Planning Sensing Strategies in a Robot Work Cell with Multi-Sensor Capabilities *

S. A. Hutchinson, R. L. Cromwell, and A. C. Kak

Robot Vision Laboratory
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

ABSTRACT

In this paper we present an approach to planning sensing strategies in a robot work cell with multi-sensor capabilities. The system first forms an initial set of object hypotheses by using one of the sensors. Subsequently, the system reasons over different possibilities for selecting the next sensing operation, this being done in a manner so as to maximally disambiguate the initial set of hypotheses. The "next sensing operation" is characterized by both the choice of the sensor and the viewpoint to be used. Aspect graph representation of objects plays a central role in the selection of the viewpoint, these representations being derived automatically by a solid modelling program.

1. Introduction

With current techniques in geometric modeling, it is possible to generate object models with a large number of features and relationships between those features. Likewise, given the current state of computer vision (both 2D and 3D) and tactile sensing, it is possible to derive large feature sets from sensory data. Unfortunately, large feature sets can also require exponential computational resources unless one takes advantage of the fact that most objects can be recognized by a few landmarks. The problem then becomes one of developing computer procedures capable of analyzing geometric models to yield the most discriminating feature sets. In solving this problem, one has to bear in mind that in the robotic cells of today we have available to us a variety of sensors, each capable of measuring a different attribute of the object.

For it to be useful to robotic assembly, we need to add another dimension to the problem as stated above. Say, we have a robot trying to determine the identities of the objects in its work area. The robot should only invoke those sensory operations that are most relevant to the disambiguation of whatever hypotheses the robot might entertain about the identities of those objects. Therefore, the most discriminating features invoked by the robot must be determined at run time and, of course, must make maximum advantage of all the sensors that are available.

If we limited ourselves to just vision sensing and if the run-time capability was not important, problems of this sort have recently been solved by a number of researchers, most notably Ikeuchi [8] and Hanson and Henderson [6]. Ikeuchi's work is based on the automatic synthesis of *interpretation trees* which are used to guide feature selection. In Ikeuchi's approach, the higher level nodes in the interpretation tree yield the aspect of the object, and then the lower level nodes are used for computing the precise pose of the object. This scheme makes use of the fact that for most objects the set of features useful for discriminating between aspects differs from the set of features useful for determining the exact pose once the aspect has been determined.

In the work reported by Hanson and Henderson, a set of filters is used to select the best identifying features (based on rarity, robustness, cost, etc.) for each aspect. These features and their associated aspects are compiled into a *strategy tree* which, in purpose, is similar to Ikeuchi's interpretation tree. The strategy tree has two levels. Each node at the first level allows aspect hypotheses to be invoked on the basis of certain features and their values. For each hypothesis at a first level node, there exists a *Corroborating Evidence Subtree*, which is used to guide the search for evidence that supports that hypothesis and for carrying out the computations for geometric data for determining the object's pose.

The work that we present in this paper extends the above cited work by giving the system the ability to use multiple aspects and different sensors for the identification of an object and the computation of its pose. The sensory types currently incorporated in the system include a 3D range scanner, 2D overhead cameras, a manipulator held 2D camera, a Force/Torque wrist-mounted sensor, and also the manipulator fingers for estimating the grasp width. These sensors can be used to examine objects from arbitrary viewpoints. Also, the manipulator and F/T sensor can be used to measure other features such as weight, depth of occluded holes in the object, etc.

It is important to realize that with these additional sensory inputs, we can discriminate between object identities, aspects and poses, that would otherwise appear indistinguishable to just a fixed viewpoint vision-based system. Our system is capable of dynamic viewpoint selection if that's what is needed for optimum disambiguation between the currently held hypotheses.

We attack the problem of viewpoint and sensor-type selection as follows. After observing the object from an initial viewpoint with, say, a vision sensor, a set of hypotheses is formulated about the object identity and pose. We then search for a viewpoint that will enable the system to observe features which will best discriminate between the competing hypotheses. This is possible because, for any active hypothesis, we can predict the feature set which would be observed from a candidate viewpoint with a candidate sensor if that hypothesis was correct. By doing this for each active hypothesis, we can determine the amount of ambiguity that would be resolved using that viewpoint-sensor combination. This is the crux of our approach.

In the remainder of the paper, we will present our technique in some detail. In the next section, we will describe how object hypotheses are developed, and present a measure of ambiguity in a set of object hypotheses. Section 3 describes the method we use to predict the features that can be observed from a particular viewpoint. In Section 4, we describe the types of features that our system uses, and give a brief overview of how these features are derived from sensory data. In Section 5, we describe the algorithm that we have implemented to search through the space of viewpoints. The algorithm makes use of the object's aspect graphs, the search space being comprised of the set of viewpoints corresponding to the nodes in the graph. In this section, we also discuss future work which will incorporate uncertainty into the algorithm. The paper concludes with a brief discussion of our experimental work.

2. Approach

In order to illustrate our problem and the approach that we will take, we begin this section with a two dimensional example. The problem in the example consists of making sensory measurements for distinguishing between the two 2D objects shown in Fig. 1. We will assume that a sensory measurement yields the edges that are visible from a particular viewpoint, the length and the orientation of the measured edges being subject to experimental error. For the purpose of facilitating the explanation here, we have used the integers, 1, 2, 3 and 4, to denote the sides of one object, and the letters, a, b, c, d, e and f, to denote the sides of the other.

Suppose that the first sensor reading that we obtain is the straight line segment (denoted by the label S_1) observed from the viewpoint V, as illustrated in Fig. 2a. This line segment could correspond to a number of possible edges. Note that since there is some uncertainty in the edge extraction process, the length of the sensed edge could not be assumed to be absolutely accurate. Therefore, it is possible to map a measured edge to any model edge whose length is within some tolerance of the length of the measured edge. The possible assignments of model edges to the measured edge are illustrated in Fig. 2b. Note that if the system was constrained to

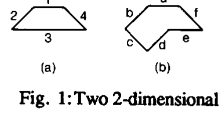


Fig. 1: Two 2-dimensional objects with edge labels.

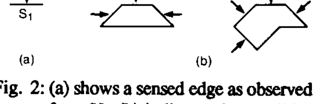
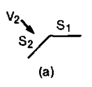


Fig. 2: (a) shows a sensed edge as observed from V. (b) indicates the possible matching model edges.

*This work was supported by the Engineering Research Center for Intelligent Manufacturing at Purdue University, and the industry supported Purdue CIDMAC program.



