```
@device(x2700)
@make(article)
@use[Bibliography="<henderson.proposals>general.bib"]
@begin(format)


@MajorHeading(The Synthesis of Logical Sensor Specifications@foot[This work
was supported in part by the System Development Foundation
and NSF Grants ECS-8307483 and MCS-82-21750.  Chuck Hansen is an ARO Fellow.]
)


@begin(center)
@b(Tom Henderson, Chuck Hansen and Bir Bhanu)

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




@end(center)

@center[@b(Abstract)]

@end(format)
A coherent automated manufacturing system needs to include CAD/CAM,
computer vision, and object manipulation.  Currently, most systems which
support CAD/CAM do not provide for vision or manipulation and similarly,
vision and manipulation systems incorporate no explicit relation to CAD/CAM
models.  CAD/CAM systems have emerged which allow the designer to conceive
and model an object and automatically manufacture the object to the
prescribed specifications.  If recognition or manipulation is to be performed,
existing vision systems rely on models generated in an @i(ad hoc) manner
for the vision or recognition process.  Although both Vision and
CAD/CAM systems rely on models of the objects involved, different modeling
schemes are used in each case.  A more unified system will allow vision
models to be generated from the CAD database.  The model generation should be
guided by the @I(class) of objects being constructed, the constraints of
the vision algorithms used and the constraints imposed by the robotic
workcell evironment (fixtures, sensors, manipulators and effectors).  We
propose a framework in which objects are designed using an
existing CAGD system and logical sensor specifications are automatically
synthesized and used for visual recognition and manipulation.
@Newpage
@Section(Introduction)
Computer vision has been an active research area for over 20 years.
In the early days, emphasis was on low level processing such as intensity
and signal processing to perform edge detection @cite(ballard82,
Rosenfeld76b).  Systems were constructed which only operated in very
constrained environments or for very specific tasks @cite(barrow78, horn70,
Witkin81).  It was quickly recognized that higher level concepts of image
@i(understanding) were needed to successfully perform computer vision.
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.

The problem we are interested in solving is model based visual recognition
and manipulation of objects in the automation environment.  This involves
building a 3-D model of the object, matching the sensed environment with
the known world and locating objects.  Not until the desired object is
located and its orientation is known can a robot gripper or hand manipulate
```

it.

Our goal is to develop a system which will work in the environment of the automated assembly process. This is not intended to provide a general model for the human visual process but rather a solution to the problem of visual recognition and manipulation in a well-known domain. The constraint we are imposing is one which limits the necessity of modeling the entire world. Rather, the known world to us is that of the automated environment in which this system is intended to operate.

Simply stated, our proposed approach is to provide an integrated environment in which the CAGD model can be used to generate appropriate recognition and manipulation strategies. A major aspect of this work is the successful development of a prototype system combining design, vision analysis and manipulation. In this paper, we address the problem of the automatic synthesis of recognition code. This synthesis is derived in terms of the shape model and the available recognition schemes and is couched in terms of logical sensor systems.

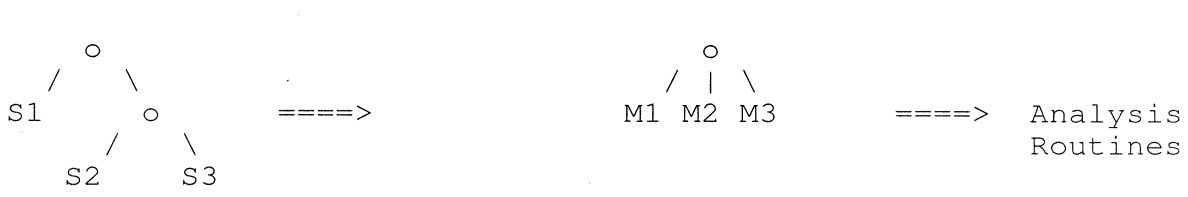
@section(The Synthesis of Recognition Strategies)
The system we describe here integrates the CAGD design system with the robotic workcell. The system contains knowledge of recognition strategies, shape representations, available sensors, and manipulation strategies. It uses this knowledge to guide the vision system and robot in the process of recognizing, locating and manipulating objects in the workcell environment.

