CAD-Based Computer Vision:
The Automatic Generation of Recognition Strategies

Charles Hansen*
INRIA
B.P. 105
78153 Le Chesnay CEDEX
France

Thomas Henderson
Department of Computer Science
University of Utah
Salt Lake City, Utah 84112
U.S.A.

Abstract

Three-dimensional model-based computer vision uses geometric models of objects and sensed data to recognize objects in a scene. Likewise, Computer Aided Design (CAD) 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 CAD and vision systems has become the focus of research in the context of manufacturing automation.

This paper explores the connection between CAD and computer vision. A method for the automatic generation of recognition strategies based on the 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 accon.plished. 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 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.

1 Introduction

Recently, 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, thus constrained, domain. This is in sharp contrast to the more general problem of attempting to reproduce an anthropomorphic vision system.

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

*author's current address is Computer Science Department, University of Utah, Salt Lake City, Utah 84112 USA

NATO ASI Series, Vol. F50
CAD Based Programming for Sensory Robots
Edited by Bahram Ravani

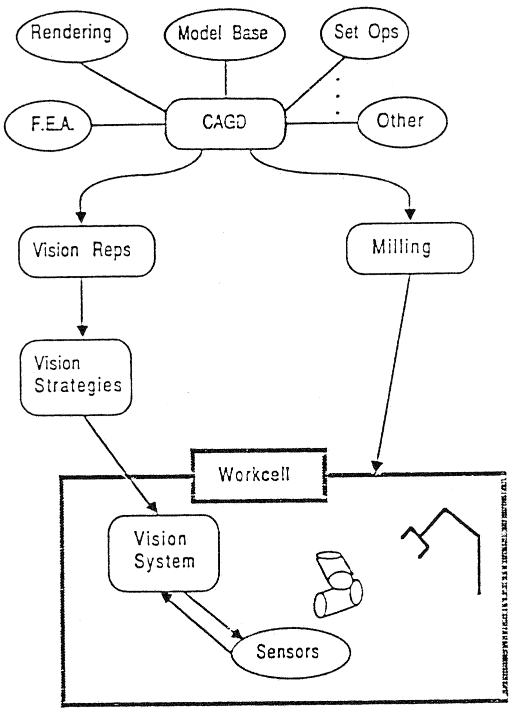

Figure 1: Integrated Automation Environment

and recognition strategies based on the CAD models. Such a system provides an integrated automation environment. Figure 1 shows how we view 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, we will focus on the automatic generation of recognition strategies based on the CAD model. It has also been determined that the use of shape, inherent in CAD 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 CAD model[1,2].

In this paper, we describe how our system automatically constructs an optimized search strategy called a *strategy trees*. These trees provide a robust mechanism for recognition and localization of three dimensional objects (occluded as well as non-occluded) 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 leading to the generation of a recognition strategy.

1.1 Related Work

Research toward the automatic synthesis of general recognition strategies has been pursued for the past several years. Goad designed a system which was concerned with automatic programming for 3-D model based vision [3]. 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 was used by an automatic programming phase. He used a unit sphere to gather *loci* of viewangles (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 didn't consider partial occlusion. However, this was a major contribution since it was one of the first attempts to automate the generation of recognition schemes.

Another very influential project in this area was the 3DPO system by Bolles and Horaud[4]. This work is the 3-D generalization of the Local Feature Focus method[5]. 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[6]. 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, adjacency 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 doesn't 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.

Others have studied the use of *interpretation trees* as a general solution to the problem of object recognition[7,8]. An *interpretation tree* provides a general matching strategy where every detected feature is matched to every possible feature in a model constrained by geometric relationships. These trees provide a bottom-up approach to object recognition since they take observed features for the scene and attempt to match these against a model. They do not provide a strategy for detecting features since every model feature can possibly occur at every level in the tree (subject to the constraints of course).

The system developed in this paper incorporates ideas from all of the systems described above. However, our system isn't 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 3-Dimensional computer vision[9]. 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. **Intermediate Representation**: 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. However, CAD models typically do not contain the explicit information needed for recognition. To obtain the geometric features of interest for 3-D recognition, techniques for the transformation from the CAD model 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. Lastly, strategy trees are defined, their use in specific domains is explained, and a technique for the automatic generation of these search trees is given.

2 Intermediate Representation

Computer vision utilizes object models in a different manner than computer graphics or CAD models. In CAD, 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. CAD 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 CAD representation. For example, a dihedral edge formed from two adjoining surfaces isn't 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.

