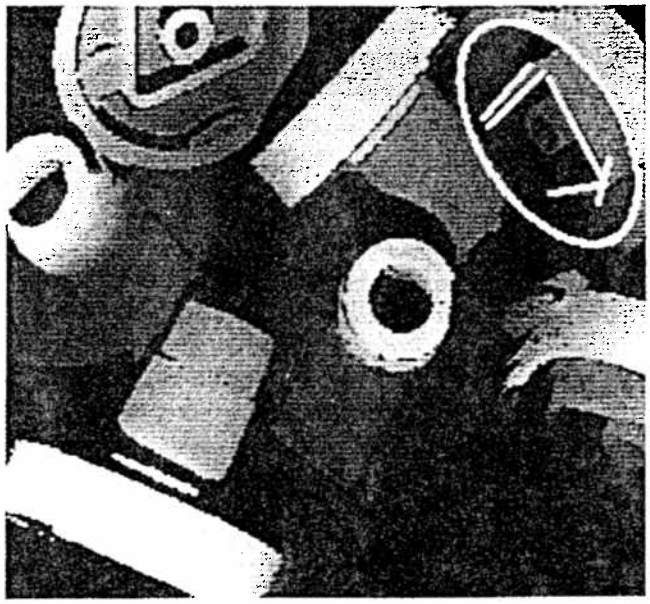


Fig. 20. Candidate lines for determining the rotation.



These parameters determine five of the casting's six degrees of freedom.

To determine the sixth degree of freedom, the program has to locate a feature that is not symmetric about the axis of the cylinder. The best features for this purpose are the line segments that are part of the design on the bottom of the casting, which unfortunately are all quite similar. So, instead of selecting one feature for the next branch of the tree search, the strategy includes a multifeature step that involves locating several lines and determining the largest subset of them that is consistent with the model. The maximal-clique algorithm used to find the largest subset of consistent matches is itself a tree search (Bolles 1979). The system views the location of this subset as only one step in the strategy, however, so that it can evaluate the success of the maximal-clique algorithm separately.

Figure 20 shows the line segments that are in the same plane as the circle and may be part of the pattern. There are five lines, each with two or three interpretations. These lines imply a graph containing 12 nodes to be analyzed by the maximal-clique algorithm. Figure 21 shows the final hypothesis, which is based on the initial circle and four of the line segments.

In Fig. 22, another circular arc is shown with its set of line segments for computing the rotation. In this case, the maximal-clique algorithm finds three line segments that are mutually compatible with the circle. Figure 23 depicts the final hypothesis.

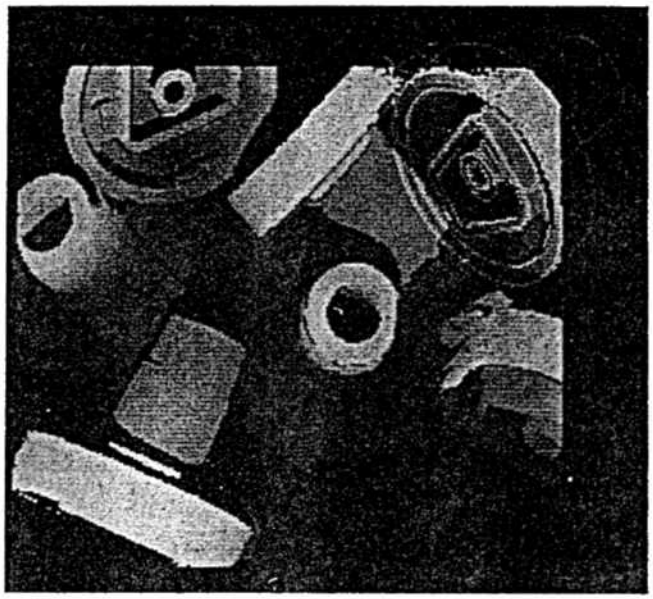
The program arrived at five hypotheses by starting with circular arcs (see Fig. 24). It then started considering tangential edges. The strategy for dealing with

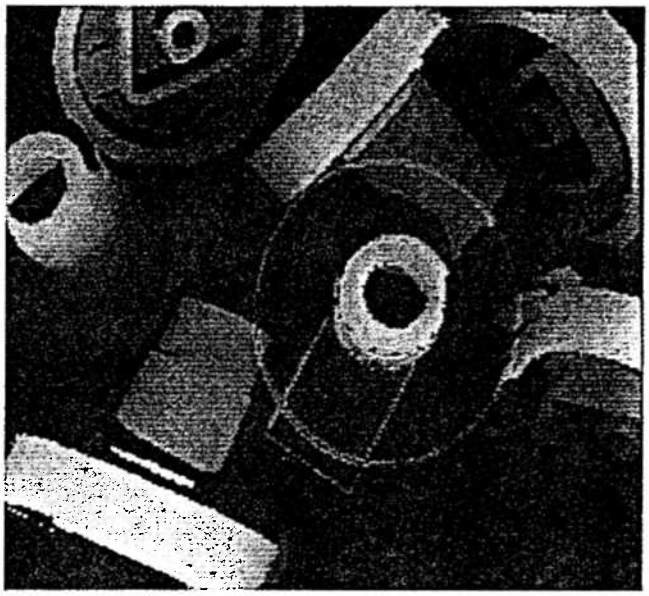
Fig. 21. Hypothesized casting.