The key issues in automatic generation of recognition strategies are:
@begin(enumerate)
generating an object model in the chosen representation from a CAGD model base, and

matching the correct shape representation with known recognition algorithms and sensors available.
@end(enumerate)
The most difficult of these these two is selecting the appropriate shape representation for the vision model. Once the shape representation is chosen, the object model for the vision system must be generated. While this may not be straightforward, algorithms can be developed which can perform this transformation. The problem of selecting a representation for the vision model is constrained by several factors. One is the availability of recognition algorithms. If we consider the available algorithms to be stored in a library, the selection can be constrained by this library. The trivial case of selecting the correct representation occurs when the recognition library of known strategies is limited to one representation. This can be considered the trivial case, even though the transformation from the CAGD model base may be nontrivial, since the selection of the shape representation is dictated by the singleton library of recognition schemes. Similarly, knowledge of the sensors available in the robotic workcell will further constrain the recognition procedure. These too can be thought of as being a library of available sensors.

The process is further complicated by the existence of CAGD models composed of multiple representations. For each complete CAGD model, there might possibly be several forms of representations contributing to the final result. If we think of the CAGD model as forming a tree of representations whose leaves are homogeneous models, we can match each of the shapes represented by these homogeneous models with some shape matching algorithm available to us in the library. Figure 1 demonstrates this idea. Consider a CAGD model to be made up of multiple structures, @B(S@-(i)), each of which might possibly be in a different representational form. For each of the @B(S@-(i))'s, the system must select an appropriate algorithm and sensor type to perform the matching in the workcell. This constrains the type vision model, @B(M@-(i)), to be used.

@Begin(Figure)
@begin(verbatim)

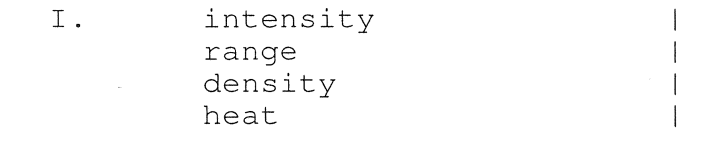


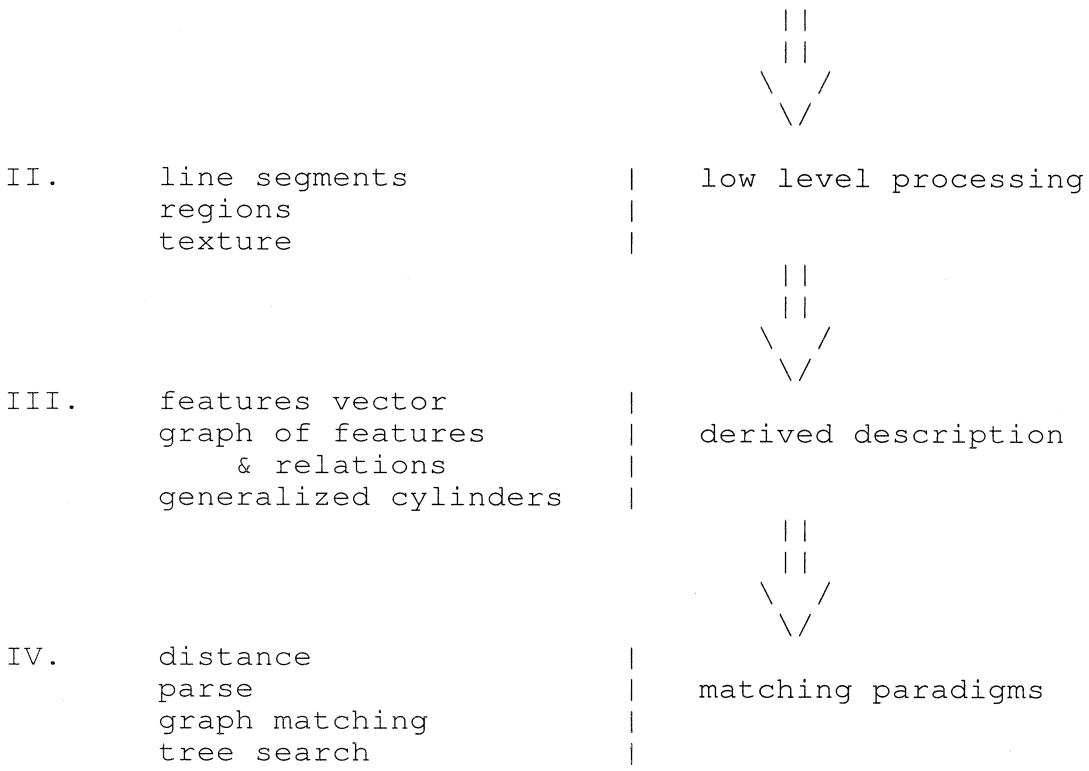
@end(verbatim)
@center(@b[Figure 1.] Relation of CAGD models to vision models)
@end(Figure)

Once the representation strategy is determined, the transformation from the CAGD representation to the recognition representation must be performed. Knowledge of this transformation is encoded along with knowledge of existing recognition algorithms. Thus, the method for the transformation can be explicit in the system. For example, if the recognition strategy uses generalized sweep, the model built from the CAGD model base would be in the form of sections of the generalized cylinder. Should planar or quadric patches be selected, the representation for recognition would be a graph structure of relations between the patches. If feature vectors are the chosen method for recognition, the features can be extracted directly from the CAGD model or the CAGD system might first produce an image of one view of the object then the features can be extracted by the same algorithm which processes the sensed data. We have implemented such an automatic inspection capability and give more details below.