In a previous paper, we have described how a set of intrinsic features can provide such a transformation[10]. The system we describe is not limited to a specific type of representation. Rather, all that is needed is a method for extracting the pertinent features to be evaluated and possibly used in the recognition process. In the experimental system developed here, a modified winged-edge model[11] 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 relationships 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 information. In this case, it is necessary to know which edges or surfaces are self-occluded by the object from a particular

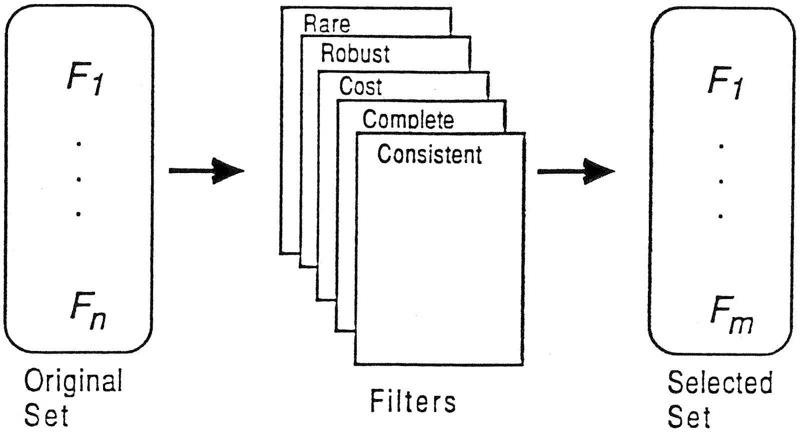

Figure 2: The Feature Selection Process

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.

3 Automatic Feature Selection

Several kinds of knowledge are required for feature selection. Geometric information permits the selection of a complete and consistent set of features, while knowledge about sensors 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 surface or other non-polyhedral objects. 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.

Conceptually, the feature selection process can be viewed as a set of *filters* applied to the complete original set of features of an object (see Figure 2). However, it should be pointed out that in some cases these *filters* only rank features rather than eliminating them. This becomes important as we generate a recognition strategy based on the feature selection process. 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 (not necessarily in this order):

- **complete** - does set of features cover all possible views of the object.
- **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.
- **consistency** - how completely does feature characterize object pose; (i.e., how many DOFs are unresolved); how well does the feature differentiate between objects; measure of likelihood of correctly identifying the object.

3.1 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.

3.2 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.

3.3 Cost of Features

The expense of feature computation can by 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 comp ıtations 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 trade off between speed and reliability of feature detection algorithms. Such knowledge needs to be utilized in this filter.

3.4 Complete Features

Three dimensional models define the entire object, yet, during scene 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. In the past, researchers have addressed this problem with stable pose analysis of both vision models and CAD models. Stable pose is a good approach if one can guarantee that parts will not be touching or overlapping (i.e. bin-picking configurations). In practice, this is generally not the case.

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[12]. 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

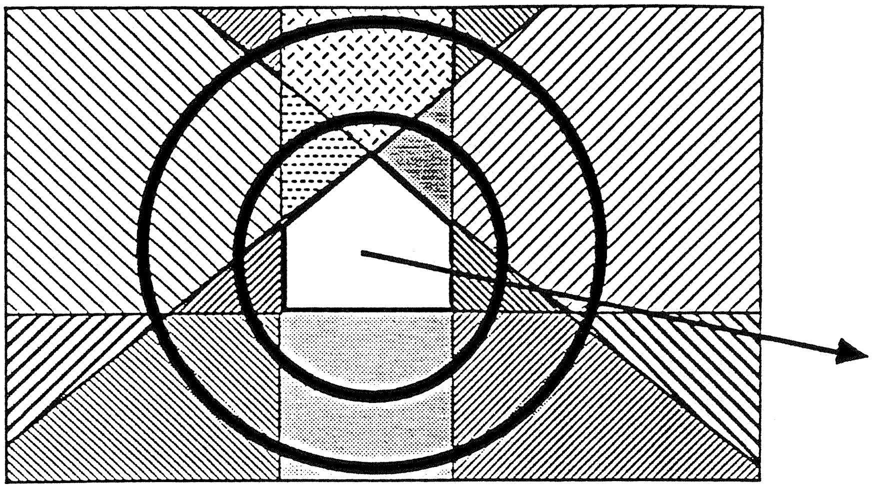

Figure 3: Aspects Depend on Both Direction and Distance

construction of aspect graphs, however, the size of the graphs poses computation limitations to their use[13,14].