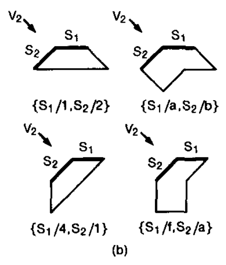


Fig. 3: (a) shows the two sensed edges, as observed from V_2 . (b) shows the four possible object hypotheses for this pair of sensed edges.

view only one aspect of the object, this is as far as the recognition process would proceed, *and the identification would be ambiguous.*

Now we are faced with the question of where to apply the next sensing operation. Since the single edge that we have found has many possible interpretations, and since no relational information can be obtained from a single 2D edge, we arbitrarily choose to apply the second sensing operation at one end of the first edge. From this new viewpoint, V_2, we observe S_1 and S_2, as shown in Fig. 3a. Now, using the two model objects, and matching the sensed edges to the possible model edges, and using the relational constraints in the model to prune away impossible hypotheses (e.g. as in [5]), we obtain the four possible matches shown in Fig. 3b. So, at this time, we have four possible hypotheses for the measured edges, S_1 and S_2. To explain the notation used in Fig. 3b, the first of these hypotheses is denoted by $\{S_1/1, S_2/2\}$, meaning that S_1 is hypothesized to be the model edge 1, and S_2 the model edge 2; both model edges belonging to the left object in Fig. 1a.

We now have enough information to attempt to make some intelligent choices about where to apply the next sensing operation. Since a sensing operation is only allowed to consist of viewing the object from a chosen viewpoint, our problem is to choose an appropriate viewpoint to disambiguate between the four hypotheses shown in Fig. 3b. Fig. 4a-4c show possible viewpoints, and the corresponding edges that would be observed for each of the four hypotheses. If we choose the viewpoint in Fig. 4a, it is clear from the figure that we will never be able to distinguish between the third and the fourth hypotheses because the new visible edge from this viewpoint makes the same angle with the previously measured edge S_2. However, if we choose the viewpoint as shown in Fig. 4c, the visible edges will uniquely identify the object and its orientation, regardless of which hypothesis is actually correct. Thus, the best viewpoint is the one shown in Fig. 4c.

This approach can be extended to three dimensions. Consider Fig. 5, which shows a piston and a crankshaft. If range data processing finds a

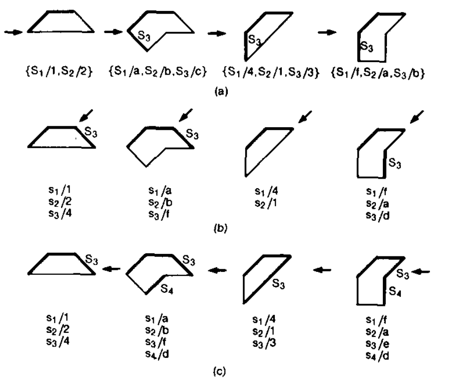


Fig. 4: In (a), the edges observed from the candidate viewpoint appear similar for the last two hypotheses. In (b), the observed edges appear similar for the first two hypotheses. In (c), the observed edges have a unique appearance for each of the four hypotheses.

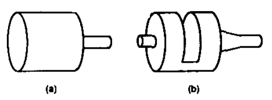


Fig. 5: shows a piston, (a), and a crank shaft (b).

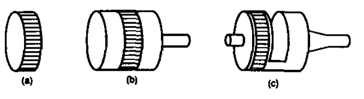


Fig. 6: If a cylindrical surface is measured (a), the possible object hypotheses are as in (b) and (c).

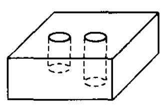


Fig. 7: A block with two holes of different depths.

cylindrical surface, as shown in Fig. 6a, then the possible object hypotheses are as shown in Fig. 6b. If the next sensing operation is properly selected, we can guarantee that the object will be uniquely identified. It is also possible to choose the next sensing operation in such a way that it will not remove any ambiguity. Fig. 7 shows another three dimensional example. Here the two holes in the block are of differing depths. Thus, the only measurement that will uniquely determine the orientation of the block is the measurement of the depth of the holes. Even if the system finds all of the planar surfaces, angles between adjacent surfaces, convex edges and edge junctions that there are to find, it will still not be able to determine the orientation of the block until it measures the depth of at least one of the holes.

While these examples serve to illustrate the problem, they don't really show any clear approach to its solution. In the remainder of the paper, we will develop a technique for determining optimal sensing strategies, given a set of sensory data which has been obtained. Our method hinges on the ability to predict the set of features which might be observed from different viewpoints given a set of competing hypotheses about the object's identity and position. We must also be able to use these predictions to determine how much ambiguity can be eliminated from a set of hypotheses by observing the object from a candidate viewpoint. For this purpose, we will introduce a formalism which can be used to express the ambiguity in a set of feature hypotheses. Our formalism will also allow us to quantify the possible reduction in this ambiguity by future sensing operations.

Before introducing our measure of ambiguity, we first explain how hypothesis sets are generated in our system. The method we use is to form an initial set of hypotheses, and then refine this set using successive sensor readings. The algorithm we use to accomplish this is as follows. We are given a set of sensed features, $S = \{S_1, S_2 \cdots S_k\}$, a set of model features, $M = \{M_1, M_2 \cdots M_n\}$, and a function, $f : S \rightarrow 2^M$, which maps each sensed feature onto possible model features; 2^M represents the power set of M. [To illustrate the function f, for the example discussed via Figs. 1 through 4, $f(S_1) = \{1,2,4,a,b,c,f\}$.] We use the following algorithm to refine a hypothesis set, given a new sensor reading, S_i.

```
refine-hyp-set (hyp-set, $S_i$)
    new-hyp-set <- nil
    for h $\in$ hyp-set
        for m $\in$ f( $S_i$)
            if consistent(h,{$S_i$/m}) then
                new-hyp-set $\leftarrow$ append(new-hyp-set, h $\cup$ {$S_i$/m})
    return (new-hyp-set)
```

A few things need to be said about hypothesis consistency at this point. First, we form the initial hypothesis set by calling refine-hyp-set with the null set as the first argument. By definition, all possible matches of sensed features to model features are consistent with the null set. Thus, the initial hypothesis set is trivially defined by

$$H_1 = \{ \{ S_1/m \} \mid m \in f(S_1) \}$$

Second, a match, $\{S_i/m\}$, is consistent with a hypothesis h if for each $S_j/p_k \in h$, if a relation, R, holds between S_j and S_i in the sensed data, then that relation also holds between p_k and m in the object model. Relationships used by our system include surface adjacency, dot product of the normals of two surfaces, and dot product of the vector connecting the centroids of two surfaces with the surface normal of one of the surfaces. It should be noted that the values of the dot products are never exact, so we have established tolerances on the allowable differences. It should also be noted that some types of surfaces are prone to error in these measurements, and are therefore exempted from these tests, (e.g. the measurement of the average surface normal of a cylindrical surface).

