CAGD-Based Computer Vision

CHARLES HANSEN AND THOMAS C. HENDERSON, SENIOR MEMBER, IEEE

Abstract—**Three-dimensional model-based computer vision uses geometric models of objects and sensed data to recognize objects in a scene. Likewise, computer aided geometric design (CAGD) systems are used to interactively generate three-dimensional models during the design process. Despite this similarity, there has been a dichotomy between these fields. Recently, the unification of CAGD and vision systems has become the focus of research in the context of manufacturing automation.**

This paper explores the connection between CAGD and computer vision. A method for the automatic generation of recognition strategies based on the 3-D geometric properties of shape has been devised and implemented. This uses a novel technique developed for quantifying the following properties of features which compose models used in computer vision: robustness, completeness, consistency, cost, and uniqueness. By utilizing this information, the automatic synthesis of a specialized recognition scheme, called a strategy tree, is accomplished. Strategy trees describe, in a systematic and robust manner, the search process used for recognition and localization of particular objects in the given scene. They consist of selected 3-D features which satisfy system constraints and corroborating evidence subtrees which are used in the formation of hypotheses. Verification techniques, used to substantiate or refute these hypotheses, are explored. Experiments utilizing 3-D data are presented.

Index Terms—**CAD, geometric models, model-based vision, recognition strategies, recognition strategy synthesis.**

I. Introduction

Computer vision has been an active research area for over 20 years. In the past, emphasis was on low level processing such as intensity and signal processing to perform edge detection. More recently, models of objects and knowledge of the working environment have provided the basis for driving vision systems. This is known as model-based vision. The pursuit of the fully automated assembly environment has fueled interest in model-based computer vision and object manipulation. This involves building a 3-D model of the object, matching the sensed environment with the known world and determining the position and orientation of the recognized objects. The goal is to provide a solution to the problem of visual recognition in a well-known domain.

In the automation environment, recognition schemes and representations have typically been constructed using ad hoc techniques. Although objects used in the assembly process are designed with a CAD system, generally there is no direct link from the CAD system to the robotic workcell. This means the recognition systems are constructed independently of the CAD model database. What is desired is a systematic approach for both the generation of representations and recognition strategies based on the CAD models. Such a system provides an integrated automation environment. Fig. 1 shows such an integrated system. As can be seen, the system is composed of several components: a CAD system, a milling system, a recognition system, and a manipulation system. In this paper, the automatic generation of recognition strategies based on the CAGD model is studied. It has also been determined that the use of shape, inherent in CAGD models, can also be used to drive the recognition process. Others have been studying portions of this system. Recent work by Ho has focused on the generation of computer vision models directly from a CAGD model [2], [9].

The work described here investigates the use of geometric knowledge in constructing *strategy trees*. These trees provide a robust mechanism for recognition and localization of three-dimensional objects (occluded as well as nonoccluded) in typical manufacturing scenes. The run time matching of 3-D models to a scene can be expensive. If the search technique is optimized, cost can be decreased, thereby improving run time performance. One way to accomplish such optimization is by the off line examination and evaluation of the 3-D model.

A. Related Work

One of the first researchers to study the automatic synthesis of general recognition strategies was Goad [6]. He was concerned with automatic programming for 3-D model based vision. His work generated a recognition scheme for matching edges based on a general sequential matching algorithm. His algorithm proceeded in three steps: 1) predict a feature, 2) observe (match) a feature, and 3) back-project (refine the object hypothesis based on step 2). These three steps form a template which is used by the automatic programming phase. He used a unit sphere to gather *loci* of view-angles (camera positions) which represent orientations of the object. His work differs from that described here in that he obtained 3-D interpretations of 2-D intensity images rather than 3-D sensor data. The only features used were straight edges from intensity images and the search trees were generated from a template and ordered by hand rather than automatically. His system did not consider partial occlusion. However, this was a major contribution since it was one of the first attempts to automate the generation of recognition schemes.

Manuscript received August 7, 1987; revised May 15, 1989. Recommended for acceptance by J. L. Mundy. This work was supported in part by the National Science Foundation under Grant IRI-8802585 and DARPA Contract DAAK1184K0017.

The authors are with the Department of Computer Science, the University of Utah, Salt Lake City, UT 84112.

IEEE Log Number 8929950.

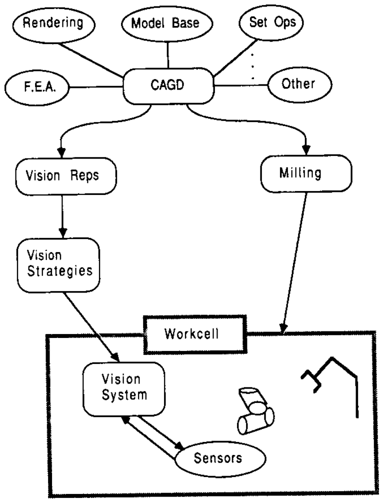

Fig. 1. Integrated automation environment.

Another influential project was the 3DPO system by Bolles and Horaud [4]. This work is the 3-D generalization of the local feature focus method [3]. Their system annotates a CAD model producing what is called the extended CAD model. From this model, feature analysis is performed to determine unique features from which to base hypotheses. The focus feature in their system is the dihedral arc. When the recognition system finds a dihedral arc, it looks for nearby features which are used to discriminate between model arcs with similar attributes. From these, an object's pose is hypothesized and subsequently verified. The work here work closely parallels the 3DPO system. However, focus features were hand chosen in 3DPO as were the local features used for discrimination.

Recently, Ikeuchi has explored the use of *interpretation trees* for representation of recognition strategies [10]. His system uses the concept of visible faces to generate generic representative views, called aspects. From this set of aspects, an interpretation tree is formed which discriminates among the different aspects. His system uses a variety of object features such as: EGI, face inertia, adjancency information, face shape, and surface characteristics. Most of these features are based on planar faces. A very specific interpretation tree is generated for an object using a set of object specific rules. The rules were selected by hand rather than generated automatically. There does not appear to be any algorithmic approach for the application of the rules to discriminate between the aspects. The branching on the tree seems to be a function of the particular aspects chosen rather than being based on the geometric information in the model.

The system developed in this paper incorporates ideas from all of the systems described above. However, the system is not dependent on a certain class of features but rather can be extended to include many classes of features not implemented at this time. The system also performs automatic selection of features based on a set of constraints: feature filters. These features are used to form a *strategy tree* which provides a scheme for hypothesis formation, corroborating evidence gathering and object verification. The flexibility of this approach makes it significantly different from related work.

Our main goal of is the automatic synthesis of recognition system specifications for CAD-based three-dimensional computer vision [8]. Given a CAD model of an object, a specific, tailor-made system to recognize and locate the object is synthesized.

To attain this goal, the following problems have been solved.

1) Geometric Knowledge Representations: The use of geometric data is central to a strong recognition paradigm. Weak methods can only be avoided when better information is available. The Alpha_1 B-spline model allows the modeling of freeform sculptured surfaces. To obtain the geometric features of interest for 3-D recognition, techniques for the transformation to a computer vision representation have been developed.

2) Automatic Feature Selection: The part to be recognized or manipulated must be examined for significant features which can be reliably detected and which constrain the object's pose as much as possible. Moreover, such a set of features must *cover* the object from any possible viewing angle. In solving the feature selection problem, a technique is available for synthesizing recognition systems. This produces much more efficient, robust, reliable, and comprehensible systems.