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 an aspect is dependent not only on viewing direction but also on the distance from the viewer to the object, we must determine how large to scale this viewing sphere. Figure 3 demonstrates this idea. In this figure, the white pentagon represents an object and each shaded area represents a different aspect. If we look along the viewing direction represented by the arrow, we notice that the aspect changes. 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. Thus, knowledge of sensor configuration plays a role.

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, same apparent aspects are obtained, but they are spread across many more cells. Each tessellation cell, which we call 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. After all features are accumulated, neighboring tessels which contain equivalent sets of 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.

It should be noted that this is similar to approach Ikeuchi takes when generating of viewpoints for his interpretation trees[6]. 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. In our method, different classes of features are used.

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 can't be reliably detected when the sensing angle reaches a certain position, this knowledge can be used to eliminate features from tessels.

3.5 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.

3.6 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.

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.

4 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.

Figure 4 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

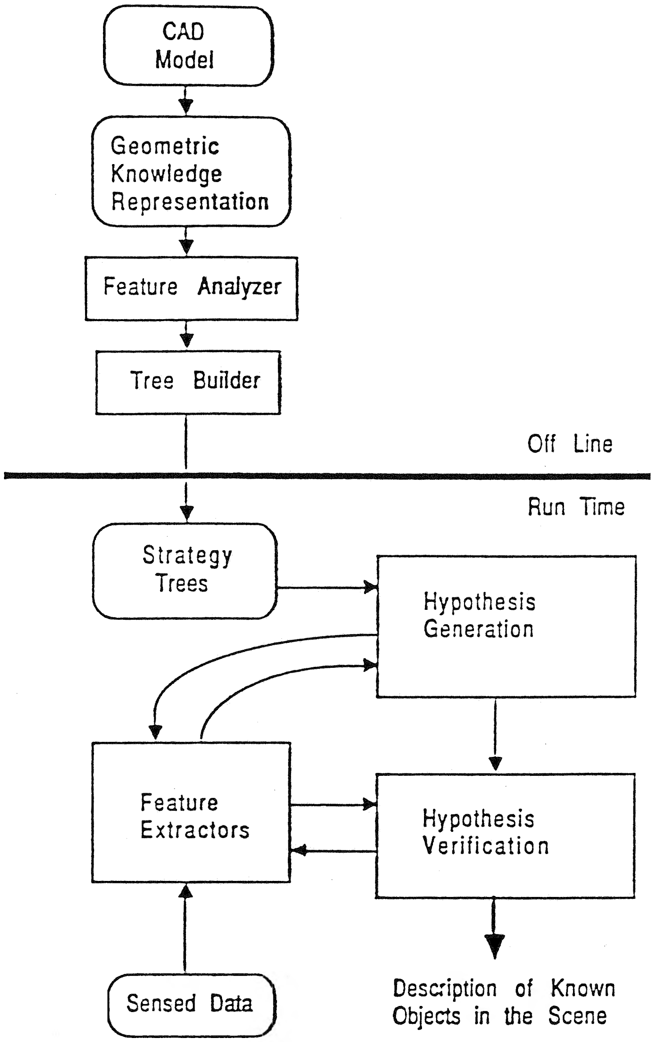

Figure 4: Overview of Strategy Trees

hypotheses are verified for correctness and a description of recognized objects and their poses are the end result. It should be stressed that the off-line processing time, while expensive, greatly reduces the amount of time spent matching. This is because the search strategy which is generated is optimized for a particular object in a specific workcell environment for a certain task. If either the object is changed or the sensing environment is changed, the strategy might also change.

