CAGD-Based 3-D Visual Recognition

Tom Henderson, Chuck Hansen, Ashok Samal, C.C. Ho and Bir Bhanu

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

Abstract

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 *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. We are implementing a framework in which objects are designed using an existing CAGD system and recognition strategies based on these design models are used for visual recognition and manipulation. An example of its application is given.

1. 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. Systems were constructed which only operated in very constrained environments or for very specific tasks. It was quickly recognized that higher level concepts of *image 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.

[1]This work was supported in part by NSF Grants ECS-8307483, MCS-82-21750, DCR-8506393 and DMC-8502115.Chuck Hansen is an ARO Fellow.

Simply stated, our 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 describe:

1. the design of an object using the Alpha_1 CAGD system,
2. the analysis of 3-D range data to find the object,
3. the recognition of the object using a new multi-constraint discrete relaxation matching algorithm.

2. Object Design

The current modeling environment is the Alpha_1 Computer Aided Geometric Design (CAGD) System [3]. It models the geometry of solid objects by representing their boundaries as discrete B-splines. It allows the construction of simple objects into a more complex object using set operations. It supports several modeling paradigms, including direct manipulation of the B-spline surface, creation and combination of primitive shapes, and high-level operators such as bend, twist, warp and sweep. By using set operations on sculptured surfaces, the modeling task becomes simpler and more complete than with other design systems.

The guidelines which are followed in using Alpha_1 are:

1. Analyze the object. Usually a complex object can be divided into simpler parts which can be more easily designed. A rule of thumb is "Use the same procedure that people use to create the object." Once the procedure is decided, one can concentrate on each subpart.
2. Measure parameters. Make a precise measurement of parameters. If the design data is available, it is best to use it.
3. When the surface patches of each part are designed, make sure that they have the correct orientation and set the adjacency information correctly. Otherwise, an object which is not valid may be created.
4. Put all subparts in the correct position and orientation. Then use the combiner to perform the appropriate set operations.

Now we show the design of a typical object which we'll call *green piece*. The object is designed in a sequence of steps starting with the main plate, then adding indentations and all the holes. To obtain these, we design curves using B-splines first and then various high-level operators for surface construction; e.g., revolving a curve about an axis, extruding a curve, and filling in the surface between curves. Threads in the holes are designed by filling two surfaces between two twisted curves. The final design is shown in Figure 1. Based on

the design information the 3-D features listed in Figure 2 were extracted. These were used for matching purposes in the analysis of the 3-D data.

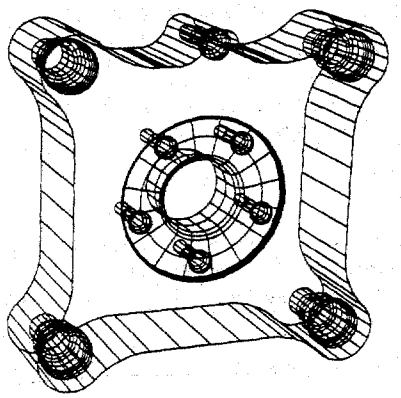

Figure 1. CAGD Design of Green piece

```
% Model definition for Model-1

(setf ModelGP
  (MakeModel
   (Model-Name 'ModelGP)
   (Model-Origin '(0.0 0.0))
   (Model-Feature-Instance-List
       '((Hole865    H1  2.000 2.000 0.0)
         (Hole575    H2  0.375 0.375 0.0)
         (Hole575    H3  0.375 3.625 0.0)
         (Hole575    H4  3.625 0.375 0.0)
         (Hole575    H5  3.625 3.625 0.0)
         (Hole330    H6  2.000 3.835 0.0)
         (Hole250    H7  1.250 1.750 0.0)
         (Hole250    H8  1.563 2.563 0.0)
         (Hole250    H9  2.438 2.563 0.0)
         (Hole250    H10 2.700 1.750  0.0)
         (Hole250    H11 2.000 1.250 0.0)))))
```

Figure 2. 3-D Features Extracted from Model

3. 3D-Data Analysis and Object Recognition

The system we are developing is based on the notion of *specialization*. This means that we take advantage of any particular information that can be culled from the CAGD shape model. This knowledge is then encapsulated in a special package which provides for the recognition of an object or part of an object. Thus, instead of using a general recognition technique on all parts to be recognized (i.e., a weak method), we produce specially packaged code (i.e., logical sensors) for recognition. These are then instantiated independently as needed, and controlled as logical sensors.