3) Strategy Tree Synthesis: Once a robust, complete, and consistent set of features has been selected, a search strategy is automatically generated. Such a strategy takes into account the strongest features and how their presence in a scene constrains the remaining search. The features and the corresponding detection algorithms are welded, as optimally as possible, into a search process for object identification and pose determination. The automatic synthesis of search strategies is a great step forward toward the goal of automated manufacturing. Generation of strategies is constrained, not only by the feature selection process but, by the actual task to be accomplished. Thus, strategies for a specific task might not be as strong when applied to a different task; strategies are task specific.

The remainder of this paper explains how these three components can be exploited to automate the process of selecting proper features and recognition schemes for specific goals. Algorithms are described which were developed for feature selection and which give supporting evidence for their formulation. Last, strategy trees are defined, their use in specific domains is explained, and a technique for the automatic generation of these search trees is given.

II. Geometric Knowledge Representation

Computer vision utilizes object models in a different manner than computer graphics or CAGD. In CAGD, the models must contain information about the 3-D object for rendering, performing finite element analysis, milling, and other processes. Computer vision is concerned with recognition of the objects from sensory data. CAGD models must contain information for the local design operations such as what shape to extrude or what is the profile curve for a sweep operation. Features used in construction of models are implicitly rather than explicitly used in the CAGD representation. For example, a dihedral edge formed from two adjoining surfaces is not modeled as an edge per se but as two surfaces with adjacency information.

With computer vision models, the ability to index into an object model for the purpose of recognition is needed. For example, if a 30 degree dihedral edge of length 4 inches is detected in a scene, it is necessary to determine which 30 degree dihedral it matches in the model. One approach is to index into the model and extract all 30 degree dihedral edges with similar attributes (length, adjacent faces, etc.). Some way to represent this information is required. We propose to use intrinsic features as the interface between CAD and vision. Recent research by Ho has examined the generation of several classes of computer vision models directly from a CAGD system [9].

In the experimental system developed here, a modified winged-edge model [1] is used as the interface between CAD and vision, where relationships between features are explicit in the model. It is extended for inclusion of non-planar surfaces. In addition to special mechanisms for matching, access to the geometric knowledge of the object is required for the automatic generation of strategy trees. From this modified winged-edge description, an index on feature attributes can be generated which can quickly and efficiently access the geometric knowledge contained in the model. The edge and surface information used in the aspect computation, provides additional geometric knowledge. In this case, it is necessary to know which edges or surfaces are self-occluded by the object from a particular viewpoint. When not fully visible, the knowledge of the extent of occlusion can be used in determining the potential of the feature for use in the matching process.

III. Automatic Feature Selection

Several kinds of knowledge are required for feature selection. Geometric knowledge permits the selection of a complete and consistent set of features, while the sensor knowledge provides information on the robustness and reliability with which such features can be extracted. On the other hand, domain specific information about the task can be used to select feature extraction algorithms based on their complexity, robustness, etc.

Object recognition techniques are based for the most part on geometric features of the objects to be recognized. This includes corners, edges and planar faces for polyhedra, as well as points, arcs of distinct curvature and regions of constant curvature for sculptured surfaces. Other features such as axes of inertia, profile curves, surface texture properties, reflectance, etc., can also be used. Another area of current research in CAD systems is the possibility of designing by feature, which could include process knowledge. Such capabilities would facilitate the feature selection process for object recognition.

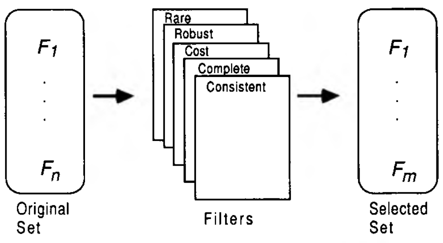

Fig. 2. The feature selection process.

The feature selection process can be viewed as a set of *filters* applied to the complete original set of features of an object (see Fig. 2). Filters select and rank features; order of application is important. Conceptually, the filters remove features from the input, in order of application, which do not meet the filter's criteria. The goal here is to automate and optimize this filtering process. The filters select features based on the following qualities:

- *rare*—histogram the features; rare features are useful for quickly identifying the object; these features make good root nodes in a search tree;
- *robust*—measure of how well the features can be detected; error and reliability;
- *cost*—measure of complexity (space and time) for computing feature;
- *complete*—does set of features cover all possible views of the object;
- *consistency*—how completely does feature characterize object pose; (i.e., how many DOF's are unresolved); how well does the feature differentiate between objects; measure the likelihood of correctly identifying the object.

A. Rare Features

The first *filter* in the feature selection phase is used to determine the uniqueness or commonality of features. This can be tuned to filter out either common features or unique features. Model features are histogrammed according to occurrences. This occurrence histogram can be used to select those features which rarely or often occur depending on the system needs.

B. Robust Features

There are two types of feature robustness a system can quantify: the robustness of a feature itself and the robustness of the extraction techniques which are applied to obtain the feature. Furthermore, features should be dependable with respect to artifacts in the scene. For example, concave dihedral edges can occur whenever a polyhedron is placed upon another polyhedron; moreover, this is likely to occur due to occlusion in a polyhedral scene. On the other hand, the likelihood of a convex edge being formed as an artifact of occlusion is very low. The knowledge of such robustness, or lack thereof, can be incorporated into the robust feature filter.

C. Complete Features

Three-dimensional models define the entire object, yet, during scheme analysis only a single view is available, or possibly multiple views, but not a complete view. How then, can the model be matched with the sensed data from the scene? Unless special fixturing is used in the manufacturing environment, we must assume that the pose of the object in the scene is unknown. One solution is the use of aspect graphs. An aspect graph is a representation of an object's topology; thus it captures all viewpoints of an object [13]. The *aspect* is the topological appearance of the object from a particular viewpoint. Slight changes in the viewpoint change the size of features, edges and faces, but do not cause them to appear or disappear. When a slight change in viewpoint causes a feature to appear or disappear, an *event* takes place. An aspect graph, or visual potential graph, is formed by representing *aspects* as nodes and *events* between aspects as paths between corresponding nodes. Several researchers have developed algorithms for the construction of aspect graphs, however, the size of the graphs poses computation limitations to their use [11], [14].

We use a discrete approximation by placing a tessellated sphere around the model, where each of the polygons represents a different viewpoint. The tessellation can be made arbitrarily fine, thus obtaining any desired granularity. Since the distance of the sensor from the work space is known *a priori*, and the sensor's physical characteristics (focal length, sensing field size, etc.) are also known, it is possible to position the sphere to correspond to the sensor's position.

An icosahedral tessellation of a unit sphere is used and then the tessellated sphere is uniformly scaled to the proper size. In experiments, it has been found that a tessellation of 80 fully covers the set of aspects. If the tessellation is subdivided to 320 cells, the same apparent aspects are obtained, but they are spread across many more cells. Each tessellation cell, called a tessel, can be thought of as a feature accumulator. That is, all object features which are visible from a tessel (i.e., that viewpoint and distance from the model) are recorded. Tessels which contain the same features are merged into the same aspect. When no more tessels can be merged, the minimal aspect set for the model/sensor pair is reached. Each aspect corresponds to a topologically different viewpoint; since all possible viewpoints are considered, complete coverage of the model is achieved.

This is similar to what Ikeuchi does in the generation of viewpoints for his interpretation trees [10]. However, the technique described here differs from his in that he uses a CAD system to generate 60 views and then, by hand, combines views with similar aspects where the only features considered are faces.