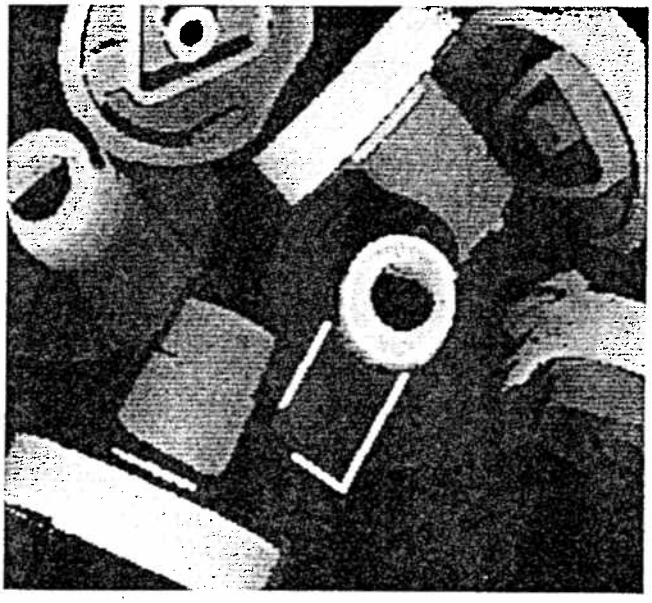
Fig. 22. Circle and candidate lines.

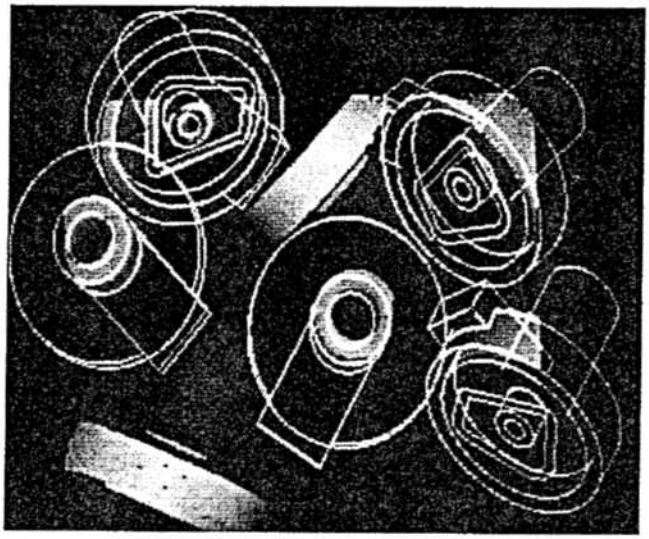
Fig. 23. Hypothesized casting.

Fig. 24. Five hypotheses derived from arcs.









tangential edges is to locate other cylinders and circles that have axes almost collinear with the initial cylinder. Locating one additional feature is generally enough to compute five degrees of freedom. Thus the strategy is the same as for circles: locate a feature to determine the sixth degree of freedom. Figure 25 shows a cylinder and a compatible circle. Figure 26 shows a hypothesis based on the five degrees of freedom computed from the cylinder and circle. The system missed the seventh casting in the scene because it only found one of its features.

The tests we use for checking the compatibility of

Fig. 25. Cylinder and compatible circle.

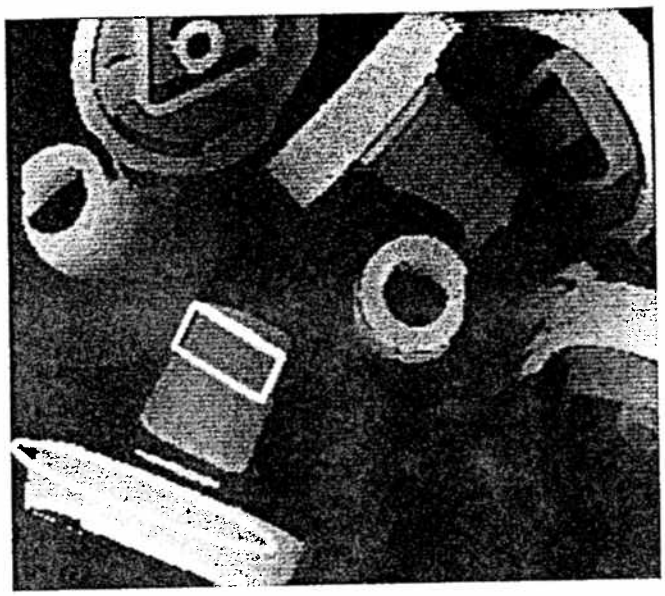
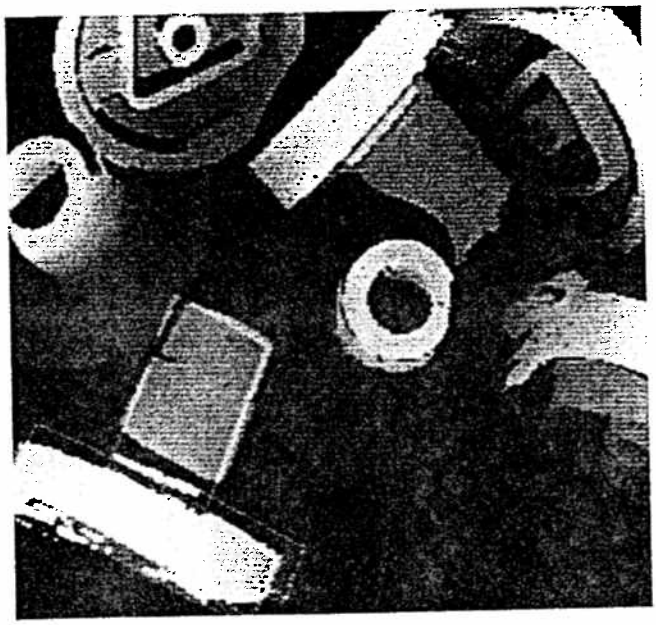


Fig. 26. Hypothesized casting.



one feature with another are extensions of the two-dimensional point-to-point tests used in the local-feature-focus system (Bolles and Cain 1982) and the 3-D point-to-plane tests used by Grimson and Lozano-Perez (1984). Since the observed features are segments of lines, circles, and cylinders, the tests are segment-to-segment matches in the sense that they can use the lengths of the features to constrain the extent to which one feature can slide along a matching feature. Long features constrain the sliding more than short ones. So far we have not tried to develop a minimum set of tests. We have simply implemented a set of inexpensive tests to eliminate obvious mismatches.