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

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 permit 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
```

Figure 5: The Aspect Coverage Algorithm

expediting the hypothesis generation phase, which increases the computational expense of the verification phase.

4.1 Description of Strategy Trees

A strategy tree consists of three major parts:

1. **The Root** - Which represents the object to be recognized.
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. After level 1 features are automatically selected, they are sorted by feature class (i.e. included angle, arc radius, etc) and grouped together to form one level 1 node for each feature class.
3. **Corroborating Evidence Subtrees, CES** - Whose purpose is threefold: the discriminate between between the features within a feature class, 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.

4.2 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 Figure 5, is used to form level 1 nodes by extracting the best, unique features from the aspects.

When D not the empty set, it means there is at least one feature which is contained in all the aspects. Thus, that feature is used as a level 1 node. In the case where D is null but all the

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 D_{ij} is null for some j, then D will also be null. Additionally, it is known that the aspect, A_i is completely contained in aspect A_j. A_i must be a subset of A_j because the set difference is null and if the two aspects, A_i and A_j, contained the exact same elements, they would have been merged at the tessel stage. Since A_i is contained in 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 A_j.

From this set of level 1 features, the features are automatically sorted by feature class. These classes are determined from the feature histogram and are merged according to knowledge about the sensor resolution. That is, if the sensor is known to have a resolution such that angles can be detected with 2.5 degrees, angles within this tolerance are considered to be of the same class. Similarly, arcs of a specific radius and curvature are grouped together. One level 1 node is constructed for each feature class under consideration and the CES will discriminate between the individual features within a class.