Our method can be further refined by including knowledge of the sensing characteristics determined in the robust feature phase of the process. If it is determined that a feature cannot be reliably detected when the sensing angle reaches a certain position, this knowledge can be used to eliminate features from tessels.

D. Cost of Features

The expense of feature computation can be divided into two classifications: time and space. However, time is usually the more critical element. Thus, in the experiments the cost in time of feature computations is of greatest concern. The amount of time for feature calculation is determined by both the algorithms which are available and the hardware at hand. Certain feature computations can occur at the hardware level making those features more attractive (faster) to obtain. In addition to the possibility of specialized hardware, there is a tradeoff between speed and reliability of feature detection algorithms. Such knowledge needs to be utilized in this filter.

E. Consistent Sets of Features

Although features may fulfill the requirements of the above filters for a specific workcell and task configuration, they may not discriminate between views of the object or between different objects. A feature set is considered consistent if it possesses the necessary geometric information to distinguish between aspects. Symmetric objects pose problems for this type filter since multiple aspects appear similar to the system. The consistency filter forces the set of features to be strong enough to form a hypothesis.

The geometric information contained in features differs with feature type. It is desirable to use features which make available the maximal amount of pose information possible. One way to measure geometric content is in terms of degrees of freedom, DOF, which remain unknown after a feature is matched to the model.

F. Use of the Filters

When used in combination, these filters provide the mechanism with which to build a strategy tree. The task requirements may be such that the result of these filters is the null set of features. This can be dependent on the order in which the filters are applied to the complete feature set. For example, if the filter for rare features determines that a 1/4 inch dihedral edge is the *best* feature and is applied prior to the robustness filter, that dihedral might not be accepted by the robustness filter since it is so small. Thus, the set of features would be null after the application of the robustness filter. Whereas, if the robustness filter is applied first, it wouldn't accept such features and when the rare filter is applied to the features accepted by the robustness filter, it would determine a different set of features as being *best*. The order of application is to be determined by knowledge of both the task to be accomplished and experience; this aspect of our approach merits further study.

Since there is this possibility of null feature sets when filters are applied such that they absolutely eliminate features, the filters need to be applied in a relative manner.

That is, the filters should rank the features rather than just eliminate those which don't meet the criteria. If the features are ranked by the filters, null sets should never occur. However, the order of application is still important.

IV. Strategy Tree Synthesis

Strategy trees describe the search strategy used to recognize and determine the pose of objects in a scene. This is a generalization of a hierarchical classifier or decision tree. The use of strategy trees permits one to exploit knowledge of relations between the geometric features in the models. Such trees also define a sequence of measurements or evaluations of the scene data so as to eliminate certain classifications at particular nodes.

Fig. 3 is an overview of how strategy trees are used in the system. The system consists of two parts: the off-line model analysis and strategy generation and the run time environment. The CAD model is analyzed in terms of the geometric knowledge needed for object recognition. This geometric information, which is analyzed by the feature selection process, is used by the strategy tree builder to produce the core of the run time recognition system. During run time, the strategy tree provides the search structure and control for the hypothesis generator. By using the information provided from the feature extractors and the strategy trees, the hypothesis generator attempts to hypothesize pose descriptions for recognized objects in the scene. These hypotheses are verified for correctness and a description of recognized objects and their poses are the end result.

Another benefit of the tree structure is the inherent parallelism of trees. This occurs whenever there is a branch; thus, trees with greater breadth will, in general, have higher inherent parallelism. The sequentiality of trees refers to the depth of paths in the tree. Strategy trees are shallow trees with many branches in the first two levels. Thus, there is a great deal of inherent parallelism in these trees.

The matching strategy consists of two phases: the hypothesis generation phase and the hypothesis verification phase. This recognition technique is known as hypothesize and verify. The hypothesis generation phase is controlled by the strategy tree and the verification phase substantiates or refutes the hypotheses generated from the strategy tree. As will become apparent in the next subsection, the confidence of a hypothesis can be increased at the hypothesis generation phase which has two effects: increased cost of hypothesis generation and decreased cost of the verification phase. Conversely, the confidence in an initial hypothesis can be decreased, thereby expediting the hypothesis generation phase, which increases the computational expense of the verification phase.

A. Description of Strategy Trees

A strategy tree consists of three major parts:

1) *The Root*—Which represents the object to be recognized.

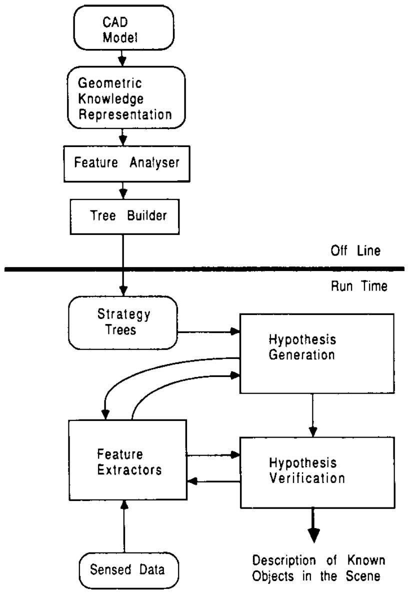

Fig. 3. Overview of strategy trees.

2) *Level 1 Features*—Which are the strongest set of view independent features chosen for their ability to permit rapid identification of the object and its pose.

3) *Corroborating Evidence Subtrees, CES*—Whose purpose is twofold: they direct the search for corroborating evidence that supports the hypothesis of the level 1 features and they direct the search for geometric information to completely determine the pose prior to hypothesis generation.

Strategy trees determine the procedure a recognition system follows for object recognition. There will be at least one strategy tree for each model under consideration. If a model is used in a different task or environment, there could possibly be a different strategy tree for each of those tasks. The level 1 features are selected using the *feature filters*. These conform to the requirements which constrain the task, environment, and model yet contain the strongest geometric information which leads to a solution. The corroborating evidence subtrees, CES, are constructed using geometric information derived from the CAD model.

B. Construction of Strategy Trees

A method is now needed for extracting the features of interest from the aspects. The level 1 nodes of the strategy tree are built from these features. Recall, that an aspect is a feature accumulator which forms a topologically equivalent set of features from multiple viewpoints. The aspect coverage algorithm, shown in Fig. 4, is used to form level 1 nodes by extracting the best, unique features from the aspects.

When $\boldsymbol{D}$ is not the empty set, it means there is at least one feature which is contained in all the aspects. Thus,

Algorithm

Define A_i to be the set of all features contained in the ith aspect, where $0 < i \leq$ number-of-aspects. Define the operation, $-$, to denote set difference. Define, f, to be a level 1 node containing a set of unique features, possibly a singleton set, which permits rapid identification of the object and its pose.

```
For each $A_i$
    $D = \bigcap D_{ij}$    where $D_{ij} = A_i - A_j$ $(i < j)$
            if $D \neq \emptyset$, then
                    choose $f$ from D
            if $D = \emptyset$ and no $D_{ij} = \emptyset$, then
                    select $f$ to be the union of 1 element from each $D_{ij}$
            if $D_{ij} = \emptyset$ for some $j$, then
                    $A_i \subset A_j$ so do nothing
```

Fig. 4. The aspect coverage algorithm.

that feature is used as a level 1 node. In the case where $\boldsymbol{D}$ is null but all the $\boldsymbol{D}_{ij}$'s are not empty, there is a combination of features which uniquely spans the aspects. Thus, a set of features for the level 1 node is used. In the last case, where the $\boldsymbol{D}_{ij}$ is null for some j, then $\boldsymbol{D}$ will also be null. Additionally, it is known that the aspect, $\boldsymbol{A}_i$ is completely contained in aspect $\boldsymbol{A}_j$. $\boldsymbol{A}_i$ must be a subset of $\boldsymbol{A}_j$ because the set difference is null and if the two aspects, $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$, contained the exact same elements, they would have been merged at the tessel stage. Since $\boldsymbol{A}_i$ is contained in $\boldsymbol{A}_j$, a level 1 node is not created at this point. Rather, this aspect will be covered by the level 1 node generated from aspect $\boldsymbol{A}_j$.