Industrial systems have to be robust to be practical. For the 3DPO system, this means that the high-level system has to work even when the low-level system misses features and finds extra ones. It recovers from most missing features by focusing on other features of the object. If several features on an object are missed or are obscured by other objects, the system may take a while to recognize the object because it has to proceed through its list of features. Of course, if all the major features are missed, the system inevitably will miss the object altogether.

8. Hypothesis Verification

After the system hypothesizes an object's pose, there are three things it can do to increase an arm's chances of acquiring the object correctly. It can

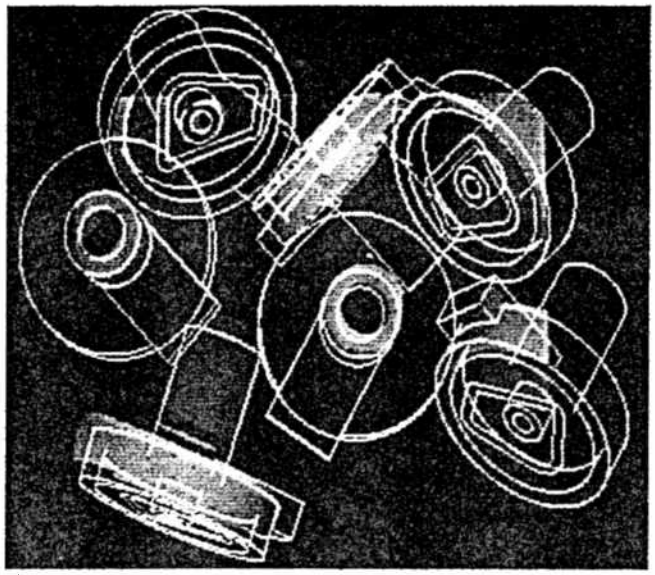
1. verify the hypothesis,
2. refine the pose estimate, and
3. determine the object's configuration.

In this section we describe a verification technique. In the next section we show how it can be extended to determine which objects are on top of the jumble. Although the pose refinement step is an essential component of a complete system, it will not be discussed here. We simply observe that Rutkowski and Benton (1984) have described an approach to pose refinement that looks promising.

There is only one way to check a hypothesis: compare predictions with data gathered from the scene. Predictions may differ in type, but the process of checking a hypothesis is nonetheless identical. If too many predictions disagree with the data, the hypothesis is rejected.

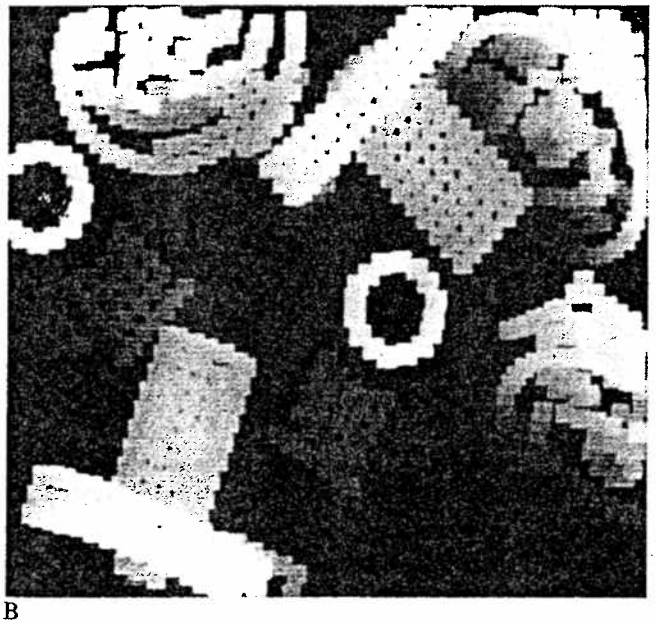
Predictions can be object features (e.g., holes, corners, or surface patches) or they can be sensor data

Fig. 27. Range prediction procedure. A. Seven hypothesized castings. B. Synthetic height image. C. Measured height image.