Once the level 1 nodes are built, it is necessary to generate the CES, Corroborating Evidence Subtrees. The CESs simply substantiate that a hypothesis should be generated based on a detected feature matching with level 1 node. Sufficient evidence must be found that a correct hypothesis is being made before a hypothesis is actually generated and passed to the verification phase for validation. This process serves two purposes: find spatially local supporting evidence for the level 1 feature (discriminate between features within the feature class) 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. Furthermore, there might be several features with similar attributes or nearly similar attributes within the class. This is because Level 1 nodes were collapsed into classes of features with similar attributes. The CES discriminates between these by locating spatially local supporting evidence.

Occlusion becomes a factor during the determination of the CES strategy. Since dihedral edges and arcs provide the most consistent information (solve the most DOFs), 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 can't 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 (i.e. only a portion of the feature is visible). 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* or *regions of constant curvature*).

- **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 degrees and a dihedral edge in the scene is detected with an included angle close to 30 degrees. A CES is generated for all 30 degree angles in the model. In other words, an attempt is made to determine which dihedral was detected. The use of corroborating evidence focuses the search strategy by pruning unattractive paths at an early stage of the search.

4.3 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. Note that the strategy tree doesn't guarantee that an object will be found. If a level 1 feature is not located for a particular pose then the strategy will fail. Since level 1 features are robust, this will only happen in the presence of massive occlusion.

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 doesn't 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 **Non-occluded**
   (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 non-occluding 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 **Non-occluded**.

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 isn't 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[5,4,15]. 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 simple means that the predicted feature/depth can't be observed because something is possibly blocking sight of the object. In the presense of occlusion, it can't 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.

5 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 experiments

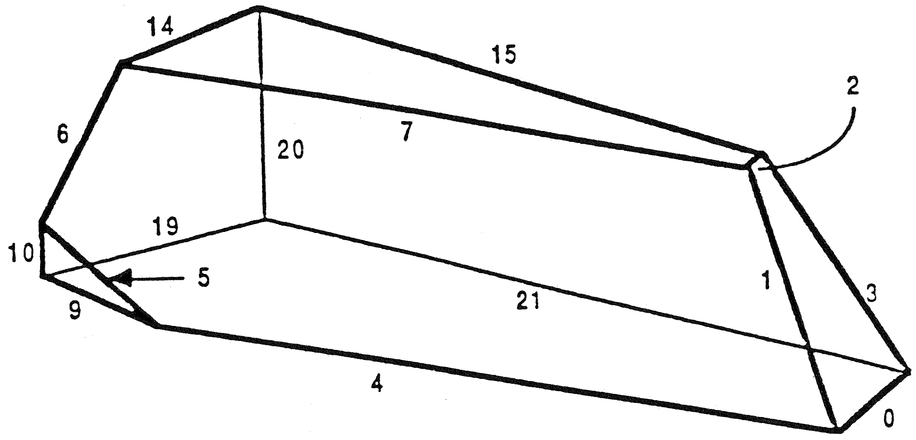

Figure 6: Wireframe Drawing for Poly_1

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[16].

5.1 Geometric Design