Once the level 1 nodes are built, it is necessary to generate the CES, corroborating evidence subtrees. The CES's simply substantiate that a hypothesis should be generated based on a feature matching a level 1 node. Sufficient evidence must be found that a correct hypothesis is being made before a hypothesis for the verification phase to validate is generated. This process serves two purposes: find spatially local supporting evidence for the level 1 feature and completely constrain the object's pose. Which features are used in this local corroboration is dependent on which class of feature(s) the level 1 node contains.

Occlusion becomes a factor during the determination of the CES strategy. Since dihedral edges and arcs provide the most consistent information (solve the most DOF's), they are used for level 1 nodes more often than regions or curved surfaces. Edges and arcs are composed of a starting point, an ending point, and the connecting edge or arc. When forming a strategy to handle occlusion for these features, both ends of the feature must be considered since it cannot be known *a priori* which end is occluded. Generally, four cases are considered when forming the subtrees for local feature corroboration: 1) detected feature is not occluded, 2) one end of detected feature is occluded, 3) other end of detected feature is occluded, or 4) both ends of detected feature are occluded. For some features, such as faces or regions of constant curvature, there is no concept of direction; hence, the end conditions check can be replaced with adjacency information.

There are several rules which are implemented to control the construction of the CES level. These rules are feature dependent and are expandable should other classes of features be included in the system (e.g., *generalized cylinders*).

- *Dihedral edge* rules are:
  - First look for another dihedral edge nearby which matches the model.
  - Failing this, look for an appropriate 2-D corner.
  - Failing this, use the approximate areas of adjacent faces.
- *Dihedral arc* rules are:
  - First look for another dihedral edge nearby which matches the model.
  - Failing this, look for an appropriate 2-D corner.
  - Failing this, look for the surface type of adjacent faces or other attributes of the adjacent regions (area, radius of cylinder).
- *Planar region* rules are:
  - First determine the orientation of the adjacent faces.
  - Failing this, look for a nearby dihedral edge which matches the model.
  - Failing this, look for an appropriate 2-D corner.
- *Curved surface* rule is:
  - Determine surface types of adjacent surfaces.

A CES is generated for every feature in the model which has similar attributes as the level 1 node. For example, suppose the level 1 node is a dihedral edge of included angle 30° and a dihedral edge in the scene is detected with an included angle close to 30°. A CES is generated for all 30° angles in the model. In other words, an attempt is made to determine which 30° dihedral was detected. The use of corroborating evidence focuses the search strategy by pruning unattractive paths at an early stage of the search.

C. *Usage of Strategy Trees*

The strategy tree *guides* the search through possible solutions. When a level 1 node is matched in the strategy tree and it is supported by the corroborating evidence subtrees, then a hypothesis is generated. The hypothesis is passed to an object verifier which determines whether the hypothesis is valid within some confidence level.

The combinatorial explosion of the matching process is controlled by the use of heuristics. For a detected feature to match a level 1 node, it must satisfy the following rules.

1) The attributes in the detected feature must be less than or equal to the attributes in the model (i.e., the length of a detected edge must not be longer than a model edge, area of a detected surface must not be greater than the area of the model, the included angle of a dihedral arc must be within some range of the model).

2) If the detected feature is not occluded, the attributes must be within some tolerance of the model's values.

These simple rules greatly reduce the possible matches to the level 1 features. The check "less than or equal to" for feature attributes is used due to the possibility of occlusion. In dealing with 3-D data, perspective does not alter the measurable attributes. Even with occlusion, a feature cannot appear larger (longer for edges, larger area for surfaces) than the original model.

In the above method, occlusion must be detected in the range data. Three simple cases suffice to determine whether occlusion is present or not. These tests are performed at the boundary of the detected features (i.e., dihedral edge—endpoints, surface/face—bounding edges).

1) Feature ends with a jump edge. In this case, look at the relationship between the feature and the part of the scene which forms the jump edge (scene-jump):

a) feature is nearer than scene-jump. Implies *nonoccluded.*

b) scene-jump is nearer than feature. Implies *occluded.*

2) Feature ends with a shadow edge. This is an unfortunate artifact of triangulation systems. However, this is the prevalent class of 3-D sensor in use at research labs at the present. It is unfortunate because the cause of the shadow edge is unknown. It could be the shadow is caused by the actual edge of the object (e.g., the back-top edge of a cube), or is caused by occlusion, or is caused by a nonoccluding object casting a shadow on the feature in question. Since the cause is not known, it must be considered occluded even though it may not be. Implies *occluded*

3) Feature ends with neither a shadow edge nor jump edge. It is known conclusively that the feature is *nonoccluded.*

Once a level 1 node has been matched using the heuristics described above, and a determination made as to whether the feature is occluded or not, the local CES can be evaluated, as prescribed by the strategy tree. This local evidence gathering limits the number of hypotheses generated and passed to the object verification phase by determining whether a hypothesis is justified by the local evidence. If there is not supporting local evidence, as prescribed by the strategy tree, then that level 1 match fails and the detected feature is marked as unmatched. If there is enough local supporting evidence, a hypothesis is generated for the object verification phase to accept or reject.

Two forms of verification have been examined: structural and pixel correlation. Structural verification refers to verifying spatial relations among the features which should be present in the scene. This is similar to relational graph matching in 2-D. Pixel correlation refers to the verification technique of matching predicted depth, pixel by pixel, in a generated image and the sensed *image*. This corresponds to template matching in 2-D.

Either of these methods provides for verification. This follows the hypothesis verification techniques used by others [3], [4], [12]. One of three states is assigned to the match of the hypothesized feature or pixel with the observed feature or pixel.

- *Positive Evidence:* When the observed feature or depth is approximately the same as predicted. This means the observed object matches the transformed model in the predicted image.
- *Neutral Evidence:* When the observed feature or depth is closer to the sensor than the predicted one. This seems counterintuitive but it simply means that the predicted feature/depth cannot be observed because something is possibly blocking sight of the object. In the presence of occlusion, it cannot be determined whether the difference between the prediction and the scene is due to an incorrect hypothesis or due to an occluding object. This also holds for shadow pixel/region in the range image for the same reason.
- *Negative Evidence:* When the observed feature or depth is much farther from the sensor than the predicted one. This definitely points to an incorrect hypothesis since the observed feature/depth is not occluded but is not where it should be.

If these measures are accumulated for the predicted range image or structural features, the hypothesis can be quantified and accepted or rejected accordingly. This quantification provides a measure of confidence in the hypothesis.

