COMPUTER VISION AND IMAGE UNDERSTANDING
Vol. 61, No. 3, May, pp. 468–474, 1995

Industrial Inspection and Reverse Engineering

TAREK M. SOBH, J. OWEN, C. JAYNES, M. DEKHIL, AND T. C. HENDERSON

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

Received May 15, 1994; accepted January 11, 1995

We propose a new design for inspection and reverse engineering environments. We have designed and experimented with such an environment for capturing sense data of mechanical parts in an intelligent way. We construct a sensing → CAD interface for the automatic reconstruction of parts from visual data. We discuss the use of the dynamic recursive context for finite state machines (DRFSM) as a new discrete event dynamic system (DEDS) tool for controlling inspection and exploration. We also implement a graphical interface for designing DRFSM DEDS controllers.

1. INTRODUCTION

Even as CAD/CAM allows us to design and manufacture increasingly complex objects with tighter tolerances, it can be used to help ensure that those tolerances are met. This is the inspection problem and entails the use of highly accurate sensing equipment in combination with a design specification, typically a CAD model. The reverse engineering problem has similar goals, but the CAD model is not available.

The objective of this research project is to explore the basis for a flexible software and hardware environment for a variety of inspection and reverse engineering activities. The emphasis of our research was on devising a framework for control of sensing systems for inspection and reverse engineering. As such, we felt it was appropriate to use low-cost, easy-to-implement vision sensing and simulated touch sensing. In this context, we will concentrate on the adaptive extraction of world properties and on producing reliable descriptions of sensed objects for manufacturing and/or refinement purposes. We use an observer agent with some sensing capabilities to actively gather measurements of mechanical parts.

In our research [13, 15–18], we have concentrated on a subset of mechanical parts, namely machined or milled parts (as opposed to castings, extrusions, etc.). By selecting this subset, we have been able to formulate assumptions which simplify our sensing methods. In addition, we are able to assume that the sensing environment will be controllable. This paper will give an overview of our research.

2. METHODOLOGY

A goal for this project was to design a system which allows us to combine vision, which can be relatively quick, and touch, which is typically slow but gives accurate data [7, 11, 19]. Vision would be used not only to sense the object, but also to observe the touch sensor, which must be maneuvered to contact the part, running the risk of breakage. This combination requires control of disparate systems.

Our experiments have focused on the environment and control mechanisms, rather than the sensing. Experiments were performed using a black-and-white CCD camera mounted on a tripod or robot arm and a simulated CMM touch probe.

2.1. *Intelligent Control*

Discrete event dynamic systems (DEDS) are aptly suited for modeling robotic observers as they provide a means for tracking the *continuous, discrete,* and *symbolic* aspects of the scene under consideration [5, 8, 9]. In our system, the DEDS controller is able to *model* and *report* the state of the inspection process.

In inspection, the DEDS guides the sensors to the parts of the objects where discrepancies occur between the reference model and the recovered structure. The DEDS formulation can also compensate for noise in sensor readings [14]. Recovered data from the sensing module is then used to drive the CAD module. The DEDS sensing agent is thus used to collect data of a *passive* element for designing *structures*.

A *dynamic-recursive* implementation of a DEDS algorithm is used to facilitate the state recovery of the inspection process. The dynamic recursive context for finite state machines (DRFSM) is a particularly apt choice for the exploration of a set of machined parts which exhibit a recursive nature. A feature which is contained within another feature will generally be of a smaller scale than its parent, even though it might be of a similar type and require the same sensing technique. With the DRFSM implementation, the smaller scale feature can be explored using the same technique, but with more appropriate parameters [13, 15–18].

1077-3142/95 $6.00

The mechanical parts we are sensing have no interaction between the boundaries of their machined features and thus may be considered to be inside or outside one another. Although by relying on this assumption we are placing a significant constraint on the class of parts which we can inspect and reverse engineer, this class still represents a large number of real-world machined parts. Given that the focus of our project is on the system's control environment, we feel that this is acceptable.

Within our system, the recursive nature of these parts is used to keep track of features and their relationships to each other. Each feature is assigned a symbol. If a feature A contains another feature B, we store that relationship in a "part string" as $A(B())$.

2.2. *Sensing for Intelligent Control*

The vision system provides the controlling agent with specific information about the current state of the inspection. The aspects of the scene that are needed for this type of control are:

1. Number of features,
2. Contour representation of each feature,
3. Relationships of the features,
4. Location of the touch probe with respect to a part's features.

The above items are accomplished using standard 2-D image processing algorithms. By assuming control of the environment, we can select the background color and texture, the lighting conditions, and some of the physical touch probe's visual characteristics to make the part distinctive in a scene. After doing so, we can use simple thresholding to locate the part and probe in an image. The part is assumed to be present in the scene and has no intersection with the image's border. The probe is assumed to intersect the border and has a distinctly different intensity than the part.