For computational purposes, the structure of each hypothesis is more complex than what is exemplified by H_1 above. Each hypothesis is given a frame like representation in which the slots correspond to items such a pose transformation, certainty value, and the name of the object corresponding to the hypothesis, etc. By pose transformation, we mean a homogeneous transformation matrix T_{obj} that maps the model object into the hypothesized object. All the model objects are generated by a solid modeler, which determines their initial placement in the 3-space representing the robot work area. Note that, in general, it may not be possible to generate the pose transformation, T_{obj}, after the first sensory measurement (e.g., in Fig. 2, the information available is not sufficient for the computation of the pose transformation after only the measurement of S_1). However, usually after two or three measurements have been taken, the pose transformation can be established for each active hypothesis.

In addition to the relational constraints between the matches, if a pose transformation has already been established for an active hypothesis, then the location of a feature (as measured, for example, by its center of mass) must be within some tolerance of the location predicted for the matching model feature using the pose transformation. The predicted location of a model feature in each hypothesis is computed by performing the transformation $T_{obj} * C$, where C is the location of the model feature in the 3-space used by the solid modeler.

In our previous example, given $S = \{S_1, S_2\}$, our algorithm would find:

$$H_1 = \text{refine-hyp-set}(\{\}, S_1)$$
$$= \{\{S_1/1\}, \{S_1/2\}, \{S_1/4\}, \{S_1/a\}, \{S_1/b\}, \{S_1/c\}, \{S_1/f\}\}$$
$$H_2 = \text{refine-hyp-set}(H_1, S_2)$$
$$= \{\{S_1/1, S_2/2\}, \{S_1/4, S_2/1\}, \{S_1/a, S_2/b\}, \{S_1/f, S_2/a\}\}$$

In terms of complexity, for each i, the time complexity is $O(|H_{i-1}|\ |f(S_i)|\ i)$, where $|.|$ denotes the cardinality. The pairwise consistency check of S_i/m with each element of h gives rise to the i factor. Note that $|H_i| \le |H_{i-1}|\ |f(S_i)|$, which provides a worst case upper bound on the time complexity. That is, if we cannot prune the number of hypotheses using relational constraints, $|H_i|$ grows exponentially with i. Fortunately, it is generally the case that enforcing relational constraints substantially reduces the number of hypotheses from this upper bound.

At this point we can define a measure of ambiguity in a set of hypotheses to be the number of hypotheses contained in that set, $A_i = |H_i|$. This definition can be extended to define reduction in ambiguity by the measurement of additional sensed features by $\Delta A_i = A_i - A_{i-1}$. Building on this, given a set of hypotheses, H_i, obtained from a set of sensor data which has been taken, it is possible to anticipate the maximum ambiguity in the next set of hypotheses, H_{i+1}, if we know the type of sensing operation which will be performed and where it will be applied.

Consider the four hypotheses in Fig. 4a. If the first hypothesis, corresponding to the leftmost rendition in the figure, is correct, no new edges will be viewed. Thus, if no new edges appear in the sensed data, the hypothesis set will be reduced to the element, namely, $\{S_1/1, S_2/2\}$, which in words means that both both the sensed edges S_1 and S_2 belong to object 1 and correspond to the edges 1 and 2. On the other hand, if the second hypothesis is correct, one new edge will appear, and it will be oriented at approximately 45 degrees from the vertical. Therefore, if such an edge is observed in the sensed data, the new hypothesis set, H_3, will again be reduced to a single element, namely, $\{S_1/a, S_2/b, S_3/c\}$. If either of the last two hypotheses is correct, S_3 will correspond to an edge aligned with the vertical axis. Therefore, if either of these is correct, S_1, S_2 and S_3 will have the same appearance, making these two hypotheses indistinguishable based on the three measurements, S_1, S_2 and S_3. Because of this, if an edge aligned with the vertical axis is measured, H_3 will contain two hypotheses, corresponding to the two rightmost renditions in Fig. 4a. Thus, the maximum possible ambiguity corresponding to this viewpoint is two.

In contrast, in Fig. 4c, the maximum possible ambiguity is 1, due to the fact that for each hypothesis, a unique set of sensed edges will be observed from the proposed viewpoint. We will use the symbol A^i_{max} to denote the maximum ambiguity in ith refinement of a hypothesis set for a proposed sensing operation. Given this formalism, the task of our system is to find the sensing operation which will minimize A^i_{max}.

Automatically determining what type of sensing to use, and where to apply it so that A^i_{max} is minimized, is a complex task. Many considerations must be taken into account, including the cost of the sensing operation, how to go about searching the model space for the best location at which to apply the sensing operation, and how to take into account possible uncertainties in the sensory data when predicting the possible hypotheses at the next sensing application. Ultimately though, the choice will be only as good as our ability to predict the possible outcomes of the sensing operations which we can choose to apply. This will be the topic of the next section.

3. Predicting Sensor Readings

In order to determine which sensing strategy will minimize A^i_{max}, we must be able to predict the possible results of candidate sensing operations. Consider Fig. 4 again. The ability to determine which viewpoint to try depended on the ability to predict the set of possible sensor readings that would be obtained from the various viewpoints. In the example of Fig. 4, this amounted to being able to predict the geometry of the line segments that would be observed from various viewpoints relative to the edges in the four object hypotheses. In the general case, predicting sensor readings depends on the ability to determine the features which will be observed by a particular sensor from a particular viewpoint if the object under examination is the hypothesized object in the hypothesized position. In general, there will be several active hypotheses, so we must be able to enumerate the features which would be observed for each of these hypotheses, using the candidate sensing strategy.

In order to have this predictive power, the system must be able to formulate position hypotheses which correspond to the feature hypotheses. This can be done once we have completely determined the position (location in 3-space and orientation) of any of the hypothesized features. If we have only observed one feature, this is usually not possible. For example, if we observe a planar surface, we are able to determine the normal to the surface, but not necessarily the rotation about the normal. Furthermore, edge locations tend to be noisy, so it is also unlikely that we will be able to obtain an accurate position of the centroid of the surface. If we have observed two features, our chances are much better. For example, once we have found two adjacent planar faces, we are able to measure two sets of surface normals, which fixes the object's orientation.