A

C



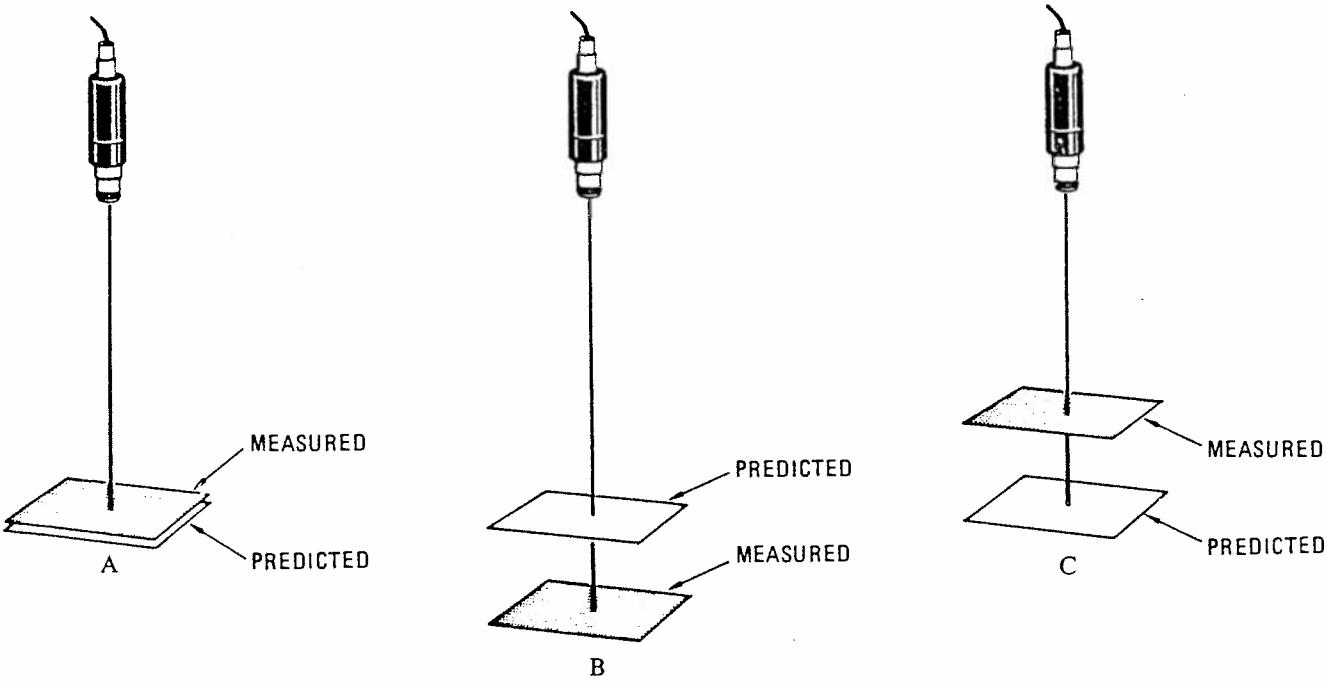
B

(e.g., the expected intensity of a point on the surface of an object). Most matching strategies have feature-level verification built into the matching process. They use the first few features to narrow the number of possible matches down to one—which is equivalent to making a hypothesis—and then match additional features to increase confidence in that hypothesis. These systems generally report the hypotheses that contain the most matching features to be the best matches.

Data-level comparison is another type of verification that can be done. With this kind of comparison, the program employs a hypothesis to predict the data that would have been measured by the sensor if the object had been in the hypothesized pose and then compares these predictions with the data actually measured by the sensor. In this paper, we describe data-level techniques that complement traditional feature-level techniques. We concentrate on range data because they encode the geometry of an object directly and are relatively easy to predict.

The 3DPO system forms one hypothesis at a time (such as the one shown in Fig. 21) and then tries to verify it. To check a hypothesis, the program predicts the range data, compares it with the actual data, and then makes decisions based on the correlation between the predicted and actual data. The predictions are an estimate of what the sensor would have seen if the objects had been in the hypothesized poses. To make the predictions, the program uses the planar-patch model shown in Fig. 6. Given a hypothesis or set of

Fig. 28. Possible relationships between predicted and measured range data. A. Measured data are approximately equal to predicted data. B. Measured data are significantly farther from the sensor than predicted. C. Measured data are significantly closer to the sensor than predicted.



hypotheses, the program builds an image by painting in regions corresponding to the surface patches in the scene that are closest to the sensor. This is essentially the same as the z buffer technique used by a computer graphics system. Figure 27B shows a predicted range image that corresponds to the measured data in Fig. 27C. It was produced from the seven hypotheses shown in Fig. 27A. (The seventh casting, which was missed by the hypothesis generation procedure, was added interactively to the scene description to illustrate this range prediction procedure and the configuration understanding procedure discussed in the next section.)

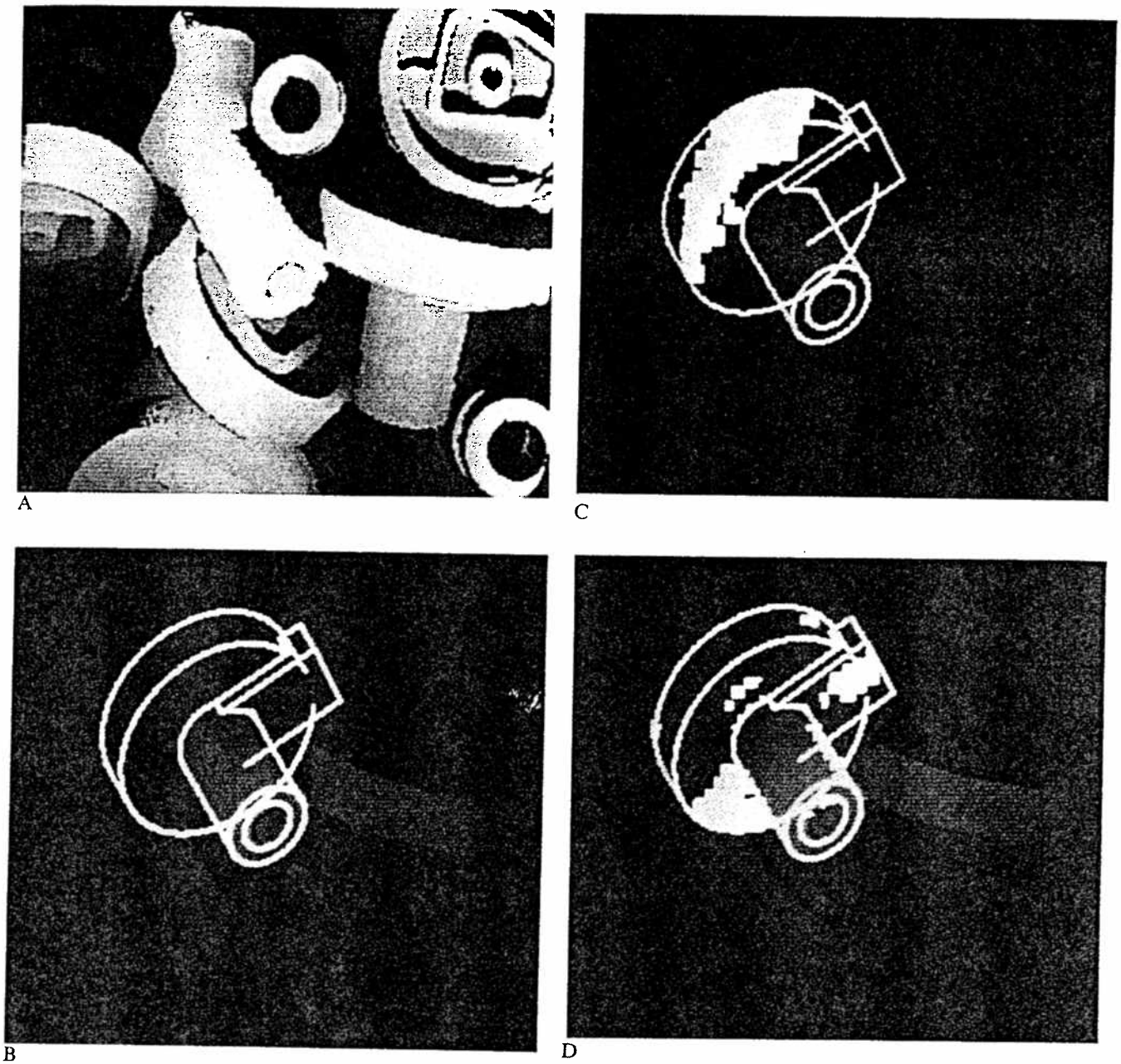
When a measured range value is compared with a predicted value, three situations can occur:

