Journal of Intelligent and Robotic Systems **14**: 323–340, 1995.

A Sensing Strategy for the Reverse Engineering of Machined Parts*

TAREK SOBH and JONATHAN OWEN
Department of Computer Science and Engineering, University of Bridgeport, Bridgeport CT06601, U.S.A.

(Received: 23 September 1993; in final form: 25 May 1994)

Abstract. The reverse engineering of machined parts requires sensing an existing part and producing a design (and perhaps a manufacturing process) for it. We have developed a reverse engineering system that has proven effective with a set of machined parts. This paper describes the system, presents some results, and discusses strategy for a new system.

Key words. Reverse engineering, representation, machined features.

1. Introduction

CAD/CAM (Computer Aided Design, Manufacturing) often involves the design and manufacture of machined parts. The problem of reverse engineering is to take an existing mechanical part and to inspect or produce a design, and perhaps a manufacturing process, for the part. The main purpose of our work is to explore the difficulties involved in the reverse engineering problem, and develop a robust and efficient solution.

Reverse engineering has a variety of potential applications. It may be illegal for an organization to obtain the original model or drawings for a part, but legal to reverse engineer it. Original plans may have been lost or have been completed before CAD was in vogue. Some design is still done using clay models, which then must be digitized. A manufacturer may go out of business, making replacement parts difficult to obtain. An organization may wish to analyze a competitor's product.

In this paper, we describe our research in reverse engineering. The background used in developing our research is discussed below, followed by a discussion of the research itself. Aspects that are discussed include two- and three-dimensional image processing, and a sensing to CAD interface. Results are presented, followed by discussion of a strategy for further research.

* This work was supported by ARPA under ARO grant number DAAH04-93-G-0420, DARPA grant N00014-91-J-4123, NSF grant CDA 9024721, and a University of Utah Research Committee grant. All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

2. Background

Relative to other fields, few published results are available on reverse engineering. One example is the work of Sarkar and Menq [17], who approximate surfaces containing multiple features by least-squares fitting of B-splines. The features they consider are *geometric* features such as cylinders, cones, spheres, and sculptured surfaces. They segment the data (3,000 3D points) from a CMM into regions based on the $\nabla^2 G$ operator. B-splines are then fit to these regions and optimized for error in a computationally efficient manner. Their problem domain consists of objects produced in the die and mould industry. In this industry, designs are often modified on the shop floor (depending on production requirements), and the original designs must be updated to match the manufactured parts.

Motavalli and Bidanda [16] describe a reverse engineering system which uses a traditional image processing approach to acquire range data (turntable and laser/triangulation), process it (smooth, fit curves, etc.), and produce engineering drawings in the form of orthographic projections. This was performed on concentric rotational parts and the results were judged suitable for use with a lathe.

Hoppe *et al.* [8] have developed an algorithm which takes an unorganized set of 3D points on or near a surface as input and generates a simplicial surface that approximates it. They also present a method for optimizing the mesh [9], and in recent results [10], fit a piecewise-continuous surface to it. Results are presented for a plaster model of a cat and a distributor cap from data taken with a laser scanner.

Although reverse engineering does not have a large body of literature to itself, related fields such as object recognition have many things in common. Research in the following areas was reviewed:

1. Sense data/sensor types.
2. Multiple view integration/occlusion avoidance.
3. Feature interaction.
4. Segmentation.
5. Feature extraction.
6. Surface representation/reconstruction.

The first three will be discussed briefly.

2.1. SENSE DATA

Much research has been devoted to developing systems that sense range information. These systems can be categorized as active or passive, depending upon

whether they project energy onto a scene or not [4].*

Passive systems typically use gray-scale intensity images as input. These systems compute shape using binocular stereo, motion, shading, etc. Passive systems are useful for a wide variety of applications. For example, it may not be practical to use an active system when stealth is required or when distances are large (as in astronomical observation).

For more controlled situations, as in dealing with industrial parts, active system such as laser range finders are an attractive alternative to passive systems. The target may be densely sampled and the output is in the form of a range image, which directly encodes the information of interest. Although much progress has been made in understanding the image formation process in recent years, intensity images encode much more information than shape. Reflectance, texture, illumination, camera parameters, etc. must all be considered in computing shape from an intensity image. With the advent of low-cost, accurate active range systems, the use of range images is becoming widespread.