V. Experiments and Discussion

The concepts which have been outlined above have been implemented in an experimental system. This section describes the sensing and computational environment. The synthesis of strategy trees is demonstrated with an example polyhedron. The equipment used for the experiment consisted of a Technical Arts 100A White Scanner, DEC VAX class processors and an HP Bobcat. The images used in the experiments are part of the the Utah Range Database which was compiled for standardization of research on range images for the research community [7]. Feature computation was coded on a VAX 750 in C. The automatic generation of strategy trees and the matcher were coded on an HP Bobcat in HP Common Lisp.

Range data was obtained with the White Scanner 100A which returns actual Cartesian data. The structured light is a laser beam which is spread into a plane of light and directed onto the work space. The sensing mechanism is a GE CCD camera with a 240×240 image.

A. Geometric Design

A polyhedron, called poly_1, was designed using the Alpha_1 design system. Of course, this simple polyhedron does not exploit the freeform power of Alpha_1 but suffices as an example of how the system functions. Starting with a primitive object, a parallelpiped, planes of intersection are defined with which to remove portions of the primitive. For poly_1, two portions are removed, and this is accomplished with the set different of two planes and the primitive. Fig. 5 shows the polyhedron rendered using Alpha_1.

The construction of the hierarchical winged-edge model from the CAD model is quite simple. Form an object consisting of faces which consist of edges which consist of vertices. Fig. 6 shows the labeled edges of this winged-edge model. The edge numbering is used throughout the remainder of this chapter. Table I lists the dihedral edges for poly_1. Table II lists the faces for poly_1. These are

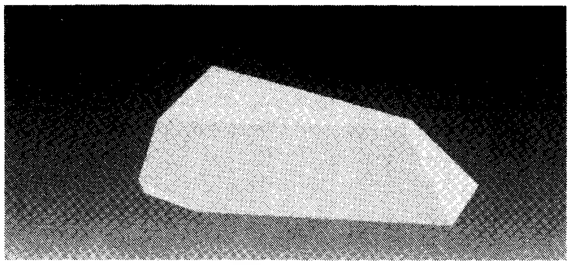
Fig. 5. Poly_1 rendered by Alpha_1.

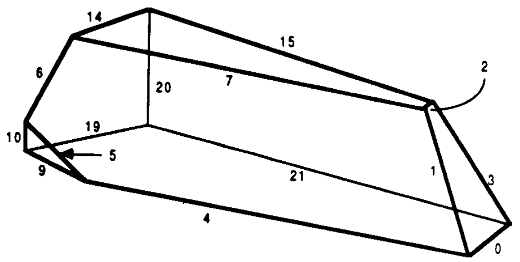

Fig. 6. Wireframe drawing for poly_1.

TABLE I
EDGE ATTRIBUTES IN POLY_1

edge	*angle*	length	*adjacent faces*
4	45.8	5.99	1 6
0	42.6	1.4	0 6
6	76.09	1.52	1 4
7	134.19	5.84	1 3
1	132.48	2.45	0 1
5	132.5	1.53	1 2
2	137.33	0.28	0 3
21	90	7.1	5 6
15	90	5.5	3 5
19	90	3.41	4 6
14	90	2.25	3 4
3	90	2.18	0 5
20	90	1.48	4 5
9	90	1.45	2 6
10	90	0.5	2 4

TABLE II
FACE ATTRIBUTES IN POLY_1

face	*area*	normal	*poly type*
0	1.8225	-0.678 0.000 0.735	convex
1	13.9725	-0.240 0.676 0.697	convex
2	0.3625	0.000 1.000 0.000	convex
3	6.9438	0.000 0.000 1.000	convex
4	4.4643	1.000 0.000 0.000	convex
5	9.2925	0.000 -1.000 0.000	convex
6	18.5328	0.000 0.000 -1.000	convex

used by the feature selection process as well as in the generation of strategy trees. Note the grouping of the edges in Table III denoted by the horizontal lines. Due to noise in the data and error in the feature extraction methods, the system cannot discriminate on angle value alone. Thus, dihedrals are grouped together if they are within 5 degrees of each other.

TABLE III
HISTOGRAM OF DIHEDRAL EDGES

angle in degrees							
0-35	*35-55*	*55-65*	*65-82.5*	*82.5-100*	*100-125*	*125-145*	*145-360*
0	2	0	1	8	0	4	0

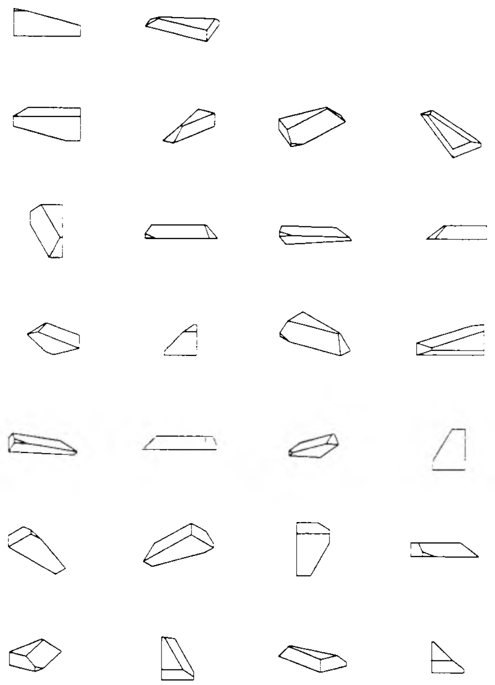
Fig. 7. Aspects for poly_1.

B. Aspect Generation

In order to determine *coverage* of the object, aspects must be determined. In generating views of an object from various viewpoints, hidden line or hidden surface removal is necessary to determine which features are visible. Aspects are formed by merging tessels which are topologically equivalent. Fig. 7 shows the 26 different aspects formed for poly_1 by merging the 80 tessels.

C. Feature Selection

The next step in the process is the evaluation of features. The *filters* are applied to the complete set of features. For the rare filter, a feature histogram is used to determine which features do not occur often in the model. Table III shows the histogram for the angle of all dihedral edges.

Robustness must be determined with respect to both the sensor and the suite of algorithms used. Through experimentation, it has been determined, for the sensor configuration used here, that an edge length under 1.0 inch cannot be reliably detected. Similarly, if a face is below a certain size, its surface area cannot be reliably detected, nor can the pointwise normals or the dihedral edges which form the face. This is because too few data points are sampled on such a small face.

Dihedral edges were selected as the most consistent feature since they solve 5 DOF's as well as satisfying the other selection criteria (e.g., robust, inexpensive, rare). For this reason, the consistent filter ranks dihedral edges as the best level 1 feature. Other features can be selected and used in conjunction with dihedral edges if dihedral edges do not suffice for all aspects.

The ability of the strategy tree to provide a path for recognition given an arbitrary object orientation is assured through the use of the aspect generation. Thus, the complete filter must be sure that at least one feature from every aspect is included as a level 1 node. It is desirable to use features which are visible from the greatest number of different viewpoints.

Feature cost has not been incorporated at this time. It is clear that algorithmic cost could be included via logical sensors and this is an area of future research.

D. *Strategy Tree Synthesis*

The order of application of these filters affects the generation of the level 1 nodes. Incorrect application of the suite of filters will generate an inefficient strategy tree. It must be stressed that a *correct* strategy tree will be built, but that the tree will be far from optimal. If the application of the filters absolutely drops features from the set, it is possible to generate a null set of features. For example, the histogram given in Table III includes all dihedral edges; even those which do not have an acceptable level of robustness. If the rare filter is applied first, edge 6 is selected as the most unique feature since only one of these edges occurs in the model. However, this edge is adjacent to a small face; thus the robustness filter would remove this edge. The only solution at this point is to generate a strategy tree with a *backup* strategy formed from a feature which is less consistent than a dihedral edge (the feature selected by the consistent filter). An example of such a feature and strategy is to look for planar faces and the associated relations between them.