We can establish a position hypothesis once we have established a correspondence between two sensed surfaces and two model surfaces, provided the two are planar. We do this as follows. Let Sn_1 and Sn_2 be the surface normals for the two sensed surfaces, and Mn_1 and Mn_2 the surface normals for the corresponding surfaces in the object model. Also, let Cs be the centroid of either of the sensed surfaces and Cm the centroid of the corresponding model surface. We can now define two coordinate frames in terms of $<Sn_1, Sn_2, Cs>$ and $<Mn_1, Mn_2$ and $Cm>$. Our method is to find the transformations, T_s and T_m, which map each of these frames into a standard frame, and then compose these to find T_{obj}. The standard frame is with Cs and Cm at the origin, Mn_1 and Sn_1 lying along the negative Z axis and Mn_2 and Sn_2 lying in the positive half of the Y-Z plane. We find T_s as follows:

$$R_z = \frac{-Sn_1}{|Sn_1|} \qquad R_x = \frac{Sn_1 \times Sn_2}{|Sn_1 \times Sn_2|} \qquad R_y = R_z \times R_x$$

$$R = \begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix} \qquad T_s = \begin{bmatrix} R_x & 0 \\ R_y & 0 \\ R_z & 0 \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -Cs_x \\ 0 & 1 & 0 & -Cs_y \\ 0 & 0 & 1 & -Cs_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We compute T_m in a similar fashion. A derivation for these equations can be found in [4]. Finally, we compute T_{obj} using the equation:

$$T_{obj} = T_s^{-1}\, T_m$$

While it is occasionally useful to be able to predict the location of a hypothesized feature, more generally we wish to solve the inverse problem, i.e. given a location and an object hypothesis, which feature should we expect to find there? The solution to this inverse problem is not as simple as just performing a transformation. Instead, we must rely on a geometric modeling system to aid us in performing this task. In particular, we want to give the modeling system a viewpoint, and ask it what features we should expect to observe (for a particular object in a particular position) using a particular sensor. Obviously this is an oversimplification, since the geometric modeling system has no notion of sensing, and the features we should expect to observe depend on the type of sensing we plan to use. For example, 2D vision will not be able to distinguish between curved and planar surfaces, while 3D vision will not be able to find small holes in the object. Therefore, we must augment the geometric model, so that we can determine the set of features which will be observed by the different sensors.

The augmented geometric model contains a great deal of information in tabular form. This information includes the surface areas, the surface normals, adjacent surfaces, surface types, and location of centroids for each of the model surfaces. There is also a set of information concerning the actual geometric model (e.g. name of file containing a CSG description of the object). The augmented model also includes a table of model features which can be observed by each type of sensor. These tables are used so that the system will not have to invoke the full power of the geometric modeling system unless absolutely necessary.

The augmented model also contains an aspect graph for the object. The aspect graph specifies which model features can be observed from each tessel on a viewing sphere centered at the world origin. Using the aspect graph, determining a set of visible features is relatively straight forward. First, T_{obj} is used to determine the tessel on the sphere from which the object is being observed. The center of the viewing tessel is computed by normalizing the vector, $T_{obj}^{-1}V$, where V is the viewpoint in world coordinates. The set of features visible from this tessel is then intersected with the set of features which can be observed by the particular sensor to determine the set of features which is visible to the sensor from the specified viewpoint (providing the object hypothesis is correct).

Having described how the set of observable features is determined for a given object hypothesis and a particular sensing strategy, we now turn our attention to the features themselves. In particular, we will now describe what features are used by the system for each of the sensing modalities, how those features are derived from sensory data and how the feature information about the object models is stored.

4. Observable Features

In this section of the paper, we describe the features which can be observed by each of the sensors that our system uses. Our system currently uses a structured light scanner to obtain 3D information about the scene, overhead and a manipulator held cameras to obtain 2D information about the scene, a force/torque sensor mounted on the robot's wrist, and a manipulator which can be queried to find the distance between its fingers.

4.1. 3D Features

The richest set of features available to the system comes from range data. Range data is gathered for a set of points in the scene, using a range scanner which the robot manipulates. This initial data is converted to *x,y,z* data. Subsequent processing of the range data yields the desired attributes and relations, which are used to formulate the feature hypotheses. This section also describes how the model features and relationships are stored and accessed during hypothesis formulation.

4.1.1. Surface Segmentation

The input range data is in the form of R(u,v), that is, a value relating to the range is given over the grid defined by the sampling variables u and v. For our range scanning, these values are converted to a range map of the following form through use of a transform described in [2].

$$\mathbf{X}(u,v) = (x(u,v), y(u,v), z(u,v))$$

Using algorithms described in [15,16] surface normal vectors as well as mean and Gaussian curvature values are determined for each point with adequate neighbors for the window convolutions involved. This provides three more local property maps:

$$\mathbf{N}(u,v) = (N_x(u,v), N_y(u,v), N_z(u,v))$$

$$M_{curvt}(u,v)$$

$$G_{curvt}(u,v)$$

These local surface features are used in the segmentation and surface characterization steps. The label map Label(u,v) is initialized at this time, where each point is given a label of *valid* if valid range data existed for that point, or *invalid* if not.

For each point labeled as *valid*, the local surface features are examined to detect edge points. We have explored three methods of edge detection. While no single one of these finds adequate edge points to fully segment all surfaces in most scenes, using all three provides accurate surface segmentation at the expense of additional computation and occasional edge artifacts.

A label of *jump edge* is placed in Label(u,v) if the 3-D distance from the corresponding range point X(u,v) to one of its neighbors is greater than some threshold. A greater degree of sophistication can be added by also requiring that the range discontinuity to the neighbor on the opposite side of the point in question be significantly less or of opposite sign. The allows the later segmentation of a surface that is nearly parallel to the light stripe without labeling every point on its surface as an edge, just those at the front and rear boundaries.

The next most obvious type of boundary between surfaces is formed by points where the surface curvature is at some maximum value. A threshold corresponding to some minimum allowable radius of curvature is set. If the magnitude of the mean or Gaussian curvature exceeds the respective threshold, the point is given a label of *curvature edge*.

Finally, all points with valid surface normal values are examined. If a point in the neighborhood has a surface normal which differs sufficiently from the local surface normal, the points between these two are given the label of *surface normal disparity edge*. Typically the neighborhood is considered to be those points within two pixel locations of the point under consideration. This allows accurate labeling of edges without labeling the entirety of a small cylindrical surface as an edge.

At the completion of this step, the label map contains the following labels: *invalid, valid edge*, and *valid non-edge*. (Where *valid edge* is the union of *jump edge, curvature edge*, and *surface normal disparity edge*) The *valid non-edge* points are then grouped into contiguous regions in Label(u,v). Each region is given a surface label, and the points in the region are given labels indicating the surface they belong to.

4.1.2. Determination of Surface Attributes and Relations

Our purpose for analyzing range data, a fairly expensive computational undertaking, is to find surface attributes that are useful at a higher level. If a given surface exists, what is desired is not a list of points belonging to it, but rather a description of that surface in terms of its location, orientation, surface area, etc. We also want to know the geometric relationships between the surfaces. This section briefly describes the information that is derived. Details of how the attributes and relations are found are contained in [3].

Probably the most important feature of a surface is its 3-D shape. Surface types are classified as *planar, cylindrical, elliptically cylindrical, spherical*, or *unknown*. The term *cylindrical* is used in place of the more correct but unwieldy *circularly cylindrical*, but denotes cylinders whose cross section is approximately circular rather than elliptical. Also, we are dealing with generalized cylinders and spheroids, so an ellipsoid would be classified as *spherical* and a cone would be classified as *cylindrical*.