Ultrasound and radio wave range finders are currently used mostly for low-resolution applications, such as navigation aids. Several methods exist for using lasers as range finders, and non-coherent white light may be used in a similar fashion. Range may be measured by measuring the amount of time required for a pulse to travel to and from a target. Range may also be measured by detecting the phase difference between a received wave and a reference signal. Less sophisticated, but effective are triangulation-based range finders project a spot, strip, or pattern on the target and use trigonometry to compute range. Triangulation schemes are susceptible to another form of the occlusion problem in that the camera and source must be separated from each other, causing shadow effects.

Arman and Aggarwal [3] recently published a comprehensive survey of model-based vision system using dense range images.

2.2. MULTIPLE VIEW INTEGRATION/OCCLUSION AVOIDANCE

Although a single view may provide enough information to recognize an object from a limited set, occlusions often will hide needed features. Thus some researchers have directed efforts at integrating multiple views. This research is very useful in reverse engineering since the goal is to generate a complete model of the object and it is vary rare that a single view will provide adequate information.

Chen and Medioni [5, 6] describe a technique for integrating multiple view data by projecting a triangulated shell onto the data points. An iterative subdivision technique adjusts the triangulation to a desired error tolerance. Results are

* This may be a different definition for active vision than the one with which the reader is familiar. Another definition considers any motion of the camera or changing of the camera's parameters to be active vision.

presented for a plaster model of a human tooth, a free-form wooden 'blob', and more recently [7], a telephone handset.

Another aspect of view integration is the integration of data from other types of sensing. One type that is of particular interest to us is touch data, as gathered by a *coordinate measuring machine* (CMM). Researchers who combine vision and touch include [1] and [18]. Allen successfully integrated touch data gathered by a sensor on a Puma with passive stereo data to fit Coon's patches to objects including a vase, cereal bowl, and coffee cup. Stansfield used a similar setup to develop a more general framework for object recognition. Related to this work is the sculptured surfaces inspection performed by [11]. Hsieh assumed that a crude model was available (possibly from vision), and used it to gain an accurate description of a variety of surfaces using a CMM. These surface included a sculptured pocket, compressor blade, a human femur, and a human vertebra.

2.3. FEATURE INTERACTION

Also related to our research is the field of feature interaction. Our review of the literature found little information about feature interaction in the vision community. However, several efforts in CAD modelling, process planning, and inspection show pertinent information.

[14] describe an algebraic representation of feature interactions that aid in generating manufacturing spesifications from a CAD model. Given a set of features that describe a machinable part, they search for combinations of these features that optimize the manufacturing process for accuracy, speed, and cost. To understand how accuracy may be affected by the process plan, consider a cantilever beam with a thru-hole (see Figure 1). This beam must be cut from the center of a larger block. If the hole is drilled after the beam is cut out, errors may be induced due to vibration of the beam. If cut before, the drilling process will be longer, but more accurate. The two situations are geometrically equivalent, but have different effects.

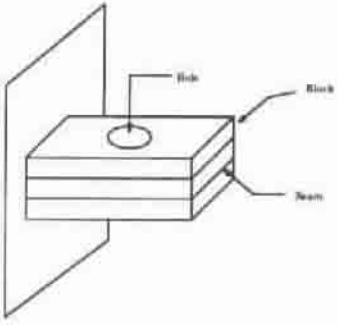

Fig. 1. Interaction of hole and two cuts on cantilever beam.

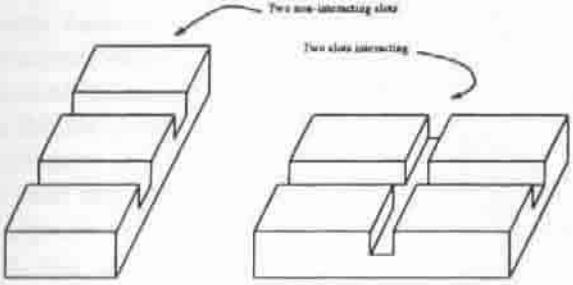

Fig. 2. Interaction of slots.

[15] describe an integrated system for design, process planning, and inspection that reasons about the interactions between features to ensure proper inspection. An example they describe is where two perpendicular slots intersect on a face of a machined part (see Figure 2). The resulting object that must be inspected has not two pairs of parallel line segments, but two pairs of parallel lines, each of which must be inspected along two line segments.