To demonstrate the ideas presented in this paper, we will look at a simple example. A polyhedron, we call Poly_1, was designed using the Alpha_1 design system described elsewhere[17]. 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 for straight arcs and radii and placement for circular arcs. Figure 6 shows the labeled edges of this winged-edge model of the polyhedron. The edge numbering is used throughout the remainder of this chapter. Table 1 lists the dihedral edges for poly_1. Table 2 lists the faces for poly_1. These are used by the feature selection process as well as in the generation of strategy trees. Note the grouping of the edges in Table 3 denoted by the horizontal lines. Due to noise in the data and error in the feature extraction methods, the system can't discriminate on angle value alone. Thus, dihedrals are grouped together if they are within 5 degrees of each other.

5.2 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. Figure 7 shows the 26 different aspects formed for poly_1 by merging the tessels.

5.3 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 don't occur often in the model. Table 3 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 can't be reliably detected. Similarly, if a face is below a certain size, its surface area can't be reliably detected, nor can the pointwise normals or the

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 1: Edge 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

Table 2: Face Attributes in Poly_1

				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

Table 3: Histogram of Dihedral Edges

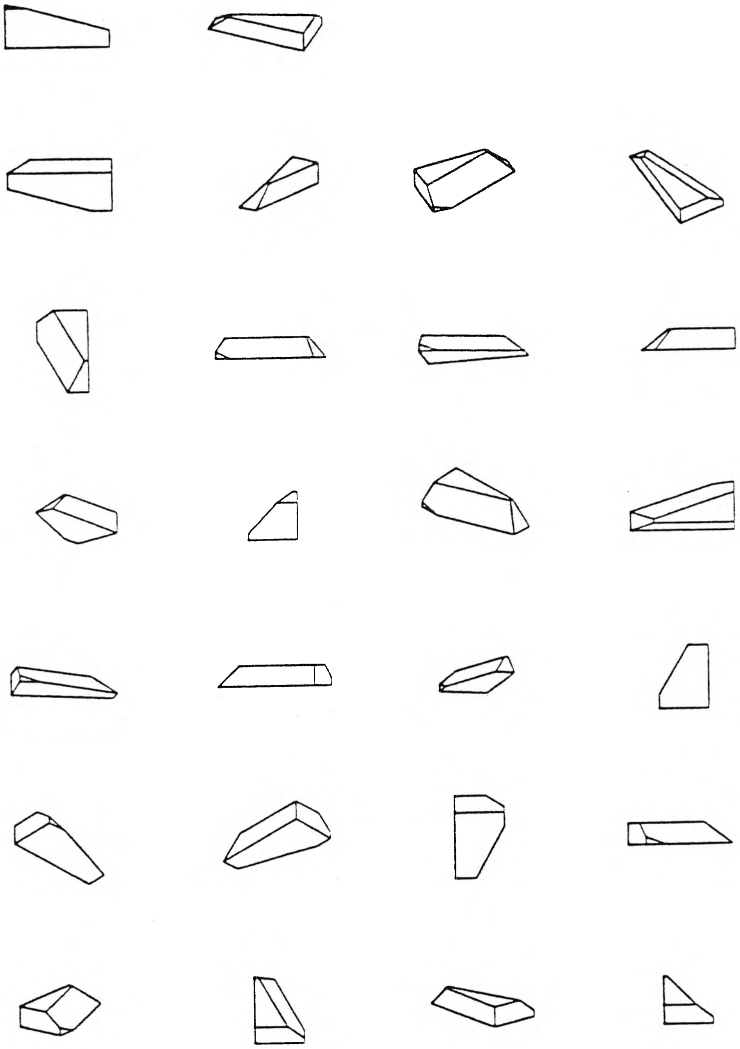

Figure 7: Aspects for Poly_1

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 DOFs. For this reason, the consistent filter ranks dihedral edges as the best level 1 feature.

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.

5.4 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 absolu-ely drops features from the set, it is possible to generate a null set of features. For example, the histogram given in Table 3 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 non-optimal 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 isn't 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 non-robust 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 isn't 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 non-occluded jump edge. All others are non-occluded.

In the non-occluded 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.

Figure 8 shows the initial level 1 features for poly_1. Figure 9 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 2. 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. Figure 10 shows the Corroborating Evidence Subtree for the dihedral edge 7. Note that there are 4 possible branches shown for clarity. The non-occlusion branch is composed of an OR of the partial occlusion cases. Thus, during run-time, the results of the partial occlusion are used by the non-occlusion branch.