Since strategy tree synthesis is automated, it is desirable to minimize the possibility of the null feature set and nonoptimal level 1 nodes. This is accomplished by *ranking* the features with the filters. Thus, each filter produces a ranked list of the current feature set. As the strategy tree is built, the application of filters now means to choose the feature with the highest rank from that set.

The order in which the filters are applied was determined through experimentation. It has been found that if the complete filter is applied first, the desired coverage is assured. From this filter, a set of aspects is produced which contains visible features. Level 1 features are selected for the strategy tree such that all aspects are represented by a level 1 node. However, one feature might be visible from multiple aspects. Using the histogram, form a set of the features which are contained in the greatest number of aspects (highest histogram value), possibly a singleton set. From this set of features, use the rare filter to determine which of these features are unique. From the ranked set of unique features, use the robustness filter to rank the robustness of each of these features. Select the feature which is most robust. If this feature is robust enough, then use it as a level 1 node. If it is not robust, repeat the algorithm for the next lowest histogram value. When a level 1 node is generated, remove, from the set of aspects, all the aspects which contain this feature. Recompute the histogram with the remaining aspects and repeat. Either a set of level 1 nodes has been generated which spans the entire set of aspects or there are aspects remaining which contain only nonrobust features. In the latter case, a *weaker* level 1 node must be formed for each of these aspects. This level 1 node will contain a feature which is not the most consistent type of feature. In this case, rather than having a dihedral edge as a level 1 node, the *back up* strategy is to match a face. At this point, the CES can be built.

One corroborating evidence subtree is generated for each dihedral edge which has attributes similar to the level 1 node. For example, for the level 1 node, edge 7, a CES must be formed for each of the edges in the 125–145 range. The reason for this is that when a 135° edge is located it should match one of these edges, but which one is not known until corroborating evidence is gathered.

The next branch in each CES is determined by looking at the ends of the dihedral edge to determine if they are occluded. Recall that occlusion is determined by the end type of a particular edge. Shadow is assumed to be occluded, jump edge depends on whether it is an occluded jump or a nonoccluded jump edge. All others are nonoccluded.

In the nonoccluded case, use the rules described above for the type feature which forms the particular level 1 node. In the example, most level 1 features are dihedral edges so the dihedral rules are used. The rules are applied in the following order.

1) Attempt to find a dihedral edge close to the endpoint of the current edge. If found, use this to quickly form a hypothesis.

2) Attempt to find the local 2-D corners. If found, these can help determine which hypothesis should be formed. For example, if a 135° edge is located, the adjacent 2-D corner can help to determine which, if any, of the 125–145 edges have been located.

3) Use the areas of the adjacent faces and relations between them to generate a hypothesis.

Fig. 8 shows the strategy tree for poly_1. The edges are represented by their edge number in the model. Note that there is a CES for each dihedral edge which is similar to the level 1 node. These are derived from Table II. For level 1 node 7, edges 1, 5, and 2 all have similar dihedral angles. Thus, there is a CES for each of these edges as well as edge 7. Note that the same CES can appear under multiple level 1 nodes. When matching, the rules on attribute similarity are used to invoke these CESs. Fig. 9 shows the corroborating evidence subtree for the dihedral edge 7. Note that there are 4 possible branches

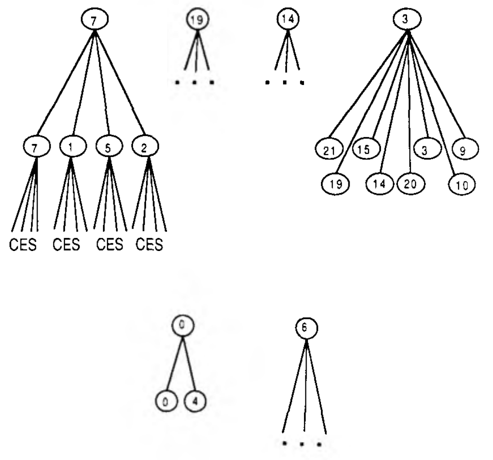

Fig. 8. Strategy tree for poly_1.

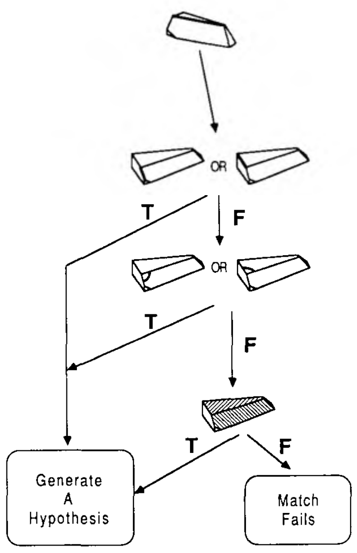

Fig. 9. Corroborating evidence subtree for edge 7.

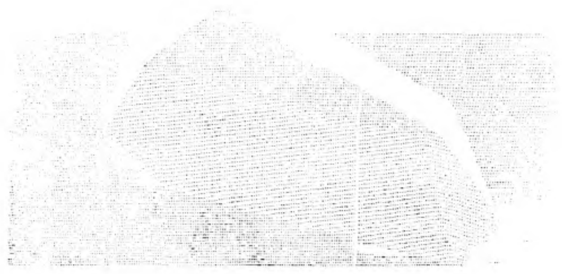

Fig. 10. Scan data for poly_1.

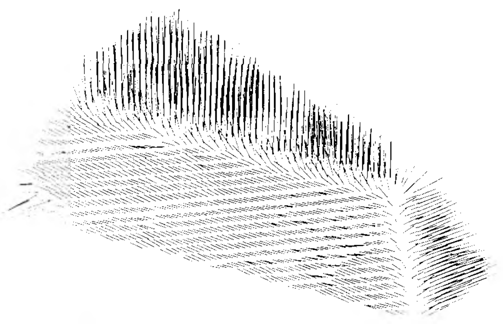

Fig. 11. Surface normals for poly_1.

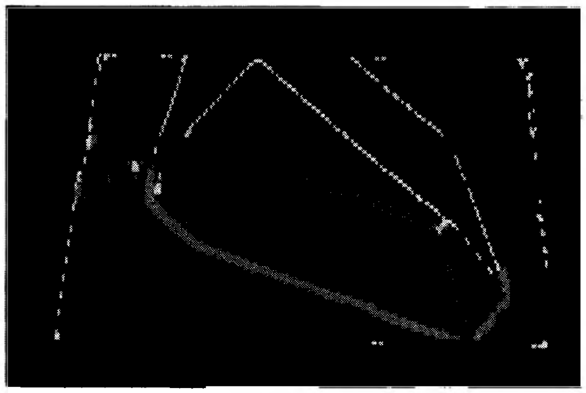

Fig. 12. Points of curvature for poly_1.

shown for clarity. The nonocclusion branch is composed of an OR of the partial occlusion cases. Thus, during runtime, the results of the partial occlusion are used by the nonocclusion branch.

E. Recognition

Now that the off-line procedure is completed, the usage of strategy trees can be demonstrated with an example of matching. A range image is obtained and low-level 3-D feature extraction performed on that data. The object is scanned, in this case poly_1. Fig. 10 shows the data for poly_1 from the Utah Range Database. This is an unsmoothed image with bad data points missing. A 3×3 Gaussian mask is used to smooth the image and replace missing data points with an average of surrounding points.