3. Sensory Processing

The sensory processing utilized in our research included a combination of 2D and 3D levels. These levels were used to extract several properties of an object that were then used to construct a rough CAD model as described later. These properties included:

- Number of features.
- Contour representation of each feature.
- Relationships of the features.
- Approximate depth estimates between contours.
- Depth of features.

Experiments were conducted using a black and white CCD camera, mounted to the end effector of a Puma 560 robot arm. The arm was used to position the camera in the workspace, moving between calibrated positions. The camera supplied 640 × 480 pixels to a VideoPix video card in a Sun Workstation. Low level image processing was done using the IMLIB image processing library routines developed at the University of Utah [12].

3.1. TWO DIMENSIONAL IMAGE PROCESSING

The camera captures the current image and passes it to the 2D image processing layer. The main goal of this layer is to discover geometric features. These

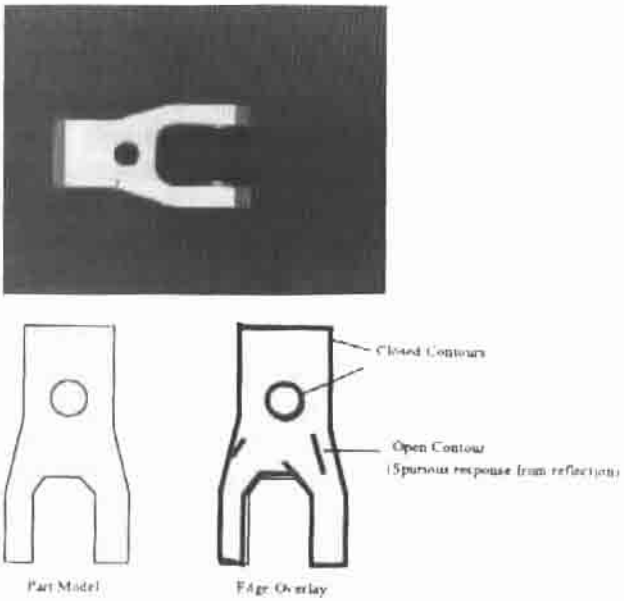

Fig. 3. A contour discovery example

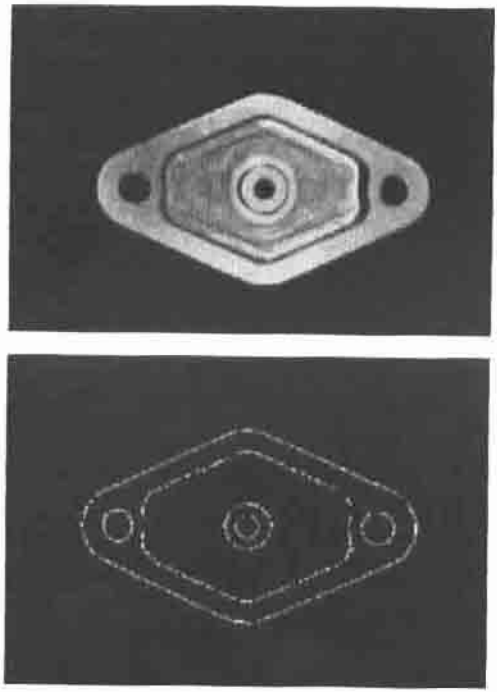

Fig. 4. An image and its contours

geometric features will be passed to other layers as contours.

Geometric features, such as edges, typically cause visual features, market by fluctuations in intensity. A zero-crossing edge detection algorithm, implemented in the IMLIB library [13], was used for finding these visual features.

A difficulty in determining geometry from intensity images is that they carry much more information than just the geometry of individual objects in a scene. Each object's surface properties, the properties of lights, reflections from other objects, etc. are all encoded in the intensity. We minimize many of these problems by carefully controlling the lighting conditions of the experiments. The static environment of the manufacturing workspace allows us to provided a diffuse light source at a chosen intensity.

The zero-crossing operator used calculates orientation and response magnitude at each pixel location in the image. Edges that exceed a minimum responce are located and then followed iteratively (using the orientation as a guide) until they close or drop below reponse and intensity thresholds. All branches are followed and the strongest retained. Edges that do not close or are below a minimum length are dropped. The remainder is a set of closed contours that are likely to be the signatures of geometric features in the intensity image. These are represented by a structure similar to a 'chain code'.