If the surface is *cylindrical* or *spherical*, the radius of curvature is found. Two radii are found for *elliptical cylindrical* surfaces. The cylindrical axis is found for *cylindrical* or *elliptically cylindrical* surfaces, and the major axis is found for *spherical* surfaces.

The following attributes are found for all surfaces:

- *Location* – The surface location is found as the average *x,y,z* location of all points belonging to the surface.
- *Orientation* – Similarly, the surface orientation is the average surface normal of the points belonging to the surface.
- *Area* – The area is found by a series of cross products, as described in [3]. A similar attribute is the area found by a similar operation carried out only on the four corners of the surface.
- *2-D Shape* – The 2-D shape is found by first finding the corners of the surface, as described in [3]. The relative locations of the corner points allow the 2-D shape to be classified as one of the following: *irregular, trapezoid, parallelogram, rectangle*, or *square*. Searching out in a number of directions from the point in the label map corresponding to the range point closest to the surface's centroid finds approximations to the average and variance of the 2-D radius of the surface. If the ratio of the average over the variance is above some threshold, the 2-D shape classification is changed to *round*. The relative distances between adjacent corners are examined to find the major and minor axes of the surface, which may be expressed in vector form.
- *Texture* – The texture of surfaces includes important information. This is captured in two attributes, the variance of location and the variance of orientation. The variance of location for each surface is the average 3-D distance between a point belonging to the surface and the average of the point's neighbors. The variance of orientation is the average angle between the individual surface normals of the points on the surface and the average normal for the surface.

Once the above attributes have been found, a number of relationships are found for pairs of surfaces. The most important relationship between two surfaces is their adjacency, if it exists. An analysis of the surface normals and center locations of a given adjacent pair of surfaces indicates the type of adjacency, either *convex* or *concave*. A description of the methods used for finding adjacency relations is given in [10] and [3].

A second relationship output is the angle between two surfaces. As the average surface normal for each surface is known, the dot product of these vectors determines the angles between any pair of surfaces.

At times a single surface may be oversegmented, resulting in two or more components. Noise in the range data gathered could be responsible for this, as could occlusion by another surface. To hypothesize that two planar surfaces could be components of the same surface, it is adequate to find a *coplanar* relationship between them. Two surfaces are coplanar if their surface normals are parallel, and the surface normal of either one and the vector joining the two centroids form a right angle. To make the same hypothesis about two cylindrical surfaces, their axes must also be parallel.

4.2. 2D Features

The features which are visible to the 2D camera are not nearly as robust as those visible to the range scanner. In particular, surface types cannot be determined from 2D data, edge detection is not as good (since only gray level edge detecting can be used), and relationships between surfaces cannot be measured (except for adjacency). The primary advantage of 2D vision is that it is computationally less expensive than 3D vision. Also, since our range scanner is held by the robot, and one robot move is required for each projected stripe, using 2D vision reduces the number of required manipulations from the large number required to scan a scene to the much smaller number required to grasp the hand held camera and position it at the appropriate viewpoint.

In addition to using the hand held 2D camera to derive 2D features, our system also uses an overhead camera to guide the initial application of the range scanner. In particular, the overhead camera is used to obtain an estimate of the positions and orientations of the objects in the work space. This initial application of the 2D camera can also measure certain global features about the objects, for example: aspect ratio, moments of inertia, and object size. In the remainder of this section we will discuss both this preprocessing, and the types of features that we use from 2D vision.

4.2.1. Preprocessing

The supervisory camera is used in the preprocessing as follows. First, an image of the work cell is digitized. This image is subtracted from a reference image of the work cell, and the result is thresholded. This binary image is then subjected to a component labeling process. Then, the center of mass and principal axis of each of the components is computed. The center of mass is used as input to an inverse perspective transformation, which gives an approximate world location of the center of mass of the object. The inverse perspective transform is performed using the two-plane method of camera calibration [1]. Each of these operations is fairly common in the field of computer vision, therefore, we will not describe them here. The interested reader can find the details in a variety of references, including [9,11,13].

There are several aspects of this approach that lead to error. First, note that we use an inverse perspective transformation to determine the world coordinates of the center of mass of the object. This operation requires three input parameters, the I and J image coordinates of the point, as well as the Z world coordinate of the point. Since the supervisory camera is incapable of determining the Z coordinate, it is estimated based on the average heights of the work pieces at their possible stable orientations. Additional error results because the binary image which is used in calculating the center of mass of an object may not correspond exactly to the top surface of a work piece. More often, it will be a composite of the top surface along with one or more of the sides, depending on the orientation of the object. Thus, the center of mass rarely corresponds to the center of the top surface of the object. In spite of these inaccuracies, this method does provide information adequate to limit the application of our 3D rage scanning to interesting areas of the work cell.

In addition to these duties, the supervisory camera is also used to discern certain global features of the work pieces. Using aspect ratios, elongated parts may be recognized. Often, holes in parts can be recognized using binary vision. Finally, the overall size of an object gives some clue to its possible identity. Each of these measurable 2D global features can be used to determine a set of initial object hypotheses. In some cases, this can reduce the amount of hypothesis formulation and verification that is required using local features.

4.2.2. Local Features

The local features (i.e. features that are confined to local areas of the object, such as a single surface or edge) that we can obtain from 2D image processing include holes in the object, surface texture and intensity edge information. In our current experiments, the object surfaces are all smooth, containing little or no surface texture information. Therefore, the primary 2D features that we use are holes and grey level edges.

Although gray level edge detection is not as robust as the 3D edge detection, it is generally much faster computationally. Furthermore, using object hypotheses to guide the application of the edge detector, the problem is reduced from edge detection to edge verification. In particular, once we have an object hypothesis which includes a position hypothesis, we can predict the set of edges visible to the 2D camera. If we know the camera transform, we can predict where these edges will be found in the image plane. The image obtained from the camera can then be used to verify the presence of the edge. This edge verification is done using the Dempster-Schafer formalism applied to a binary frame of discernment (i.e. edge-present/edge-not-present) [12].

4.3. Using the Manipulator to Measure Features

The last type of sensing that our system can perform is active sensing of the environment using the robot manipulator. The manipulator can be used in either of two ways. Its fingers can be closed on an object to measure its width, or, a guarded move toward an object surface can be executed to precisely measure the height of that surface. Using these techniques, we can precisely (to within the known error of the manipulator position) measure features on the objects in the world. Like range scanning, using this type of sensing requires the active participation of the robot, thus incurring the additional overhead of planning and executing robot motions.