5.5 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. Figure 11 shows the data for poly_1 from the Utah Range Database. This is an unsmoothed image with bad data points missing. A 3 x 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. Figure 12 shows the surface normals for the object. Since this is a polyhedral object, the planar face finder is used to develop a surface representation. Table 4 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 Figure 9 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 5. (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 with those listed in Table 5, 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 doesn't match the model so it won't be used in the matching process. Detected

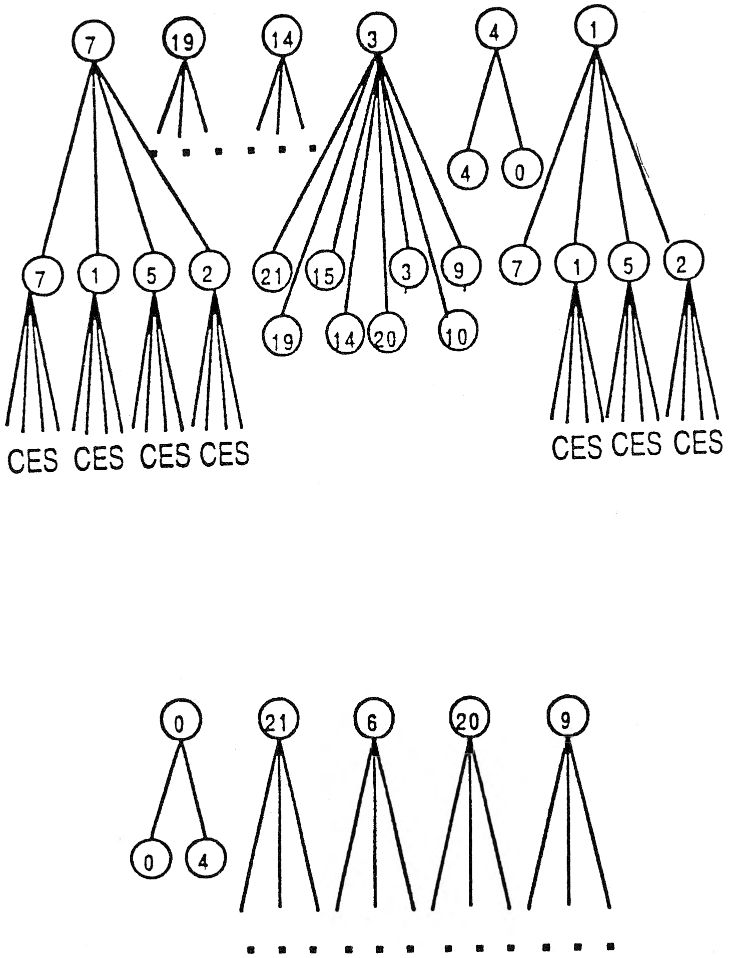

Figure 8: Level 1 Features of Poly_1

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 4: Attributes of Located Planes

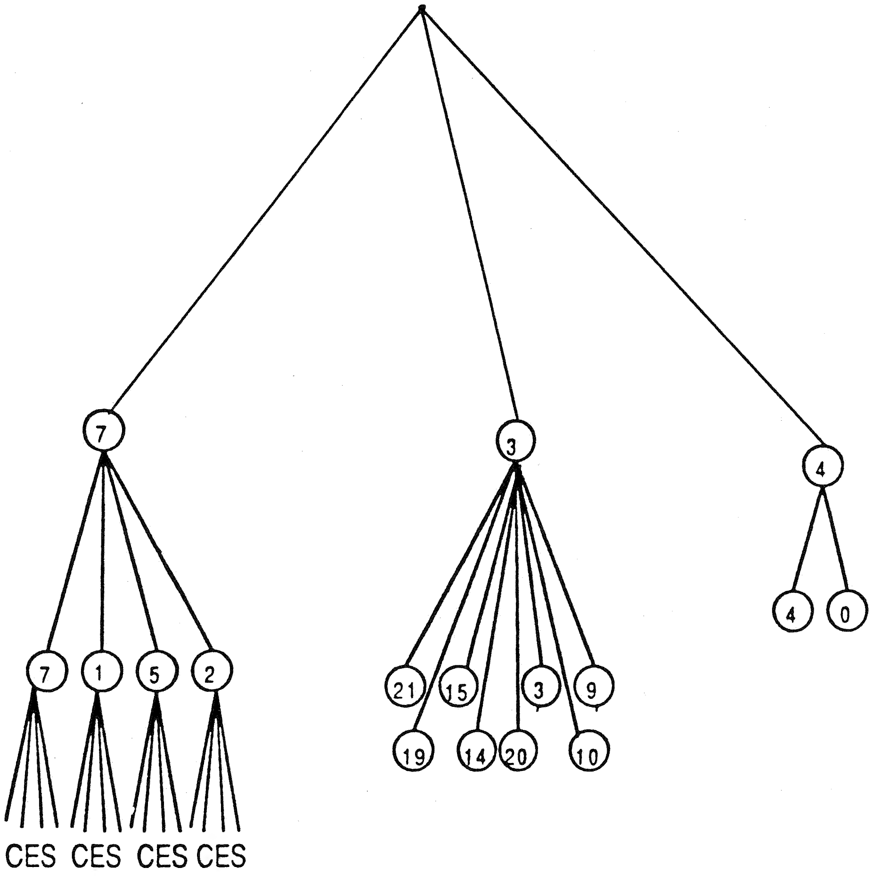

Figure 9: Strategy Tree for Poly_1

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 5: Dihedral Edges Located in Scene

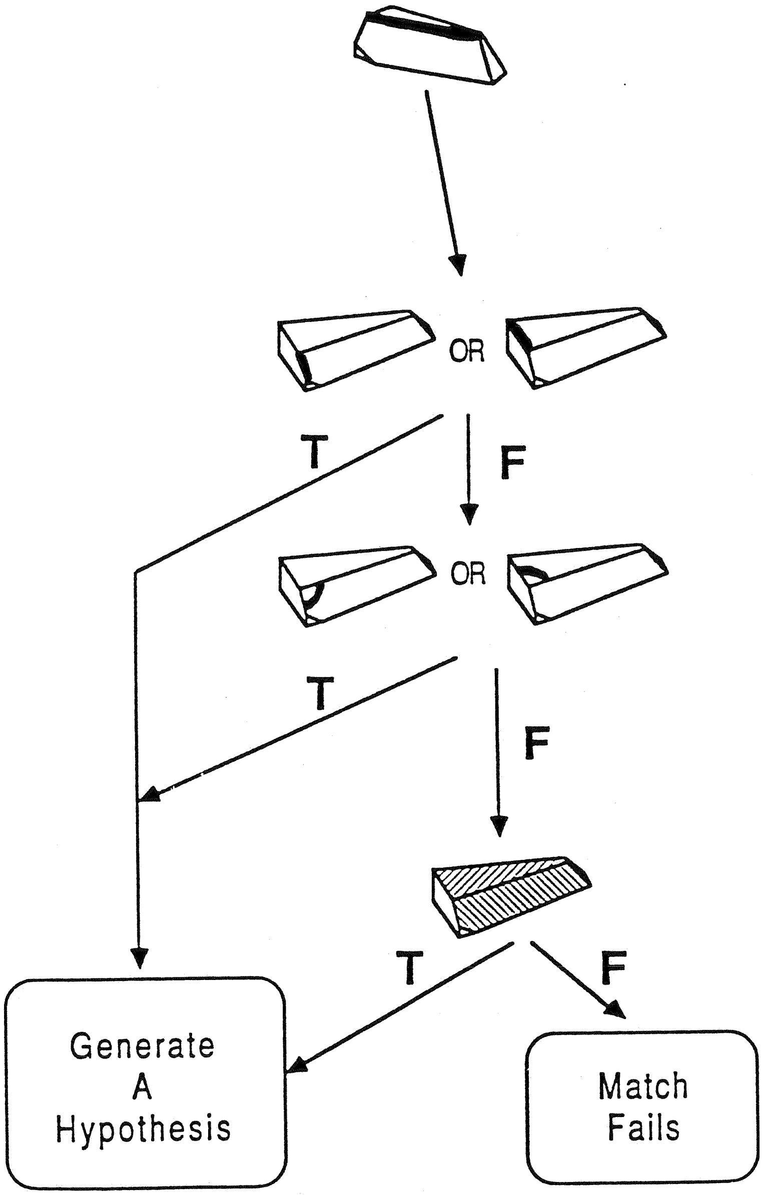

Figure 10: Corroborating Evidence Subtree for Edge 7

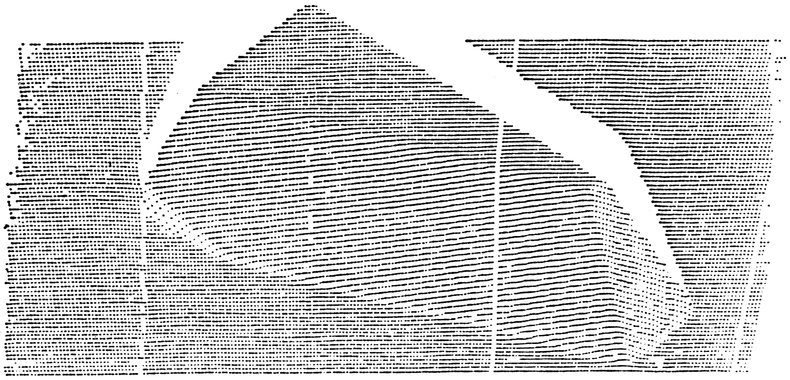

Figure 11: Scan Data for Poly_1

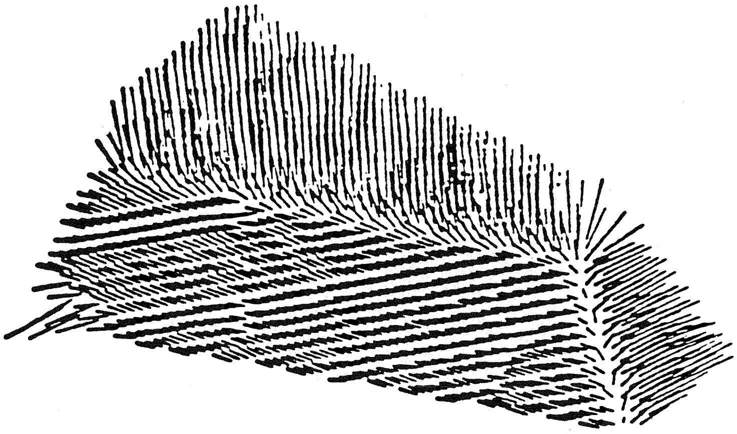

Figure 12: Surface Normals for Poly_1

edges B and C both have angles with in the 130-140 range. Both of these edges match the same Level 1 node meaing they are in the same feature class and the CES will discriminate between them. 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 node: edge 1, because the length is too long. Next the check for occlusion takes place. Detected edge B is non-occluded at both endpoints and detected edge C is occluded at one endpoint. Since it has been determined that detected edge B is a non-occluded 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 nodes which have been matched: edge 7 and edge 1.

The CES strategy first looks for an adjacent dihedral. In both cases, a dihedral is found. For the node: edge 7, the dihedral used as corroborative evidence is detected edge B. Whereas for the node: edge 1, the dihedral used as evidence is detected edge C. These two dihedrals are sufficient to solve all 6 DOFs 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.

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

6 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 feat ıre 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 strategy generation process.