Another goal of the 2D layer is to determine the relationships between the contours. The relationships will be expressed in the form of strings and trees. These will later be used to solve a form of the correspondence problem in the 3D layer.

Consider a scene with two features, a part with an external boundary and a single hole. We would like to represent this scene with the string: '$F_1(F_2())$'. This can be interpreted as, a closed region within another closed region (see Figure 5).

To discover if feature F_2 is contained within F_1 given that we know F_1 is a closed feature, we select a point (x_2, y_2) on F_2 and another point (x_1, y_1) on F_1.

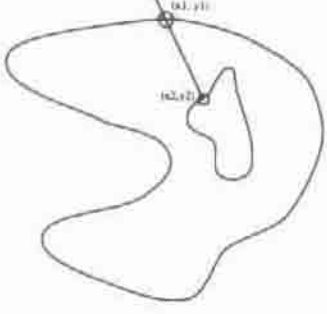

Fig. 5. A closed region within another.

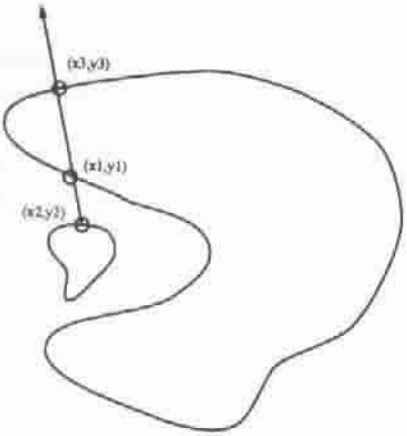

Fig. 6. A closed region outside another.

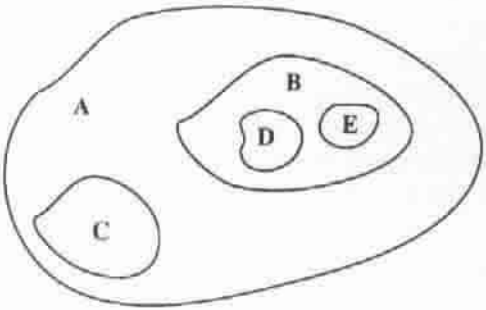

Fig. 7. A hierarchy example.

Now, we project the ray that begins at (x_2, y_2) and passes through (x_1, y_1). We count the number of times that this ray intersects with F_1. If this is odd then we can say F_2 is contained within F_1 otherwise is must lie outside of F_1 (see Figure 6).

This algorithm will hold true as long as the ray is not tangential at the point (x_1, y_1) of feature F_1. To avoid this case, we simply generate two rays that pass through (x_2, y_2) and a neighboring pixel on F_1. If either of these have an odd number of intersections then F_2 is contained in feature F_1.

An alternate method was also which employed region growing.

Using this algorithm, the relationships between the contours shown in Figure 7 would be determined as follows:

$B \in A$

$C \in A$

$D \in B$

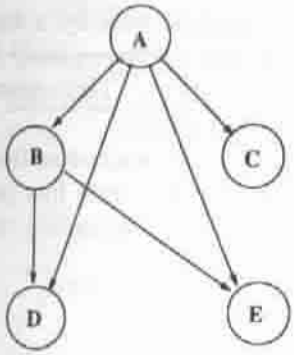

Fig. 8. The graph associated with the example.

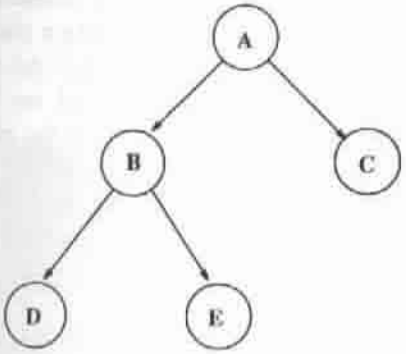

Fig. 9. The tree associated with the example.

$D \in A$

$E \in B$

$E \in A$.

Such relations can easily be represented by a graph as shown in Figure 8. A depth first search algorithm was implemented which converts such graphs to 'string' and tree representations. A tree representation for the above example is shown in Figure 9, and the 'string' representation is as follows:

A(B(D( ), E( )), C( )).