The utility of measuring object widths becomes evident when we have competing object hypotheses, and the difference in sizes of visible features of the two objects is less than what can be perceived by the 3D or 2D vision systems. Of course 2D vision is very imprecise, as discussed above, due to the use of the inverse perspective transform using an estimate for the world Z coordinate. Inaccuracies in the 3D vision system are caused by the interstripe distance of the structured light scanning technique. Differences smaller than the distance between stripes cannot be accurately measured. Rather than always use dense scanning, the manipulator can be used to perform the more precise measurements, only when they are required.

Measuring the height of object surfaces becomes particularly useful when those surfaces are obscured from the view of the vision systems. A good example of this is the piece with the two holes of differing depths shown in Fig. 7. When cases like this arise, the vision systems are unable to observe the distinguishing features of the object. In such cases, the manipulator is used as a probe to resolve the ambiguities. Manipulator probing can also be used to determine the existence of protrusions from object surfaces, especially when these protrusions are obscured from the view of the vision sensors (e.g. when the work piece is positioned such that it occludes the surface which has the protrusion).

Features which can be detected using the manipulator as a sensor are also stored in tables for rapid access. Each surface has a corresponding list of holes and protrusions. Thus, once a surface hypothesis has been made, it is a simple matter to consult a table to obtain a list of holes and protrusions which are a part of that surface. When such features are sufficient to distinguish between the hypothesized surfaces, the manipulator is used to measure them.

Determining when to use the manipulator to resolve ambiguities that are too subtle to be observed by the vision systems is more difficult, since the degree of difference required to be measurable by the vision systems is dependent on the implementation, and run time parameters of those systems. Thus, the system must examine the current set of hypotheses and determine if any of them can be eliminated if precise manipulator measurements are made. This consists of comparing the sizes of the hypothesesized surfaces and noting whether the difference in size can be measured reliably by the vision system, and if it cannot, whether or not the precision of a manipulator measurement is sufficient to distinguish between the two. An alternative to this approach is to fix a priori the accuracy of the vision systems, and then construct a table of pairs of features which can be distinguished using manipulator measurements. We have opted for the latter approach.

5. Choosing the Best Sensing Strategy

In this section of the paper, we will describe the algorithm that our system uses to choose a sensing strategy. In essence, this is simply a search problem. The search space consists of the possible sensing operations from the possible viewpoints. Goal states are recognized using A^{i}_{max}. Since the space of locations can be very large (consider that the manipulator can be used *anywhere* in the robot's work envelope), we must devise some heuristic to guide the search through the space of possible sensor applications. In order to accomplish an efficient search of this space, we use the concept of the aspect graph. An aspect graph characterizes the possible viewpoints from which an object can be observed by grouping viewpoints that see the same features into equivalence classes. A node in the aspect graph corresponds to the set of viewpoints from which a unique set of object features can be observed. Arcs in the graph connect nodes which contain adjacent viewpoints. Fig. 8 shows the aspect graph, and regions which view each aspect, for the object in Fig. 1b. Also, with each node in the aspect graph, we will associate a principle viewpoint. This viewpoint is chosen by using the average location of the viewpoints which view the aspect.

Aspect graphs for objects can be generated analytically or by an exhaustive examination of the object. We generate our aspect graphs exhaustively. This is done by creating a CAD model of the object, centered within a tesselated viewing sphere (we currently use 60 tessellations [14]).

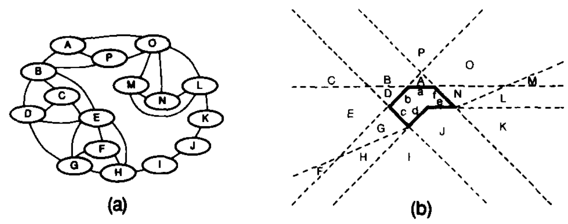


Fig. 8: (a) shows the aspect graph for the object in Fig. 1b, and (b) illustrates the regions which view the different aspects.

The geometric modeler is then used to view the object from the center point of each tessellation, and the set of visible features is recorded. Using this information, it is a simple matter to generate the aspect graph. Tessels that see the same feature set are grouped together into nodes. The arcs between nodes are generated using tessel adjacency. Finally, each aspect is assigned a principle viewpoint.

Using an aspect graph representation, when we make object hypotheses, we implicitly make hypotheses about which aspect of the object we are observing. Thus, we can redefine our search space to be the space of sensing operations applied from the principle viewpoints of the various aspects. The algorithm that we use first determines the transformation from the current viewpoint to the principle viewpoint of an aspect in the aspect graph corresponding to that hypothesis. We then apply this viewpoint transformation to each aspect hypothesis and see if we can eliminate the ambiguity in the predicted sensor reading. This is done for each type of sensing that can be used. When we find a viewpoint that eliminates the ambiguity in the predicted measurement, the algorithm terminates.

There are three basic components to the algorithm. First, there is a function that computes the predicted ambiguity for a specified view point, hypothesis set, sensor and set of predicted feature values. The algorithm used to do this prediction is shown in Fig. A1. The first step in the algo-

```
predict-ambiguity(VP,hypset,S,sensor)
    H1 ← refine-hyp-set-mult(hypset,S)
    H ← nil
    foreach h ∈ H1
        M-features ← object features matched in h
        L-features ← visible landmark features from VP
        if (L-features ⊆ M-features) then
            H ← append(h,H)
    return(length(H))
```

Fig. A1: Algorithm for predict-ambiguity.

rithm is to use refine-hyp-set-mult (a version of refine-hyp-set which allows multiple sensed values) to refine the hypothesis set using the predicted sensed values. The second step eliminates hypotheses from this set which do not contain matches for visible landmark features. Landmark features are features which are guaranteed to be found by the sensor if they are visible from the aspect. The time complexity of this algorithm is dominated by the complexity of the call to refine-hyp-set-mult, and is therefore upper bounded by $O(|\text{hyp-set}|\ |f(S)|\ i)$, if the set of predicted sensor readings contains a single element, and there are i matches in each hypothesis in hyp-set. Note that since a position hypothesis has already been established at this time, we can assume that the hypothesis set will not grow, since each sensed feature will be able to match at most one model feature, due to the location constraint discussed in section 2. Furthermore, using this same reasoning, we can assume that $|f(S)| \leq 1$. Therefore, the complexity of this algorithm is, $O(|\text{hyp-set}|\ i)$.

The function max-ambiguity uses predict-ambiguity to find the maximum possible ambiguity for a candidate sensing operation. This is done by calling predict-ambiguity with S set to the set of features visible for successive hypotheses. The maximum of these values is then returned as the maximum ambiguity. This algorithm is shown in Fig. A2. The time complexity for this algorithm is $O(|\text{hyp-set}|\ |\text{hyp-set}|\ i)$. Note that the latter terms in this expression are due to the call to predict-ambiguity.