The approach consists of three phases:

1. **Design.** The object is designed using a CAD modeling system (the Alpha_1 CAGD system in our case). This aspect was explained in the previous section.

2. **Derivation of Intrinsic Features.** The recognition strategies are based on matching intrinsic features of the object's shape with those of unknown shapes. A set of intrinsic features are derived from the CAGD system, and includes such features as: genus, surface points, number and placement of holes, color, texture, surface normals, surface curvature, etc. See Henderson and Bhanu [6] for more details on our approach to the use of intrinsic features as the interface between CAD and computer vision systems [1, 8].

3. **Synthesis of Object Recognition Strategies.** Given a set of intrinsic features for a specific object and knowledge of the representation chosen in the CAGD system (e.g., Constructive Solid Geometry, Generalized Cylinders, or Boundary Representation), plus knowledge of the available recognition techniques and feature detectors, the system will choose and hook together the appropriate recognition code. This is currently done by means of parameters, but eventually will require more expertise in using the knowledge that the system has available.

A straightforward method which we use for generation of recognition strategies is parameterization. The user is required to 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 and is under investigation [5]. The methodology, referred to as Logical Sensor Specification (**LSS**) 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 [4] for a description of a preliminary system). In either case, the system will eventually be composed of multiple sensors and recognition methods.

The final part of the system is the matching component. We have recently introduced the notion of "split-level" relaxation [7], and we have applied it here to the problem of labeling 3-D features. This approach is similar to Local Feature Focus [2]. The method is fully described elsewhere in these proceedings. One of the first steps in locating an object is to locate its features. We can recognize objects on the basis of *global* features, like number of holes, size of various segments, total area of the segments, perimeter, etc. Alternatively we can also use local features to locate objects; e.g., corners, holes, etc. We look for certain structure with respect to these local features in the image, and if we can find such a structure then we can locate the object.

4. An Example

Figure 3 shows a range data view of an actual milled version of the piece modeled in Section 2. The features extracted from that data are given in Figure 4. These were fed to the split-level relaxation matcher and the results are shown in Figure 5.

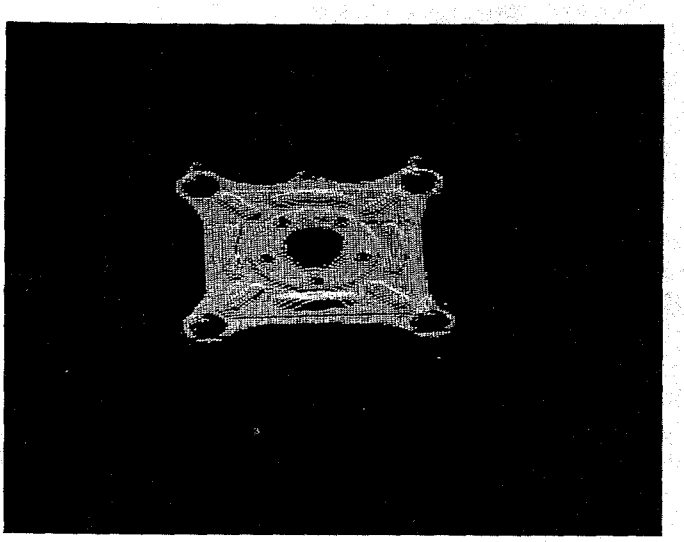

Figure 3. Range Data View of Green Piece

```
% Definition for Image-1

(setf Image
  (MakeImage
   (Image-Name 'ImageGP)
   (Image-Origin '(0.0 0.0))
   (Image-Feature-Instance-List
       '((Hole575    Hole1   1.520 -3.652 -0.487)
         (Hole575    Hole2   1.558 -0.394 -0.465)
         (Hole250    Hole3   0.800 -2.255 -0.943)
         (Hole250    Hole4   0.528 -1.363 -1.376)
         (Hole865    Hole5   0.149 -1.958 -1.294)
         (Hole250    Hole6   0.151 -2.762 -1.299)
         (Hole250    Hole7  -0.233 -1.367 -1.503)
         (Hole250    Hole8  -0.484 -2.204 -1.634)
         (Hole575    Hole9  -1.330 -0.344 -2.038)
         (Hole575    Hole10 -1.375 -3.603 -2.039)))))
```