3.2. THREE-DIMENSIONAL IMAGE PROCESSING

The goal of this layer is to transform the contours found in the previous section from image coordinates into real world coordinates and add the third dimension. It operates on a pair of images passed through the 2D layer, taken from calibrated

locations that are like in orientation, and differ in position only by a known disparity in the image-X direction. It is assumed that the contours passed from the 2D layer correspond to geometric features in the object. The camera is calibrated using an implementation of Roger Tsai's method [19].

The camera is modelled as a pinhole camera (see Figure 10), with correction for radial distortion. The textbook method for computing depths from this model uses difference between feature locations in a pair of views according to the following formula:

$$Z = fD/(x_l - x_r),$$

where x_l and x_r are coordinates on the image plane, f is the focal length, and D is the disparity. To do so requires determining which visual features correspond with one another. This is a difficult problem which has been much researched.

We approach the correspondence problem by making the assumption that the contours found by the 2D layer are planar. This is reasonable since the parts we wish to sense are composed entirely of machined features, and we have placed constraints on the way these features can interact. By making the planar assumption, we can make use of a method developed by Aloimonos [2]. This method requires only that *contours* be correlated, rather than individual points. We use the relationship trees, found by the 2D layer, to correlate contours. The disparity of the centers of mass of corresponding regions is used as above to determine depth.

Assuming that contours are planar, the remaining task is to solve the following equation (of a plane) for p, q, and c:

$$Z = pX + qY + c.$$

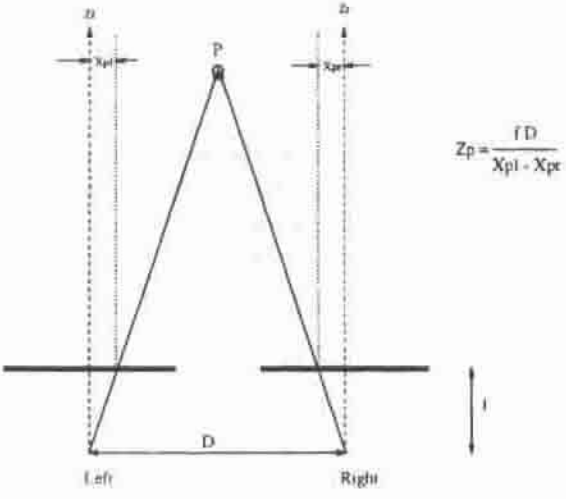

Fig. 10. Pinhole camera model.

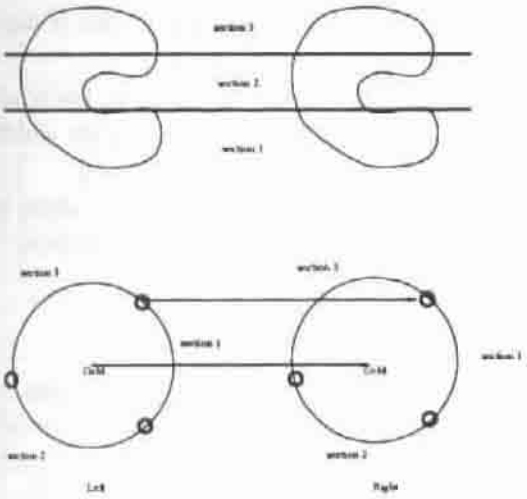

Fig. 11. Region splitting algorithms.

Fig. 12. Experimental setup.

Since this equation has three unknowns, it is necessary to split each region into three subregions in order to solve for them. Several region splitting algorithms were tested. The simplest, shown in the upper portion of Figure 11, splits contours into regions equal by vertical span. Symmetric features, such as holes, will produce collinear centers of mass when divided this way. Since three non-collinear points are needed to define a plane, another method was used. Contours were split on the basis of curve length, assuming that the 'first' point of one contour roughly corresponds with the closest point to its translation along a vector

equal to the disparity of the 'unsplit' contours' centers of mass. This is depicted in the lower portion of Figure 11.

This method was found to work fairly well for large outer regions, but produced poor results for small inner regions. This was solved by imposing the additional constraint that inner regions be in parallel planes to the outer region.

The experimental setup is shown in Figure 12. 3D information is added to the contour 'chain codes', which are passed on to the sensing/CAD interface.

4. Sensing to CAD Interface

The role of this module is to use the object properties extracted using vision to construct a CAD representation of the object. The α_1 modeller, developed at the University of Utah, was selected for its capability to represent machined features and interface with NC machining and inspection systems. The parts tested were originally designed using α_1 mechanical features, so it was known that they were representable in this modeller.