Finally, the top level function used to determine the next sensing operation is choose-next-view, shown in Fig. A3. This function merely iterates over each possible node in the aspect graphs for each object hypothesis for each possible sensor. Thus, since a call to max-ambiguity is nested in the heart of this iteration, the overall time complexity is

```
max-ambiguity(hypset,VP,sensor)
    max ← 0
    foreach h ∈ H1
        S ← predicted sensed values for h, VP and sensor
        A ← predict-ambiguity(VP,hypset,S,sensor)
        if (A > max) then
            max ← A
    return(A)
```

Fig. A2: Algorithm for max-ambiguity.

```
choose-next-view(hypset)
    Amax ← 100
    foreach h ∈ hypset
        T ← h.transform
        Node-list ← h.aspect-graph.nodes
        foreach S ∈ sensors
            foreach node ∈ Node-list
                VP ← node.princple-view
                W-VP ← T * VP
                NAmax ← max-ambiguity(hypset,W-VP,S)
                if (valid-vp(W-VP) and NAmax < Amax) then
                    Amax ← NAmax
                    Sensor ← S
                    V ← W-VP
                if (Amax = 1) then
                    return(Amax,V,Sensor)
    return(Amax,V,Sensor)
```

Fig. A3: Algorithm for choose-next-view.

$O(|\text{hyp-set}|\ |\text{Sensors}|\ N\ |\text{hyp-set}|\ |\text{hyp-set}|\ i)$, where N is the average number of nodes in an aspect graph, Sensors is the set of sensors which can be applied, and hyp-set and i are defined as above. Thus, the overall time complexity of our algorithm is $O(|\text{hyp-set}|^3\ |\text{Sensors}|\ N\, i)$,

Finally, note the use of the predicate valid-vp. This predicate is used to insure that candidate viewpoints can actually be achieved using the robot (e.g. viewpoints which lie below the work table are eliminated from consideration).

To illustrate our algorithm, consider again the two dimensional object shown in Fig. 1b. As an example, suppose that the system's first view observes a single face. Call this observed feature S_1. Since the system can see only one face, we conclude that we must be viewing the object from one of the aspects in the set {A,C,F,M} and thus, S_1 could be any of faces a,b,c or f. The corresponding hypothesized viewpoints are shown in Fig. 9. To choose the second viewpoint, we apply the algorithm. For the first hypothesized aspect, A, we pick an adjacent node in the aspect graph, first node B. We now compute the transformation from the current hypothesized viewpoint to the principle viewpoint for aspect B. This transformation is now performed on each of the four hypothesized viewpoints, and the set of features which will be observed for each of these is determined. This is illustrated in Fig. 10. Notice that this new viewpoint will not be able to distinguish between the hypotheses A and D, since if either of these is correct, we will view a similar set of features

The second node the algorithm tries is node P, also adjacent to A. The transformation from the hypothesized viewpoint the principle viewpoint of P is computed and applied to each hypothesized aspect, as above. The new viewpoints are shown in Fig. 11. From these new viewpoints, it is possible to determine which hypothesis is correct, since each of the four viewpoints will allow observation of a uniquely identifiable set of features (given the previous set of hypotheses). Thus, the algorithm terminates, and this viewpoint is returned.

A problem with this approach is the possibility of uncertainty in the position hypotheses. In order to deal with this possibility, we must revise the algorithm slightly. Since we use a tessellated sphere as a basis for creating aspect graphs, our system is capable of dealing with uncertainties in the three degrees of freedom associated with the viewing configuration. Two degrees of freedom exist in the location of the viewpoint on the sphere, and one degree of freedom exists due to the viewing rotation. In order to deal with these uncertainties, we represent the viewpoint as an error ball projected onto the surface of the sphere, Fig. 12. Since viewing rotation is important only when we are considering transformations to other viewpoints, we represent the transformational error in terms of an error cone which is projected onto the sphere, Fig. 13. Thus, when the algorithm chooses a new viewpoint, that viewpoint can lie anywhere within the error

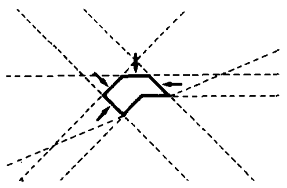


Fig. 9: Principle viewpoints of aspects which could view a single edge of the object in Fig. 1b.

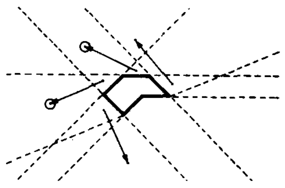


Fig. 10: A candidate viewpoint, shown applied to al four object hypotheses. Circled viewpoints "see" a similar set of edges.

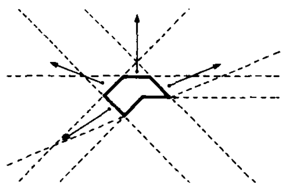


Fig. 11: A candidate viewpoint, shown applied to all four object hypotheses, from which observed edges will have a unique appearance.

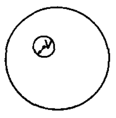


Fig. 12: An error ball projected onto the viewing sphere.

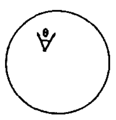


Fig. 13: An error cone projected onto the viewing sphere.

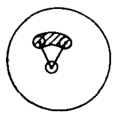


Fig. 14: Error region which defines the set of possible viewpoints when position and orientation errors are included.

region corresponding to the initial position error compounded with the transformational error, Fig. 14. Thus, when the object is viewed from a new viewpoint, it could actually be viewed from any viewpoint within the error region. Thus, the algorithm must view the object from each aspect which overlaps the error region. Note that determining these aspects amounts to determining which tessels overlap the error region, and then simply looking up these tessels in the aspect graph. These additions to the algorithm are the subject of future work.

6. Experimental Results

We verified our approach experimentally using the object shown in Fig. 15. Notice that the orientation of this object can be determined only if the location of the hole is known.

First, as discussed in Section 3, an augmented geometric model was created for the part starting from a regular CSG based solid model. This model was constructed using the PADL2 system [7], a CSG based modeler, which we have modified so that it can be interfaced with a LISP environment. The aspect graph was constructed by using PADL2 to automatically view the object from each of the 60 tessels on the viewing sphere. Tessels which viewed the same set of surfaces were grouped together into aspects,

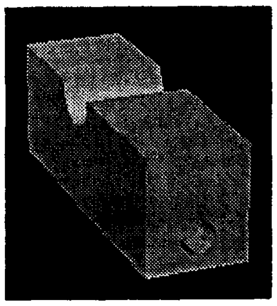


Fig. 15: Experimental object with a hole in one end, as rendered by PADL2.

```
(surface1 (surface-type . planar)
     (surface-shape . rectangle)
     (aspect-ration . 1.526054)
     (surface-area . 8.591922)
     (perimeter . 11.38)
     (centroid 29.864 7.85 -13.429)
     (surface-normal -0.046 -0.032 0.998))

(surface2 (adjacent-surfaces 3)
     (surface-type . cylindrical)
     (surface-shape . rectangle)
     (aspect-ratio . 2.065735)
     (surface-area . 2.539117)
     (perimeter . 7.142)
     (cylindrical-radius . 0.784)
     (centroid 31.778 10.887 -14.104)
     (surface-normal -0.322 -0.468 0.823)
     (apparent-axis 0.855 -0.517 0.041))

(surface3 (adjacent-surfaces 2)
     (surface-type . planar)
     (surface-shape . rectangle)
     (aspect-ratio . 1.592844)
     (surface-area . 9.1964919)
     (perimeter . 11.821)
     (centroid 33.193 13.033 -13.295)
     (surface-normal -0.046 -0.006 0.999))
```

Fig. 16: A portion of the output which is obtained from the range processing software

and aspects containing adjacent tessels were linked by arcs. Finally, a table of feature attributes for each surface on the object was created.