An extreme method for generation of recognition strategies is parameterization. This is extreme because the user is required to @I(fill in the blanks) for the sensors and algorithms for the particular object, or class of objects, modeled. Obviously, a more automated system is desired for this task. Drawing from our experience with Logical Sensor Specification in the MKS system, our proposed method is to combine several algorithms and sensors to form a specialized @I(object finder) @cite(hansen83, henderson83d, henderson84c, henderson84f, henderson85, henderson85a, henderson85c, henderson85e, henderson85h).
The methodology provides a means for abstracting the specification of a sensor from its implementation along with providing transparency of hardware and software above the implementation level. Alternatively, we are also investigating how to embed knowledge of the algorithms and sensors in
the system and to provide a rule base for the decision process. This requires a complex expert system (see@cite(henderson84e) for a description of a preliminary system). In either case, the system will eventually be composed of multiple sensors and recognition methods.

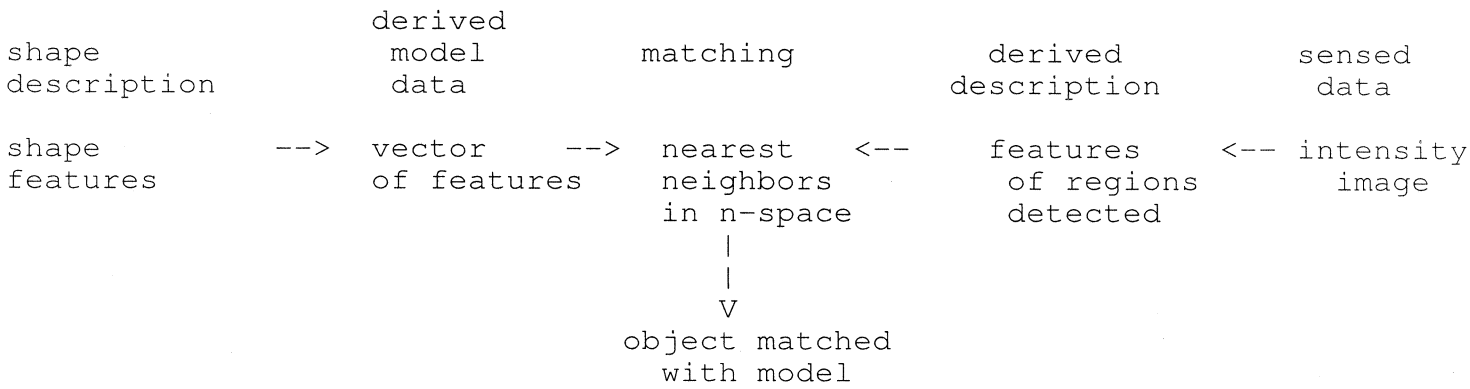
There are different recognition methods which have been successfully applied but have never been unified in a single system. We propose to include multiple recognition algorithms in the system, each of which might require a different vision model representation scheme. Figure 2 represents the general schema we have in mind for the generalization of recognition methods. The concept is to choose the proper element from @B(I), @B(II), @B(III), and @B(IV) for each of the @B(S@-(i)) in Figure 1.
@begin(Figure)
@begin(verbatim)





@end(verbatim)
@center(@b[Figure 2.] Schema for Generating Recognition Strategies)
@end(Figure)
To better understand this, let us look at a specific recognition strategy.

Feature vectors are representative of both currently available commercial systems and on going research efforts @cite(Bolles82). In feature vector matching, the model is a set of features of the particular object in vector form. When a scene is analyzed, all objects are segmented into distinct regions and the pertinent features are extracted. These are then used to recognize known objects usually with a nearest neighbors method in n-dimensional space. Figure 3 is demonstrates how this method is applied.
@begin(Figure)
@begin(verbatim)



@end(verbatim)
@center(@b[Figure 3.] Schema for Feature Vector Matching Paradigm)
@end(Figure)