The set of features used for model construction included 'block' stock, holes, profile pockets, and profile outside curves. Profile curves, as defined in α_1, are series of curve and line segments that combine to produce closed curves (see Figure 13). For our research, we assume that curved segments are circular arcs. Profile pockets are allowed to contain profile 'islands', which are areas skipped when machining the pocket. Each of the features used can be described as a closed planar curve or curves and a depth to which the curve is extruded in a direction normal to its plane. Stock, outside curve, and pocket examples are shown in Figures 14, 15, and 16.

Each contour was fit with a profile curve or circle, using an algorithm based on curvature. Curvature is defined as the instantaneous rate of change of slope or orientation with respect to curve length. Orientation was retained from the edge detection algorithm mentioned above for each contour point. First and second derivatives of curvature were computed and smoothed. Areas of near-constant curvature were marked as part of a line segment, and areas where the curvature change was near-constant were marked as part of an arc segment. Segments of sufficient length were alternately fit with the appropriate type (arc or line) and 'grown' until the error in fit exceeded a tolerance. If an arc segment accounted for the entire curve length, it was considered to be a circle which might later be determined to be a hole. If not, a profile curve of connected arcs and lines was produced.

The relation tree was traversed beginning at its root, the part's outside curve. A bounding box was fit to the root contour, and used to produce the stock. The outside curve was produced from the root's profile. Holes, pockets, or islands were produced for each child node, depending on shape, relationships,

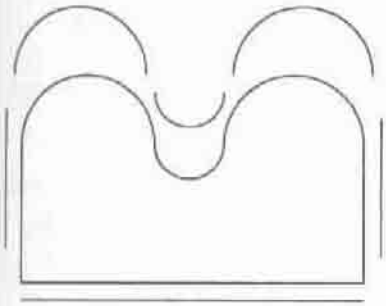

Fig. 13. Profile curve.

Fig. 14. Stock.

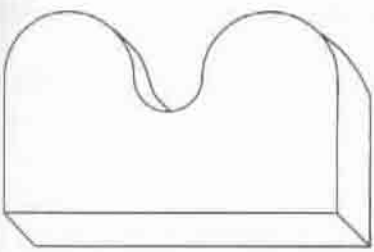

Fig. 15. Outside curve.

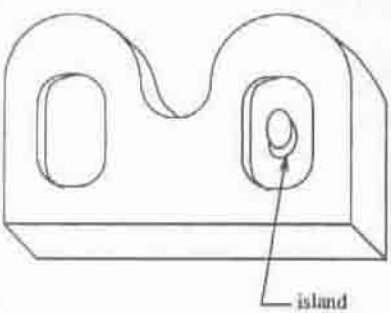

Fig. 16. Profile pocket.

and depths. Output was in the form of α_1 modelling commands, e.g.:

```
F1:= hole(
          Anchor,    %% Anchor, (3D location + orientation)
          Diameter,  %% hole diameter
          Depth,     %% hole depth
          Chamfer,   %% width of 45 degree chamfer (not used)
          Thru)      %% flag specifying if thru (T or nil)
```

4.1. RESULTS

The method described above produces a rough CAD model for the selected class of parts. We consider this to be an intermediate stage in the reverse engineering process, suitable to drive further sensing with more accurate devices such as a coordinate measuring machine (CMM). Although this is an intermediate stage, we considered it useful to apply the method and have a part machined.

The part selected was similar to the fuel pump cover from a Chevrolet engine. It is roughly diamond-shaped, with rounded corners. Three holes and a pocket with an island (a circular boss reinforcing the center hole) form the interior features. Using the techniques described above, an α_1 model was generated for the part. The reverse-engineered model was used to create a process plan and machine a part on the Monarch VMC-45 milling machine. The original part and reproduction can be seen in Figure 17.

The method was also applied to the front suspension shock linkage of the Utah Mini-Baja racer. Athough this has not been machined, the models can be seen in Figure 18 and rendered images in Figure 19. Note that this part does not conform to our specification that all features be parallel to a single plane, and thus produces somewhat simplistic results.