Once the model was created, the part was placed in the robot's work space. Range scanning was done (using the structured light scanner), and three surfaces were found. Using these surfaces, and their attributes, the system was able to develop two competing hypotheses. A portion of the output of the range processing software is shown in Fig. 16, and the two hypotheses are shown in Fig. 17. Given theses hypotheses, our algorithm chose the next sensing operation to be viewing the object with the hand held 2D camera as shown in Fig. 18. As can be seen in the figure, this sensing operation allows the end surface of the object to be viewed, and thus the presence or absence of the hole in that surface will determine the object's orientation.

7. Conclusions

In this paper we have addressed the issue of planning sensing strategies dynamically, based on an active set of hypotheses. Our algorithm uses the aspect graphs of the hypothesized objects to propose candidate sensing operations. Then, using the pose transformation and augmented object model which are associated with each hypothesis we predict the feature sets which would be observed upon application of the candidate sensing operation. Given these predictions, we are also able to predict the resulting set of hypotheses which could remain active. By repeating this process for different viewpoints and sensing operations, we are able to choose the sensing operation which minimizes the size of the largest of these sets, thereby minimizing the amount of ambiguity which can remain after the next sensing operation is applied.

```
(hyp (confidence . 1.0)
    (partname . part)
    (transform
        (0.5399 0.8411 0.0299)
        (-0.0459 -0.0059 0.9989)
        (0.8404 -0.5407 0.0354)
        (27.7738 7.1356 -17.0837))
    (matches
        (surface3 (part . 8) 1.0)
        (surface2 (part . 3) 1.0)
        (surface1 (part . l8al) 1.0)))

(hyp (confidence . 1.0)
    (partname . part)
    (transform
        (-0.5399 -0.8411 -0.0299)
        (-0.0459 -0.0059 0.9989)
        (-0.8404 0.5407 -0.0354)
        (35.6945 13.9252 -16.6782))
    (matches
        (surface3 (part . l8al) 1.0)
        (surface2 (part . 3) 1.0)
        (surface1 (part . 8) 1.0)))
```

Fig. 17: The two object hypotheses for the object as shown in Fig. 18.

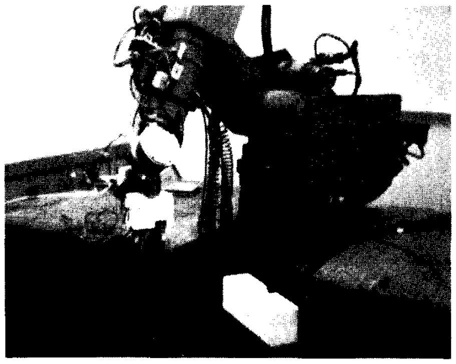


Fig. 18: The selected sensing operation is the use of the 2D hand held camera as shown.

8. ACKNOWLEDGMENTS

The work in this paper could not have been accomplished without the help of a number of people in the Robot Vision Lab at Purdue. In particular, we are indebted to Matt Carroll for his assistance with the robotic aspects of the project, and to Jeff Lewis for making the modifications to PADL2 which allowed it to be used from a LISP environment.

9. References

[1] K. L. Boyer, A. J. Vayda, and A. C. Kak, "Robotic Manipulation Experiments Using Structural Stereopsis for 3D Vision," *IEEE Expert*, Fall 1986.

[2] C. H. Chen and A. C. Kak, "Modeling and Calibration of a Structured Light Scanner for 3-D Robot Vision," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, 1987.

[3] R. L. Cromwell, and A. C. Kak, "Low and Intermediate Level Processing of Range Maps," Purdue University Technical Report TR-EE-87-41, 1987.

[4] J. D. Foley, A. Van Dam, *Fundamentals of Interactive Computer Graphics*. Addison Wesley, 1984.

[5] W. E. L. Grimson and T. Lozano-Perez, "Model-Based Recognition and Localization from Sparse Range or Tactile Data." *The International Journal of Robotics Research*, Vol. 3, No. 3, Fall 1984, pp. 3-35.

[6] C. Hansen and T. Henderson, "CAGD-Based Computer Vision," *Proceedings of the Workshop on Computer Vision*, Nov. 30 - Dec. 2, 1987, Miami, FL.

[7] E. E. Hartquist, and H. A. Marisa, *PADL-2 User's Manual*, Production Automation Project, University of Rochester, Rochester NY, 1985.

[8] K. Ikeuchi, "Generating an Interpretation Tree from a CAD Model for 3D-Object Recognition in Bin Picking Tasks," *International Journal of Computer Vision*, 1987, pp. 145-165.

[9] A. C. Kak, "Depth Perception for Robots," in *Handbook of Industrial Robotics*, ed. S. Nof, John-Wiley, NY, 1986.

[10] A. C. Kak, A. J. Vayda, R. L. Cromwell, W. Y. Kim, and C. H. Chen, "Knowledge-Based Robotics," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, 1987.

[11] A. Rosenfeld, and A. C. Kak, *Digital Picture Processing*. Academic Press, 1982, NY.

[12] R. J. Safranek, and A. C. Kak, "Evidence Accumulation Using Binary Frames of Discernment for Verification Vision," Submitted for publication.

[13] P. H. Winston, and B. K. P. Horn, *LISP* Addison-Wesley, 1981.

[14] H. S. Yang and A. C. Kak, "Determination of the Identity, Position, and Orientation of the Topmost Object in a Pile," *Proceedings of the Third Workshop on Computer Vision: Representation and Control*. Oct. 13-16 1985.

[15] H. S. Yang and A. C. Kak, "Determination of the Identity, Position, and Orientation of the Topmost Object In a Pile," *Computer Vision, Graphics, and Image Processing*, 36, 1986, pp. 229-255.

[16] H. S. Yang and A. C. Kak, "Determination of the Identity, Position, and Orientation of the Topmost Object in a Pile: Some Further Experiments," *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, 1986.