1. The measured data are approximately equal to the predicted data.
2. The measured data are significantly farther from the sensor than the predicted data.
3. The measured data are significantly closer to the sensor than the predicted data.

These situations are presented in Fig. 28. In Fig. 28A, the measured data agree with the prediction, and the system increases its confidence in the hypothesis that led to that prediction. In Fig. 28B, the sensor appears to have seen through the predicted object because the measured data are farther from the sensor than the predicted data. This is strong negative evidence, since the objects are assumed to be opaque. In Fig. 28C, there appears to be an object between the sensor and the hypothesized object. By itself, this situation is inconclusive. It neither supports nor refutes the hypothesis. Given a set of hypothesized objects for a scene, however, it is possible to determine whether or not the measured data belong to any one of them. If so, the program marks the data as explained and treats them as neutral evidence. If not, it marks the data as unexplained and treats them as weak negative evidence. The three types of predictions are referred to as *positive, negative,* and *neutral evidence,* respectively.

To illustrate the classification of predicted surface patches, let us consider the range image in Fig. 29A. In the middle of the image there is a casting that is different from the model. It has a pipelike portion that is about the same diameter as the one on the expected casting, but is significantly longer. Let us assume that the system finds the end of that pipe and hypothesizes a pose, such as the one shown in Fig. 29B. Since the hypothesis is based on the data near the end of the pipe, the predictions in that region agree with the

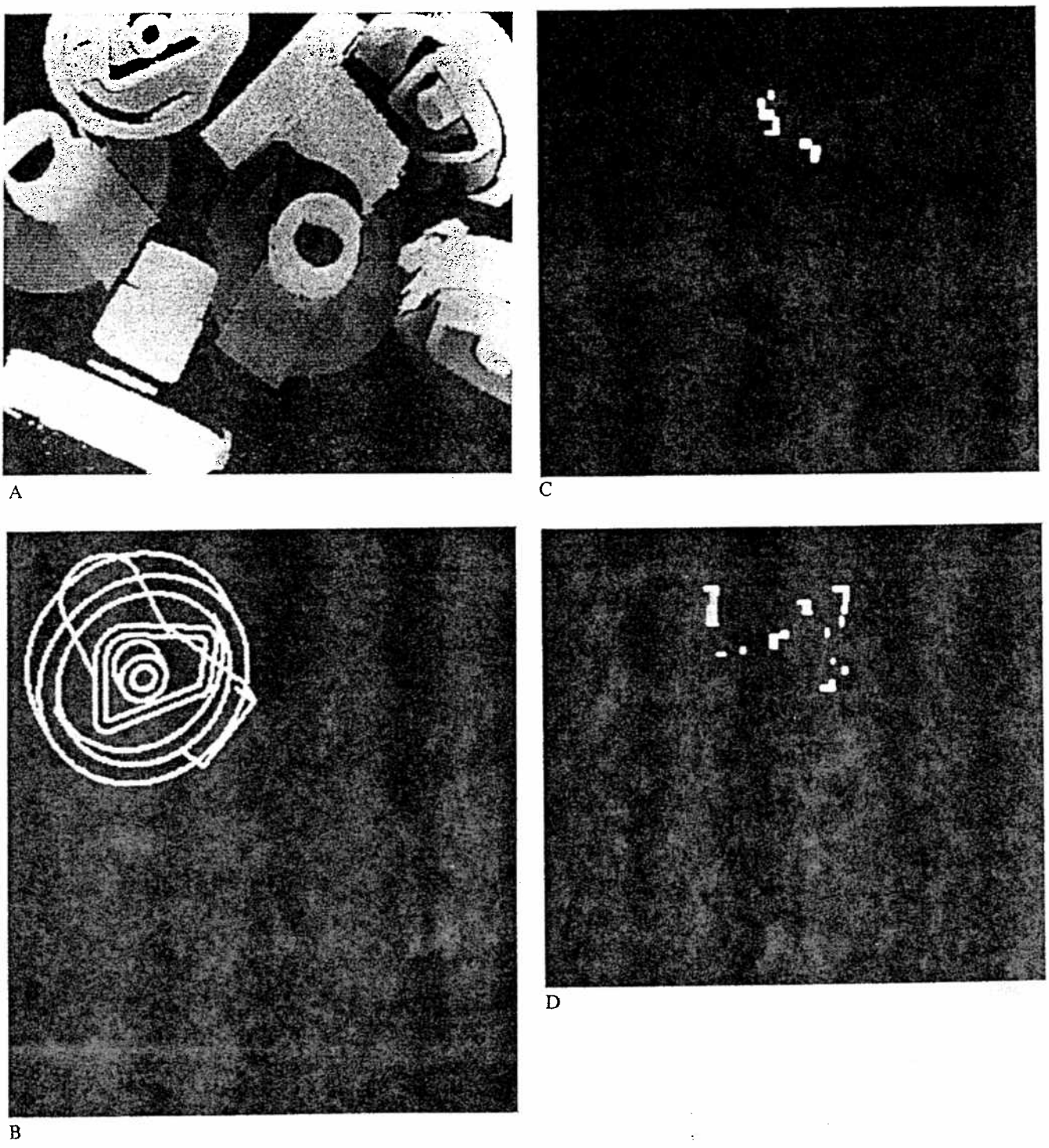
Fig. 29. A bad hypothesis. A. Measured range data. B. Hypothesized casting. C. Negative evidence. D. Neutral evidence.



measured data. In some of the other regions, however, the predictions disagree. Figure 29C shows the negative evidence (i.e., the predicted data that are farther from the sensor than the measured data). Figure 29D displays the neutral evidence (i.e., the predicted data that are closer than the measured data). This hypothesis would be rejected because of the large region of negative evidence.