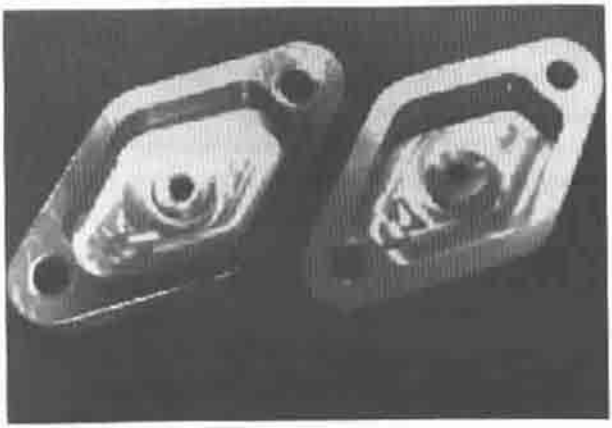

Fig. 17. Original and reproduction.

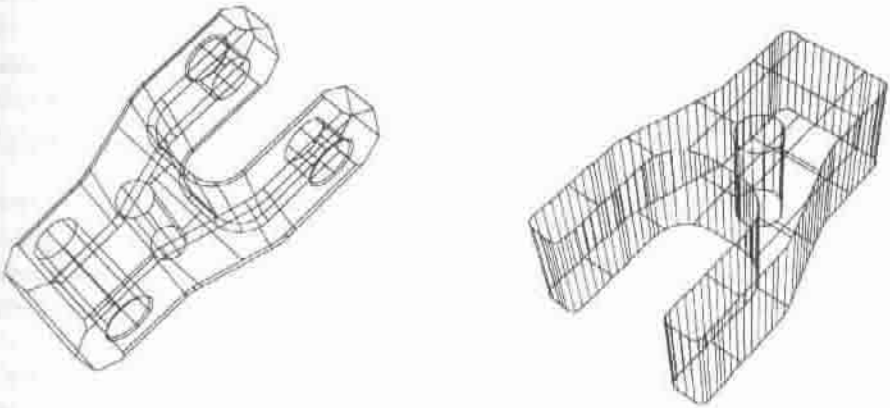

Fig. 18. Original and reverse-engineering part models.

Fig. 19. Original and reproduction.

5. New Strategy

The work described above was successful for a specific set of machined parts and utilized basic sensing tools. For further research, we wish to refine our tools and somewhat expand the set of parts which we can reconstruct. Below, we give a brief discussion of our new strategy.

In our new strategy, we would like to guide a sophisticated sensor (CMM) as *intelligently* as possible. Other efforts have required an extensive number of 'touches' to adequately explore a surface. We hope to determine the existence of as many features of a part as is possible using active vision and user interaction, and then obtain an accurate measure of their dimensions using a minimum number of touches. Sub-features not found through vision, but typical of the mechanical

features found may then be searched for through touch.

When a skilled machinist looks at a part, he or she quickly develops a *mental* machined feature representation of it. By letting our system interact with a skilled user early in the process, we anticipate that our system can use this high level representation to drive active vision and touch sensing.

Given our machined feature representation, feature interaction must be considered to avoid errors such as missed touches. For example, individual hole may be inspected by touching three equally spaced points in a plane perpendicular to the hole's axis to determine its center and diameter. However, if another hole intersects this one, portions of each hole's boundary will be nonexistent.

Feature interaction should also be considered for a thorough exploration. We propose to explore for certain aspects, or sub-features, that are common to specific mechanical features. By interpreting interacting features as separate entities with potentially different sub-features, we can be assured of obtaining a more complete representation. For example, two pockets may have different fillet radii. If they interact, they may be considered to be one pocket with one fillet radius, an invalid assumption.

By interpreting visual features as mechanical features, we anticipate that fine features which might normally go unnoticed can be captured. For example, if a hole is detected, our system can use the knowledge that holes are often counter-bored or counter-sunk to refine the data in that area. Fillets, which often occur at junctures of planes, are another mechanical feature which might easily be missed in visual processing. Ideally, this system could also be used to capture a description of very fine and often occluded features such as threads.

The distinctions of this proposed system include:

- Representation as mechanical features rather than purely mathematical forms to reduce touch sensing requirements.
- Interacting with a skilled human.
- Utilizing feature interactions to optimize touch sensing performance.
- Combining vision with touch for *accurate* and *complete* descriptions.
- End result is a representation that can be machined semi-automatically.

