```
@Make(Article)
@Device(ln01)
@Style(Spacing 1, LineWidth 8.4inches, LeftMargin 0.0inches)
@Use[Bibliography="general.bib"]
@MajorHeading(CAGD-Based Computer Vision)
@begin(format)
@begin(center)
```

Thomas C. Henderson and Chuck Hansen

Department of Computer Science
The University of Utah
Salt Lake City, Utah 84112

```
@end(center)
@end(format)
@Center[@u(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 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 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.

```
@Newpage

@Center[@u(1. 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 @i{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. 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@cite{Bhanu87,Ho87}.

The work described here investigates the use of geometric knowledge in constructing @i{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.

Our main goal of is the automatic synthesis of recognition system specifications for CAD-based 3-Dimensional computer vision@cite[Hansen87]. 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:
@begin{enumerate}
@b{Geometric Knowledge 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. To obtain the geometric features of interest for 3-D recognition, techniques for the transformation to a computer vision representation have been developed.

@b{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 @i{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.

@b{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.
@end{enumerate}

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.

@Center[@b{2. 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 isn't modeled as an edge @i{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@cite{Ho87}.

In the experimental system developed here, a modified winged-edge model@cite{Baumgart74} 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.

@Center[@b{3. 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.

The feature selection process can be viewed as a set of @i{filters}
applied to the complete original set of features of an object.
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:
@begin{itemize}
@b[rare] - histogram the features; rare features are useful for
quickly identifying the object; these features make good root nodes in
a search tree.

@b[robust] - measure of how well the features can be detected; error
and reliability.

@b[cost] - measure of complexity (space and time) for computing feature.

@b[complete] - does set of features cover all possible views of the object.

@b[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.
@end{itemize}

@u{3.1 Rare Features}

The first @i{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.

@u{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.

@u{3.3 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. 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@cite{Koenderink76}.
The @i{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 @i{event} takes place. An aspect graph, or visual potential graph, is formed by representing @i{aspects} as nodes and @i{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@cite{Kent86,PlantingaDyer87}.

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 @i{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, 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
@cite{Ikeuchi87}. 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 can't be reliably detected when the sensing angle reaches a certain position, this knowledge can be used to eliminate features from tessels.

@u{3.4 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 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 trade off between speed and reliability of feature detection algorithms. Such knowledge needs to be utilized in this filter.

@u{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.

@u{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 @i{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 @i{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.

@Center[@u{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.

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.

@u{4.1 Description of Strategy Trees}

A strategy tree consists of three major parts:
@begin{enumerate}
@u{The Root} - Which represents the object to be recognized.

@u{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.

@u{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.
@end{enumerate}

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 @i{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.

@u{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 is used to form level 1 nodes by extracting the best, unique features from the aspects.

When @b{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 @b{D} is null but all the @b{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 @b{D}@-{ij} is null for some @i{j}, then @b{D} will also be null. Additionally, it is known that the aspect, @b{A}@-[i] is completely contained in aspect @b{A}@-[j]. @b{A}@-[i] must be a subset of @b{A}@-[j] because the set difference is null and if the two aspects, @b{A}@-[i] and @b{A}@-[j], contained the exact same elements, they would have been merged at the tessel stage. Since @b{A}@-[i] is contained in @b{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 @b{A}@-[j].

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 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 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 @i{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., @i{generalized cylinders}).
@begin{itemize}
@b[Dihedral Edge] rules are:
@begin{itemize}
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.
@end{itemize}

@b[Dihedral Arc] rules are:
@begin{itemize}
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 adjancent regions (area, radius of cylinder).
@end{itemize}

@b[Planar Region] rules are:
@begin{itemize}
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.
@end{itemize}

@b[Curved surface] rule is:
@begin{itemize}
Determine surface types of adjacent surfaces
@end{itemize}
@end{itemize}

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

@u{4.3 Usage of Strategy Trees}

The strategy tree @i{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:
@begin{enumerate}
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).

If the detected feature is not occluded, the attributes must be
within some tolerance of the model's values.
@end{enumerate}
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).
@begin{enumerate}
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):
@begin{enumerate}
feature is nearer than scene-jump. Implies @b{Non-occluded}

scene-jump is nearer than feature. Implies @b{Occluded}
@end{enumerate}

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 @b{Occluded}

Feature ends with neither a shadow edge nor jump edge. It is known
conclusively that the feature is @b{Non-occluded}.
@end{enumerate}

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
@i{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@cite{Bolles82,Bolles86,KnollJain86}. One of three
states is assigned to the match of the hypothesized feature or pixel
with the observed feature or pixel:
@begin{itemize}
@b[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.

@b[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.

@b[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.
@end{itemize}
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.

@Center[@u(6. CONCLUSIONS AND FUTURE WORK)]

The concepts which have been outlined above
have been implemented in an experimental system.
The synthesis of strategy
trees has been demonstrated for polyhedra and bottle surfaces.
The equipment used for the experiments 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@cite{Hansen86}.
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 x 240
image.

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.

@center(@u[7. ACKNOWLEDGMENTS])

This work was supported
in part by NSF Grants MCS-8221750, DCR-8506393, and DMC-8502115.)