The 3DPO system makes hypotheses one at a time, checking each individually as it is formed. Figure 30 depicts a good hypothesis. Figure 30A shows the measured range data, Fig. 30B the hypothesis, Fig. 30C the

Fig. 30. A good hypothesis. A. Measured range data. B. Hypothesized casting. C. Negative evidence. D. Neutral evidence.



negative evidence, and Fig. 30D the neutral evidence. There are several small discrepancies along the edges of the object that are due to a slight misalignment of the hypothesis. Because they are all small, however, the system still accepts this hypothesis. Nevertheless, the large number of edge effects emphasizes the need for a technique to refine pose estimates. A better pose estimate would eliminate most of the discrepancies.

The feature-based approach to hypothesis verification is convenient because the features have already been detected as part of the matching process; essentially, all that remains to be done is to match a few additional features. There are at least two disadvantages, however. First, this method requires that an object possess several distinct features (e.g., it doesn't work very well for smooth objects such as an apple). Second, it places a burden on the feature detectors in that they must locate all the features and describe them in a canonical way. If any features are missed or described incorrectly, a more complicated matching strategy will be necessary. For example, if a feature is missed, the program may have to apply a specially tuned feature detector to find the feature in order to include it in the verification process. (We have used this approach in the 3DPO system to find such things as concentric circles.) If a feature detector happens to describe the low-level data in a way that is different from what was expected, the matcher will have to be smart enough to recognize alternative descriptions. For example, if the line-and-arc fitter should segment an arc predicted from the model into a sequence of short line segments, the matcher would have to recognize the pieces as parts of the arc.

One of the advantages of a data-level approach to hypothesis verification is that it is a homogeneous process that works for all types of objects. Another advantage is that it produces explanations in terms of regions, which is a convenient form for deciding how much of a scene is explained by a set of hypotheses.

One disadvantage of a data-level approach to hypothesis verification is that it requires a detailed model of the physics of the sensor. While this is relatively straightforward for range data, our current capabilities cannot handle intensity images. We can of course predict the locations of edges in an intensity image, but even then we lack models of the detectors employed to locate the edges. Since each detector has side effects, such as displacing edges and inserting new ones, precise predictions are still not possible.

A second disadvantage of data-level verification is that it is sensitive to misalignment. Slight offsets lead to discrepancies along the edges of an object. One way to reduce misalignment is to apply an iterative technique that assigns measured data points to surfaces of the object, uses these assignments to update the pose estimate, and then repeats the process until it minimizes the sum of the errors. Given a good estimate of the pose, the program can reject the hypothesis if the sum of the errors is too large, or, alternatively, it can make the data-level comparison perform a more structural evaluation of the match.

In the future we plan to investigate ways of combining feature-level verification with data-level techniques. This combination would utilize features to help develop a region-based explanation of a scene. For example, the location of a feature could be used for local correction of a global pose estimate so as to avoid the edge effects resulting from an unguided data-level comparison. Such a system would also provide a way of extending the technique to gray-scale analysis when it is impossible to predict absolute intensity values yet possible to predict intensity edges and approximate intensity values.

9. Configuration Understanding

There are several reasons why it is better to pick up an object from the top of a pile than one that is partially buried. First, the topmost object usually has more surfaces exposed and hence provides more ways in which it can be grasped. Second, its relatively accessible location minimizes the force required to extract it. This also tends to minimize the forces that might change the object's pose in the hand of the robot. This is important because the goal of the 3DPO system is to ascertain the pose of an object before grasping it so that the arm can select a grasping position that will be compatible with the pose required at the time the object is set down. If the pose of the object in the hand changes as the object is being pulled out of the pile, the system loses some essential positional information. A third reason for selecting the top object is that its removal generally causes minimal disruption to the

Fig. 31. One object occluding another.

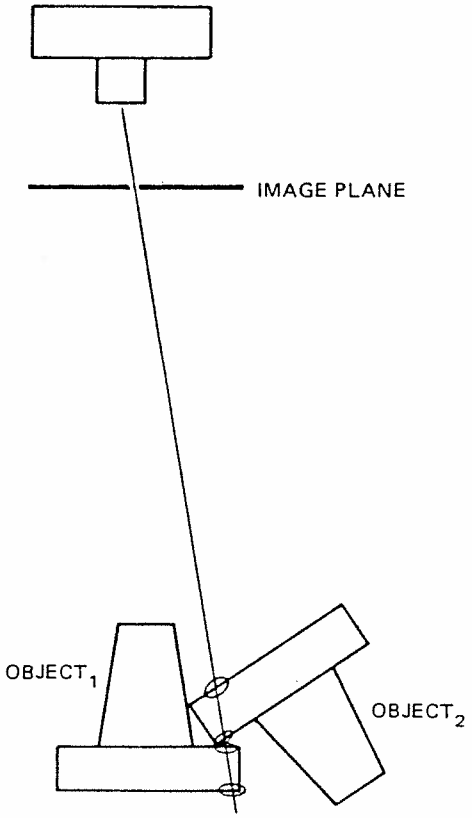
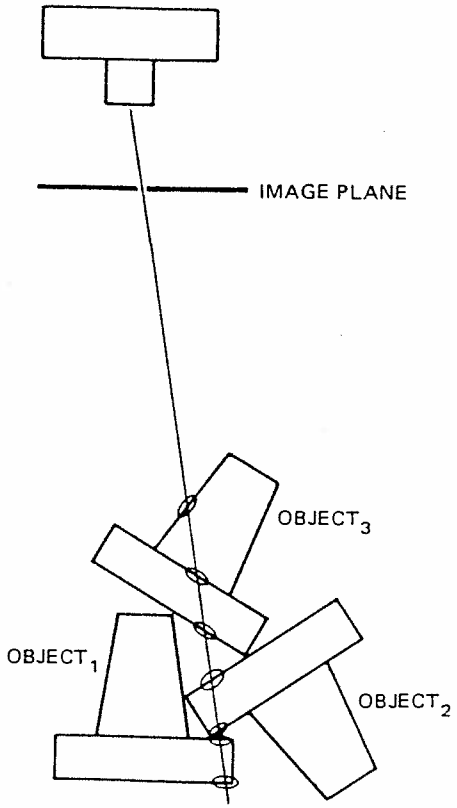