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 as system could reason about tasking information. The representation of algorithmic information provides an vast area 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] Bir Bhanu and C.C. Ho. Cagd-based 3-d object representations for computer vision. *IEEE Computer*, 20(8):19–36, August 1987.

[2] Chih-Cheng Ho. *CAGD-based 3-D Object Representations for Computer Vision*. Master's thesis, University of Utah, Salt Lake City, Utah, December 1987.

[3] Chris Goad. Special purpose, automatic programming for 3d model-based vision. In *DARPA Image Understanding Workshop*, pages 94–104, 1983.

[4] R.C. Bolles and P. Horaud. 3dpo: a three-dimensional part orientation system. *Robotics Research*, 5(3):3–26, 1986.

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

[6] Katsushi Ikeuchi. Model-based interpretation of range imagery. In *DARPA Image Understanding Workshop*, pages 321–339, 1987.

[7] E. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *PAMI*, PAMI-9(4):469–482, July 1987.

[8] O.D. Faugeras and M. Hebert. The representation, recognition and locating of 3-d objects. *Robotics Research*, 5(3):27–52, 1986.

[9] Charles D. Hansen. *CAGD-Based Computer Vision: The Automatic Generation of Recognition Strategies*. PhD thesis, University of Utah, Salt Lake City, Utah, June 1987.

[10] T. Henderson, B Bhanu, and C. Hansen. Intrinsic characteristics as the interface between cad and machine vision systems. In Pierre Dejivier and Joseph Kittler, editors, *NATO ASI on Pattern Recognition*, pages 461–470, Spa, Belgium, June 1986.

[11] B.G. Baumgart. *Geometric Modeling for Computer Vision*. Technical Report AIM-249, STAN-CS-74-463, Computer Science Department, Stanford University, October 1974.

[12] J.J. Koenderink and A.J. Van Doorn. The singularities of the visual mapping. *Biological Cybernetics*, 24:51–59, 1976.

[13] E.W. Kent, M.O. Schneir, and T.-H. Hong. Building representations from fusions of multiple views. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1634–1639, San Francisco, CA, April 1986.

[14] Harry Plantinga and Charles Dyer. *The Aspect Representation*. Technical Report CSTR-683, Computer Sciences Department, University of Wisconsin-Madison, January 1987.

[15] Thomas Knoll and Ramesh Jain. Recognizing partially visible objects using feature indexed hypotheses. *IEEE Journal of Robotics and Automation*, RA-2(1):3–13, March 1986.

[16] C. Hansen and Thomas C. Henderson. *The UTAH Range Database*. Computer Science UUCS-86-113, University of Utah, April 1986.

[17] E. Cohen. Some mathematical tools for a modeler's workbench. *IEEE Computer Graphics and Applications*, 63–66, October 1983.