Figure 4. Features Extracted from Range Data

```
%% Output for
Split-level Relaxation

Input File is params.sl

Model definition file is modelgp.sl
Model constraints file is modelgpC.sl
Image definition file is imgp.sl
Image constraints file is imgpC.sl    Optimal distance = 1.10105

Initial statistics
   Total Number of Nodes = 10
   Number of nodes with one label = 1
   Average number of labels/node = 3.0

Iteration Number = 1 Time = 51 ms
   Number of nodes with one label = 3
   Average number of labels/node = 2.5

Iteration Number = 2 Time = 1122 ms
   Number of nodes with one label = 5
   Average number of labels/node = 1.5

Iteration Number = 3 Time = 238 ms
   Number of nodes with one label = 6
   Average number of labels/node = 1.4

Iteration Number = 4 Time = 238 ms
   Number of nodes with one label = 6
   Average number of labels/node = 1.4

Total time for ARC Consistency = 1683 ms

There is only one labelling for the above

   Primtive = HOLE10 : Label = H2
   Primtive = HOLE9 : Label = H3
   Primtive = HOLE8 : Label = H7
   Primtive = HOLE7 : Label = H8
   Primtive = HOLE6 : Label = H11
   Primtive = HOLE5 : Label = H1
   Primtive = HOLE4 : Label = H9
   Primtive = HOLE3 : Label = H10
   Primtive = HOLE2 : Label = H5
   Primtive = HOLE1 : Label = H4
```

Figure 5. Matching Results

5. Conclusions

We have been studying techniques and algorithms which allow the generation of computer representations and geometric models of complicated realizable 3-D objects in a systematic manner. In order to produce recognition strategies for a machine vision system, it is necessary to specify the interface between the CAD system and the machine vision analysis system. This interface can be characterized by the set of intrinsic 3-D shape characteristics which can be produced by the particular CAD system under consideration. Given the set of intrinsic features, the system can generate the necessary recognition strategies.

We have demonstrated that these concepts are realizible by designing an object and generating a particular model for a certain recognition strategy. Future work includes extending the system to include more shape characteristics for model building as well as for selection of the proper recognition scheme.

This approach to model-based 3-D data analysis shows great promise. Moreover, the inherent capability to automate the generation of recognition and manipulation code will make systems such as the one discussed here a very important tool in the automation industry.

References

[1] Barrow, Harry and Jay Tennenbaum.
Recovering Intrinsic Scene Characteristics from Images.
Technical Report 157, SRI International, April, 1978.

[2] Bolles, R.C. and R.A. Cain.
Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method.
Robotics Research 1(3):57-82, 1982.

[3] E. Cohen.
Some Mathematical Tools for a Modeler's Workbench.
IEEE Computer Graphics & Applications :63-66, October, 1983.

[4] Henderson, T.C., Bir Bhanu, C.D. Hansen and E. Shilcrat.
ASP: A Sensor Performance and Evaluation System.
In *Proceedings of Pecora IX*, pages 201-207. October, 1984.

[5] Henderson, T.C., C.D. Hansen, and Bir Bhanu.
The Specification of Distributed Sensing and Control.
Journal of Robotic Systems 2(4):387-396, 1985.

[6] Henderson, T.C., Bir Bhanu and Chuck Hansen.
"Intrinsic Characteristics as the Interface between CAD and Machine Vision Systems.
Pattern Recognition Letters 3:425-430, December, 1985.

[7] Henderson, Thomas and Ashok Samal.
Split-Level Relaxation.
Technical Report UUCS-85-113, University of Utah, Department of Computer Science, November, 1985.

[8] Witkin, A. P.
Recovering Surface Shape and Orientation from Texture.
Artificial Intelligence 17:17-45, 1981.