Fig. 32. Stack of three objects.



rest of the pile, thus simplifying the analysis necessary for selection of the next object to be acquired.

The 3DPO system determines which object is at the top of a pile by predicting a range image from all the verified hypotheses, tagging each projected range value with the number of the hypothesized object it was derived from, and then checking to see which one is on top whenever two predictions are made at the same place in the image. As it makes predictions and compares them with the partially completed image, the program gathers statistics on the number of overlapping patches between the members of each object pair. After completing this analysis, it uses the statistics to construct a graph that represents the significant occlusions.

Figure 31 illustrates a typical occlusion. In this example, at least four range values would be predicted along the indicated ray, which corresponds to one pixel in the synthetic image. It is easy to determine that at this particular pixel object 2 is on top of object 1. When three or more objects are present along a ray, the amount of configurational information extracted depends on the amount of data the program stores for each pixel. The current program keeps track of the range value closest to the sensor and the object to which it belongs. When a new range value is predicted, the program checks the synthetic image to ascertain whether a value has already been predicted for that pixel. If not, it inserts one. If so, it compares the object identification numbers of the old and new predictions. If they are the same, the program updates the predicted range value if necessary and continues. If the objects are different, it notes which of them is on top and updates both the range value and the object number.

This process gathers all the occlusion information if there are no more than two objects along any one ray. If there are three or more, however, the occlusion relationships obtained depend on the order in which the hypothesized objects are processed. For example, let us consider Fig. 32. If object 1 is processed by the range prediction software first, object 2 next, and ob-

Fig. 33. Graph of the "on top of" relationships for the hypotheses in Fig. 27C.

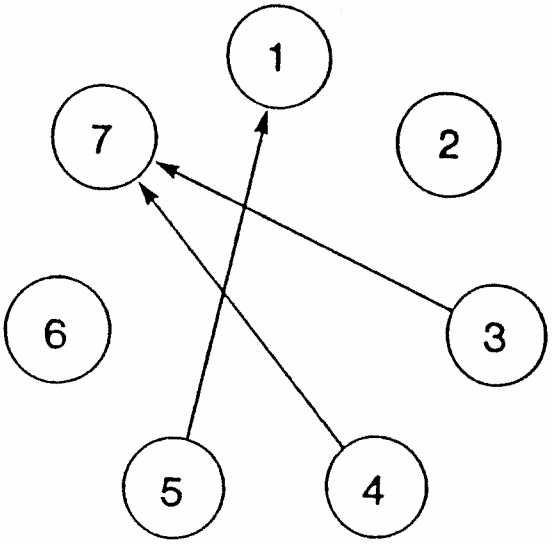
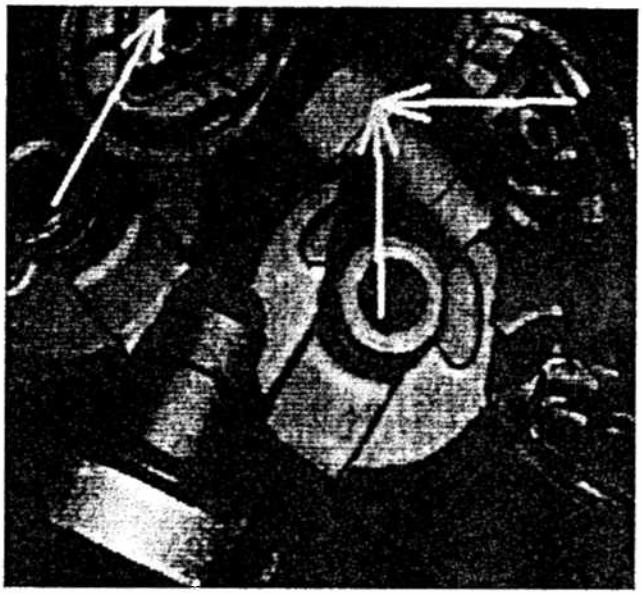


Fig. 34. Arrows point from occluded castings to occluding ones.



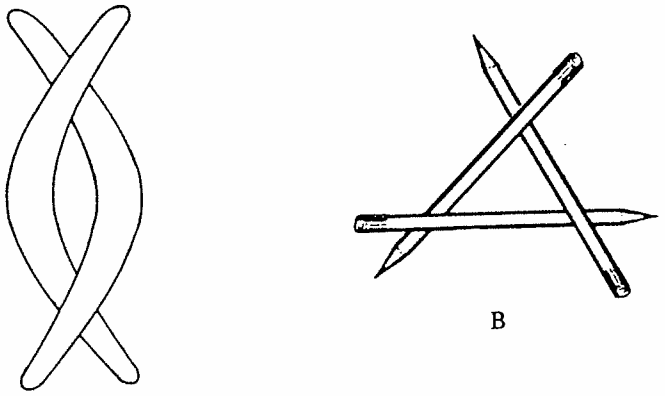
ject 3 last, the relationships that would then be computed are that object 2 is on top of object 1 and that object 3 is on top of object 2. The fact that object 3 is on top of object 1 is missed. If the program kept track of all range values along each ray, together with their objects, it would be possible to compute all such relationships. However, the data structures required to store this information are unwieldy. Fortunately, since the two top objects usually occlude lower objects completely, stacks of three or more objects are not often detected in range images. In addition, missing an occlusion relationship at a pixel is not usually critical because the relationship generally occurs at other places in the image where the program can detect it. In this case, missing one occurrence of a relationship simply reduces the estimate of the amount of overlap between the two objects—which is significantly less detrimental than missing the crucial fact that they are overlapping.