From this data, the pointwise intrinsic features are computed for the object: surface normals and surface curvature. Fig. 11 shows the surface normals for the object. Fig. 12 shows the labeled curvature for poly_1. Since this is a polyhedral object, the planar face finder is used to develop a surface representation. Fig. 13 shows the results of the planar face finder. Table IV lists attributes of the planes which were located. Two dihedral edges are located using the dihedral edge finder. These edges correspond to edge 7 and edge 1 in the model.

Now the strategy tree shown in Fig. 8 can be used. The level 1 features in the strategy tree are the dihedral edges: 7, 19, 14, 3, 4, 1, 0, 21, 6, 20, and 9. The dihedral edges located in the scene are shown in Table V. (The corresponding model edges are included to help the reader.) The system has not matched the dihedral edges at this point. By comparing these attributes to those listed in

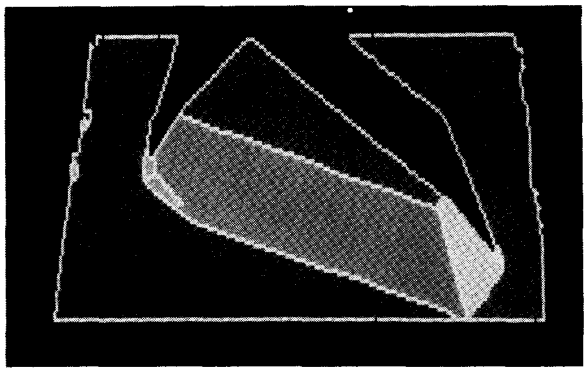

Fig. 13. Planar regions for poly_1.

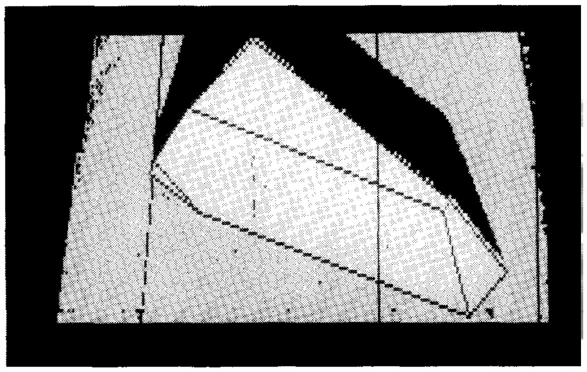

Fig. 14. Model overlayed on sensed image.

TABLE IV
ATTRIBUTES OF LOCATED PLANES

face	*area*	normal	*centroid*
1	5.799	-0.024 0.018 0.995	1.699 -2.101 -1.150
2	13.116	-0.630 0.226 0.741	-0.022 -2.324 -1.917
3	0.181	-0.128 0.899 0.391	1.177 1.167 -2.342
4	0.259	-0.695 0.588 0.392	0.710 0.858 -2.473
5	1.618	-0.448 -0.490 0.744	-0.904 -5.675 -2.070

TABLE V
DIHEDRAL EDGES LOCATED IN SCENE

detected edge	*Edges Located*		*Model Edge*
edge name	*angle*	*length*	*edge number*
A	138.393°	0.2339	2
B	136.546°	2.4619	1
C	139.558°	5.7732	7
D	150.477°	0.8748	5

Table V, the reader will notice that the attributes calculated for dihedral edge 5 are indeed erroneous. This is because the bordering face is too small to reliably recover attributes from the sensed data.

The detected edge *A* is too short for reliability so it won't be used. The detected edge *D* has an angle which does not match the model so it won't be used in the matching process. Detected edges *B* and *C* both have angles with in the 130–140 range. These edges match 2 different level 1 nodes each: model edge 7 and model edge 1. The first determination in the strategy tree is to check for similarity. If a detected edge is larger than a model edge, the match fails. Detected edge *C* fails to match the level 1 node: edge 1, because the length is too long. Next the check for occlusion takes place. Detected edge *B* is nonoccluded at both endpoints and detected edge *C* is occluded at one endpoint. Since it has been determined that detected edge *B* is a nonoccluded edge, the attributes must be close to the model for a match to succeed. For this reason, edge *B* fails to match the level 1 node: edge 7. Thus, only one Corroborating Evidence Subtree is invoked for each of the level 1 nodes which have been matched: edge 7 and edge 1.

The CES strategy first looks for an adjancent dihedral. In both cases, a dihedral is found. For the level 1 node: edge 7, the dihedral used as corroborative evidence is detected edge *B*. Whereas for the level 1 node: edge 1, the dihedral used as evidence is detected edge *C*. These two dihedrals are sufficient to solve all 6 DOF's and each of these forms a hypothesis at this point.

Since both the hypotheses are the same, the verifier only needs to check one. An image is formed with the hypothesized transform applied to the model and the perspective transform of the sensor applied to that result. For every pixel in the image, the z-depth is determined. Pixelwise evidence gathering can now be performed. The positive, negative, and neutral evidence is combined to verify or refute the match. For the hypothesized transform, the hypothesis is correct in this case. This is shown in Fig. 14.

Although the example is a polyhedral object, extensions to nonpolyhedral objects are underway. If occlusion occurs in the scene, more CES's would be invoked to corroborate possible matches. The use of this approach with multiple objects merely requires running the recognizers in parallel.

VI. CONCLUSIONS AND FUTURE WORK

It has been shown that the automatic generation of recognition strategies is possible. A method is presented which analyzed the geometric information of an object to determine the best strategy for recognition within the constraints of the sensing environment and the task. Using this information, a recognition system, a strategy tree, is produced which effectively matches models with sensed data. The strategy tree generation is performed automatically with minimal assistance from the user. The strategy tree provides a model based approach for the recognition and location of objects using 3-D sensing techniques. These strategy trees are formed using the following feature filters: robust, complete, consistent, unique, and cost effective. Using these filters, a strategy is formed which includes the use of corroborating evidence to substantiate hypotheses at formation time thereby increasing the speed for recognition.

Many areas of future research remain open. One primary area of future research is the exploration of 3-D fea-

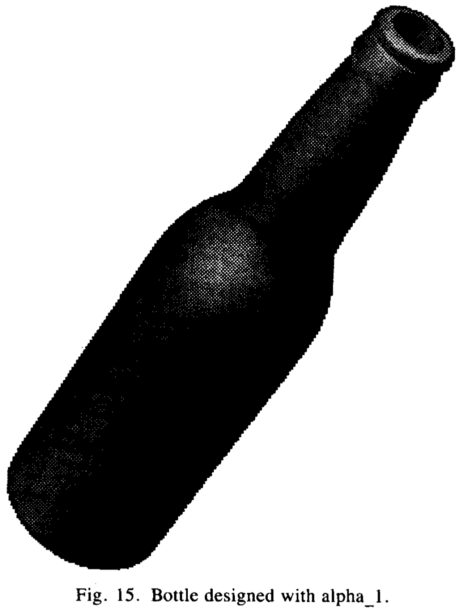
Fig. 15. Bottle designed with alpha_1.