5.1. PROBLEM DOMAIN

As before, we will limit our efforts to the inspection of machined parts. To make the feature interaction problem tractable and to take advantage of an existing CMM interface [20], we will consider parts that contain holes, slots, bosses, profile sides, and profile pockets as defined within the α_1 modelling system. Each of these features can be described by a two-dimensional curve, depth, and a three dimensional frame (location+orientation). In our previous work, we

assumed that the object to be sensed was composed *only* of these features, and further that the frames were parallel. In our next phase, we will identify these reduced dimension features in arbitrary frames, as part of complex objects.

6. Conclusions

We have developed a reverse engineering system that construct machine feature-based CAD models from sensed information. We have machined a part using one of these models, and compared it to the original. We have conducted experiments using our system and background research which have given us insights into the reverse engineering problem. Using these insights, we have formed a strategy that should perform with robustness in a complex problem domain.

The goal of reconstruction techniques as used by some object recognition systems is to obtain enough information to distinguish an object from others in a set. Thus, effort is made to reduce descriptions to the minimum required in order to reduce matching time. Some efforts have been made to obtain accurate geometric descriptions of surfaces by combining vision with touch, but these attempt to explore the surfaces using *geometric* information. We will attempt to obtain descriptions of mechanical parts, recognize their *mechanical* features, and develop an accurate and complete description in minimum time.

References

1. Allen, P.: Sensing and describing 3D structure, in *Proc. IEEE Int. Conf. Robotics and Automation*, Institute of Electrical and Electronics Engineers, 1986, pp. 126–131.
2. Aloimonos, J. and Shulman, D.: *Integration of Visual Modules An Extension of the Marr Paradigm*, Academic Press, New York, NY, 1989.
3. Arman, F. and Aggarwal, J.: Model-based object recognition in dense range images – A review, *ACM Computing Surveys* **25**(1) (1993), 5–43.
4. Besl, P. and Jain, R.: Three-dimensional object recognition, *ACM Computing Surveys* **17**(1) (1985), 75–145.
5. Chen, Y. and Medioni, G.: Object modelling by registration of multiple range image, *Int. J. Image and Vision Computing* **10**(3) (1992), 145–155.
6. Chen, Y. and Medioni, G.: Integrating multiple range images using triangulation, in *Image Understanding Workshop*, Defence Advanced Research Projects Agency, Software and Intelligent Systems Office, 1993, pp. 951–958.
7. Chen, Y. and Medioni, G.: Fitting a surface to 3D points using an inflating balloon model, 1994.
8. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W.: Surface reconstruction from unorganized points, *Computer Graphics, SIGGRAPH'92* **26** (1992).
9. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W.: Mesh optimization, *Computer Graphics, SIGGRAPH'93* **27** (1993).
10. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W.: Piecewise smooth surface reconstruction, *Computer Graphics, SIGGRAPH'94* (1994), to be published.
11. Hsieh, Y.: Reconstruction of Sculptured Surfaces Using Coordinate Measuring Machines, Master's thesis, University of Utah, 1993.

12. IMLIB staff. Imlib documentation: Image Processing User Commands, University of Utah, 1991.
13. IMLIB staff. zcedge (unix man page): Image Processing User Commands, University of Utah, 1991.
14. Karinthi, R. and Nau, D.: An algebraic approach to feature interactions, *IEEE Trans. Pattern Analysis and Machine Intelligence* **14**(4) (1992), 469–484.
15. Marefat, M., Malhotra S., and Kashyap, R.: Object-oriented intelligent computer-integrated design, process planning, and inspection, *IEEE Computer* (1993), 54–65.
16. Motavalli, S. and Bidanda, B.: A part image reconstruction system for reverse engineering of design modifications, *J. Manufacturing Syst.* **10**(5) (1991), 383–395.
17. Sarkar, B. and Menq, C.: Smooth-surface approximation and reverse engineering, *Computer Aided Design* **23**(9) (1991), 623–628.
18. Stansfield, S.: Visually-aided tactile exploration, in *Proc. IEEE Int. Conf. Robotics and Automation*, Inst. Electrical and Electronics Engineers, 1987, pp. 1487–1492.
19. Tsai, R.: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses, in G.H.L. Wolff and S. Shafer (eds), *Radiometry – (Physics-Based Vision)*, Jones and Bartlett, 1992.
20. Van Thiel, M.: Feature Based Automated Part Inspection, Master's thesis, University of Utah, 1993.