After the program has gathered statistics about occlusions in the image, it builds a graph to represent object interrelationships. Figure 33 displays the graph constructed for the seven hypotheses of Fig. 27A. Figure 34 shows the information from that graph in terms of arrows superimposed upon the intensity image returned by the range sensor. An arrow points from an occluded object to the occluding one. The arrows indicate, for example, that the object at the top center of the image is lying atop two other objects. The three objects at the butts of the arrows would not be good choices to be picked up first because other objects are known to be on top of them.

There may be situations where no object is on top. All of the objects might be partially occluded by other objects. In that case, the graph would be cyclic. Figure 35 shows two configurations of objects where none of them is on top. In Fig. 35A two concave objects are arranged so that they overlap each other. In Fig. 35B three convex objects are arranged so that all of them are partially occluded.

The information regarding what is on top of what represents the first-level understanding of an object configuration. A second level might be a specification stating exactly which objects are resting on or leaning against other objects and where they touch one another. This information would make it possible to perform a more detailed analysis of which objects would be moved in the course of extracting one of them from the pile. Unfortunately, it appears too difficult to compute these relationships from predicted range images or from 3-D models without a spatial-reasoning system that understands gravity and the geometric constraints associated with mutual contact between objects.

Fig. 35. Cyclic occlusion relationships. A. Two concave objects that overlap each other. B. Three convex objects that overlap one another.



10. Discussion

It is interesting to note that the problem of locating objects with six degrees of freedom in range data is remarkably similar to the problem of using gray-scale images to locate objects constrained to lie on a plane parallel to the image plane. In both cases, one feature determines most of the degrees of freedom. For example, a circular hole in a gray-scale image determines two of the three degrees of freedom. A circular edge in range data determines five of the six degrees of freedom. Another similarity is that a partial topology of features can be determined. In range data, the edge-surface-edge connectivity provides a direct way to grow clusters of related features. In gray-scale images, the corner-line-corner connectivity along edges provides a similar capability.

In the future we plan to continue our investigations into ways of detecting features and growing clusters so that more and more of the recognition process can be shifted from search methods to cluster formation procedures. This shift will lead to an increase in efficiency because it reduces the amount of unconstrained search required to recognize an object. We also plan to explore techniques for analyzing CAD models and selecting recognition strategies and features automatically.

References

Baumgart, B. G. 1972. Winged-edge polyhedron representation. Stanford A.I. Memo Number AIM-179. Stanford, Calif.: Stanford University.

Bolles, R. C. 1979 (Washington, D.C.). Robust feature matching through maximal cliques. *Proc. SPIE's Tech. Symp. on Imaging Applications for Automated Industr. Inspection and Assembly.*

Bolles, R. C., and Cain, R. A. 1982. Recognizing and locating partially visible objects, the local-feature-focus method. *Int. J. Robotics Res.* 1(3):57–82.

Brooks, R. A. 1981. Symbolic reasoning around 3-D models and 2-D images. *Artificial Intell. J.* 17:285–348.

Chakravarty, I. 1982. The use of characteristic views as a basis for recognition of three-dimensional objects. Ph. D. Thesis. Rensselaer Polytechnic Institute.

Duda, R. O., Nitzan, D., and Barrett, P. 1979. Use of range and reflectance data to find planar surface regions. *IEEE Trans. on Pattern Analysis and Machine Intell.* PAMI-1(3):259–271.

Faugeras, O. D., and Hebert, M. Aug., 1983 (Karlsruhe, Germany). A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces. *Proc. 8th IJCAI,* pp. 996–1002.

Faugeras, O. D., et al. 1983. Toward a flexible vision system. In *Robot Vision,* edited by Alan Pugh. United Kingdom: IFS Publications, Ltd.

Grimson, W. E. L., and Lozano-Perez, T. 1984. Model-based recognition and localization from sparse range or tactile data. *Int. J. Robotics Res.* 3(3):3–35.

Koenderink, J. J., and Van Doorn, A. J. 1976. The singularities of the visual mapping. *Biol. Cyber.* 24(1):51–59.

Nevatia, R., and Binford, T. O. 1977. Description and recognition of curved objects. *Artificial Intell. J.* 8:77–98.

Nitzan, D., Brain, A. E., and Duda, R. O. 1977. The measurement and use of registered reflectance and range data in scene analysis. *Proc. IEEE* 65:206–220.

Oshima, M., and Shirai, Y. 1978. A scene description method using three-dimensional information. *Pattern Recognition* 11:9–17.

Pavlidis, T. Oct., 1982 (Munich, Germany). Curve fitting as a pattern recognition problem. *Proc. 6th Conf. on Pattern Recognition.*

Popplestone, R. J., et al. Sept., 1975 (Tbilisi, Georgia, USSR). Forming models of plane-and-cylinder-faceted bodies from light stripes. *Proc. 4th IJCAI,* pp. 664–668.

Rutkowski, W., and Benton, R. Jan., 1984. Determination of object pose by fitting a model to sparse range data. Interim Tech. Report of the Intelligent Task Automation Program. Honeywell, Inc., pp. 6-62–6-98.

Shirai, Y., and Suwa, M. Aug., 1971 (London, England). Recognition of polyhedrons with a range finder. *Proc. 2nd IJCAI,* pp. 80–87.