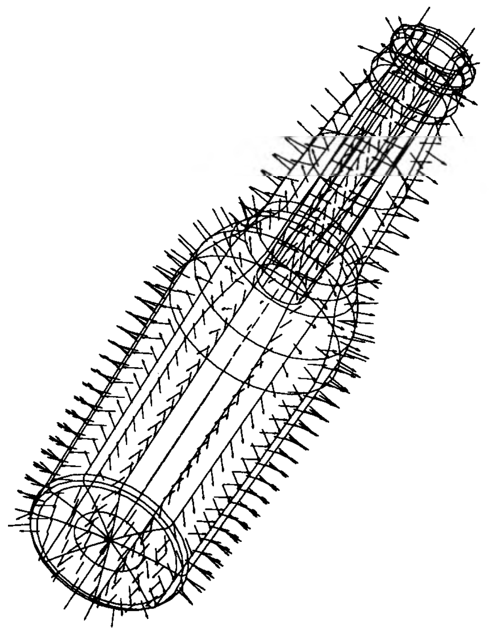
Fig. 16. Bottle normals.

ture extraction with emphasis on efficient routines. The feature extraction techniques used in this research were relatively slow when compared to the matching time. Faster feature extraction would enhance such a system.

Research into the use of other 3-D features should also be an active area. The application of these concepts to other representations, such as generalized cylinders, should be explored. Other computer vision representations, as they become available, for freeform surfaces should be incorporated into the feature selection and strat-

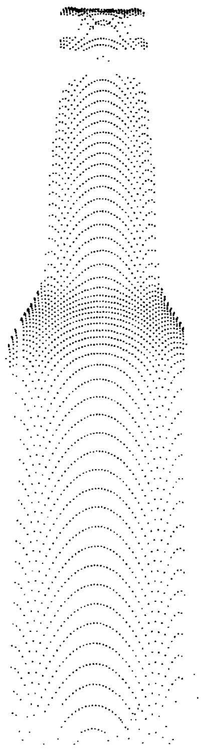
Fig. 17. Scanned bottle data.

egy generation process. Preliminary results in this area are encouraging. The bottle shown in Fig. 15 was designed using Alpha_1. A sample of surface normals on the model is shown in Fig. 16. The features for such an object include surfaces and lines of curvature (surface intersections). The main filter is robustness of their recovery in data from the laser range finder. The robustness filter eliminates lines of curvature which are formed by surfaces that are too small to be reliably detected in the range data. The selected set of features is:

cylinder:	(1.203500, 4.750000)
curve:	(0.880841, 5.839280, 2.000000, 0.726966)
curve:	(0.726966, 6.262050, 1.000000, 0.502500)
curve:	(0.485000, 9.250000, 2.000000, 0.525000).

Data from a scene with the bottle is shown in Fig. 17, and a spatial proximity graph built on those points is shown in Fig. 18. Matching to the features can be accomplished using techniques for finding lines of curvature in the data (e.g., [5]). We are currently working on this.

Another area is the use of knowledge-based techniques for the synthesis of recognizers. Specific rules have been outlined which govern the automatic generation of strategy trees. These rules could be implemented in a more general framework such as an expert system. Such a system could reason about tasking information. The representation of algorithmic information provides a vast area

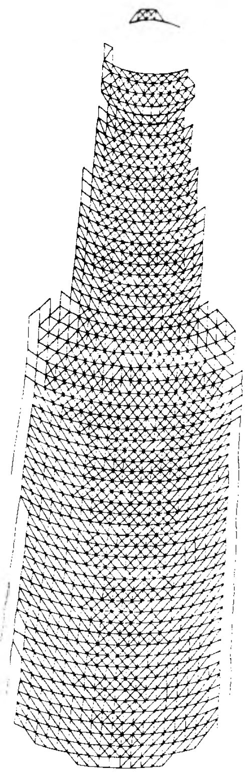

Fig. 18. Bottle spatial proximity graph.

of untapped research opportunities. The use of logical sensor specifications seems to be a good approach to the problem and should be investigated.

References

[1] B. G. Baumgart, "Geometric modeling for computer vision," Dep. Comput. Sci., Stanford Univ., Tech. Rep. AIM-249, STAN-CS-74-463, Oct. 1974.

[2] B. Bhanu and C. C. Ho, "CAD-based 3D object representation for robot vision," *Computer*, vol. 20, no. 8, pp. 19-36, 1987.

[3] R. C. Bolles and R. A. Cain, "Recognizing and locating partially visible objects: The local-feature-focus method," *Robotics Res.*, vol. 1, no. 3, pp. 57-82, 1982.

[4] R. C. Bolles and P. Horaud, "3DPO: A three-dimensional part orientation system," *Robotics Res.*, vol. 5, no. 3, pp. 3-26, 1986.

[5] M. Brady, J. Ponce, A. Yuille, and H. Asada, "Describing surfaces," in *Proc. 2nd Int. Symp. Robotics Research*. Cambridge, MA: MIT Press, 1985, pp. 45-151.

[6] C. Goad, "Special purpose, automatic programming for 3D model-based vision," in *Proc. DARPA Image Understanding Workshop*, 1983, pp. 94-104.

[7] C. Hansen and T. C. Henderson, "The UTAH range database," Dep. Comput. Sci., Univ. Utah, Tech. Rep. UUCS-86-113, Apr. 1986.

[8] C. D. Hansen, "CAGD-based computer vision," Ph.D. dissertation, Univ. Utah, Salt Lake City, Aug. 1988.

[9] C.-C. Ho, "CAGD-based 3-D object representations for computer vision," Master's thesis, Univ. Utah, Salt Lake City, Dec. 1987.

[10] K. Ikeuchi, "Model-based interpretation of range imagery," in *Proc. DARPA Image Understanding Workshop*, 1987, pp. 321-339.

[11] E. W. Kent, M. O. Schneir, and T.-H. Hong, "Building representations from fusions of multiple views," in *Proc. IEEE Conf. Robotics and Automation*, San Francisco, CA, Apr. 1986, pp. 1634-1639.

[12] T. Knoll and R. Jain, "Recognizing partially visible objects using feature indexed hypotheses," *IEEE J. Robotics and Automation*, vol. RA-2, no. 1, pp. 3-13, Mar. 1986.

[13] J. J. Koenderink and A. J. Van Doorn, "The singularities of the visual mapping," *Biol. Cybern.*, vol. 24, pp. 51-59, 1976.

[14] H. Plantinga and C. Dyer, "The aspect representation," Dep. Comput. Sci., Univ. Wisconsin-Madison, Tech. Rep. CSTR-683, Jan. 1987.

Charles Hansen received the B.S. degree in computer science from Memphis State University, Memphis, TN, in 1981 and the Ph.D. degree in computer science from the University of Utah, Salt Lake City, in June 1987.

He is currently a Visiting Assistant Professor in the Department of Computer Science at the University of Utah. He spent July 1987 through July 1988 as a postdoctoral fellow in computer vision at INRIA, Rocquencourt, France. He was an ARO fellow from August 1983 through August 1986. His research interests are computer vision, robotics, and computer graphics.

Thomas C. Henderson (M'82–SM'86) received the B.S. degree with Honors in mathematics from Louisiana State University in 1973, and the Ph.D. degree in computer science from the University of Texas at Austin in 1979.

He worked at DFVLR in Germany during 1980, and was at INRIA in France in 1981 and spent a sabbatical there in 1988–1989. He joined the Department of Computer Science at the University of Utah, Salt Lake City, in 1982 where he is presently an Associate Professor. His current areas of technical interest include multisensor systems, artificial intelligence, computer vision, and CAD-based robotics.