Edge detection, via a zero-crossing algorithm, is used to detect the visual discontinuities of a part. An edge response may be considered to be a contour if it satisfies two conditions: (1) each response must exceed a previously specified minimum value and (2) the length of each edge must exceed a previously specified minimum pixel count.

Edges are followed iteratively. An edge is followed until its response falls below the minimum or we arrive at our starting position, in which case the contour is known to be closed. If a branch in the contour is encountered, the branch location is saved and following continues. We attempt to follow all branches looking for a closed contour. Branches are considered to be part of a contour because they may represent an actual feature of the part and should be inspected. These contours are stored as "chains."

Contours which form closed curves are considered to be portions of a mechanical feature's boundary (see assumptions below). Those which form open curves may be attributed to occlusion of one geometric feature by another or the probe or lighting problems such as shadows or reflections. We attempt to avoid occlusions by selecting views appropriately and lighting problems by providing a diffuse light source and a part with a dull finish.

2-D vision is also used to determine the relationship between these contours. This relationship is determined by region growing. This allows us to construct the aforementioned part string.

Once all this information has been determined, computing the proximity of the probe tip to a feature allows us to determine the state of the inspection. Example states include:

- probe far
- probe close
- probe on feature.

These labels are expressed in terms of the current feature being sensed, which corresponds to a level of recursion within the controlling DRFSM. Transitioning from a "probe far" state to a "probe on feature" state corresponds to an error condition, halting the machine to prevent damage to the probe.

3. MODEL BUILDING FOR REVERSE ENGINEERING

In order for our system to be used for reverse engineering as well as inspection, it must be capable of building an initial model of a part. We have achieved this functionality by building on top of the algorithms described in the previous section. Additionally, rather than building a model which represents only geometry, we have created a method that produces a feature-based representation.

3.1. *Modeler*

A set of machinable features, as implemented in the α_1 modeling system [3], was selected for our representation. The α_1 system was developed at the University of Utah and allows a user to design using mechanical features and geometric construction techniques, render images using several techniques, interface with NC manufacturing equipment, drive the inspection of parts, etc. The selected feature set includes stock, profileSides, profilePockets, and holes. Holes are defined by location, diameter, depth, etc. Profile features, as implemented in α_1, are defined by profile curves and other parameters, such as depth and fillet radii. α_1 profile curves are composed of arcs, lines, and splines, linked end to end. We have limited profile curves in our feature set to arcs and lines for simplicity.

3.2. *2-D Contours → 3-D Contours*

In order to transform our image coordinate 2-D contours into world coordinates, we implemented a stereo vision algorithm by Aloimonos and Shulman [2]. This algorithm can compute the depth of a planar contour without requiring correspondence between individual elements in a pair of images. Correspondence is instead required between contours. This is readily obtainable when the relationships are known, as in our case. Essentially, it uses the disparity of the centers of mass of a corresponding pair of closed contours.

If the orientation of the contour's plane is known, disparity of the centers of mass is sufficient. If not, it becomes necessary to split the contours' outlines into three roughly corresponding parts. This gives us three equations of the form

$$Z_i = pX_i + qY_i + c, \tag{1}$$

where p, q, and c are the parameters of a contour's plane.

The technique used to split contours was to divide each of the pair into thirds. Splitting for one image could be started at any location, but the starting location for the other should roughly correspond to it. This was done by translating the first contour's starting point by the disparity of the contours' centers of mass.

The technique described above produces known 3-D coordinates only at locations marked by discontinuities in the surface geometry. Although the relationship between these contours can be used to infer something about the intervening surface, not enough information is present to produce an accurate model. To specify depths for these smooth intervening areas, we introduced hand-collected information, similar to what might be gathered by a CMM.

3.3. *Contours to Arcs and Lines*

Each contour is converted to a hole or profile curve using the curvature along the contour. Areas in which the curvature is zero (or below a small threshold) for a specified distance along the contour are considered to be line segments. Areas in which the curvature is constant for a specified distance are considered to be arcs. Curvature, k, at a point on a curve is defined to be the instantaneous rate of change of the curve's slope, ϕ, with respect to curve length, s [12]:

$$k(s) = d\phi(s)/ds,\ ds = \sqrt{dx^2 + dy^2}. \tag{2}$$

Slope is taken to be the orientation of the gradient of the difference of Gaussians (DOG) function. As previously mentioned, the DOG function is an approximation to the Laplacian. The derivatives of slope are computed using a forward difference technique, and the results are smoothed. A graph of curvature vs distance along a curve can be seen in Fig. 1.

For each arc segment, a circle is fit using a least-squares fit, and the endpoints of the segment are grown until the distance from the contour to the fitted circle exceeds a tolerance. Our implementation of the fitting procedure [10] could be extended to include general conic sections by changing the generators used from $1, x, y, x^2 + y^2$ to $1, x, y, x^2, xy, y^2$. The fitting/growing process is repeated until growing has no effect or another segment is reached. A similar method is used for the line segments.

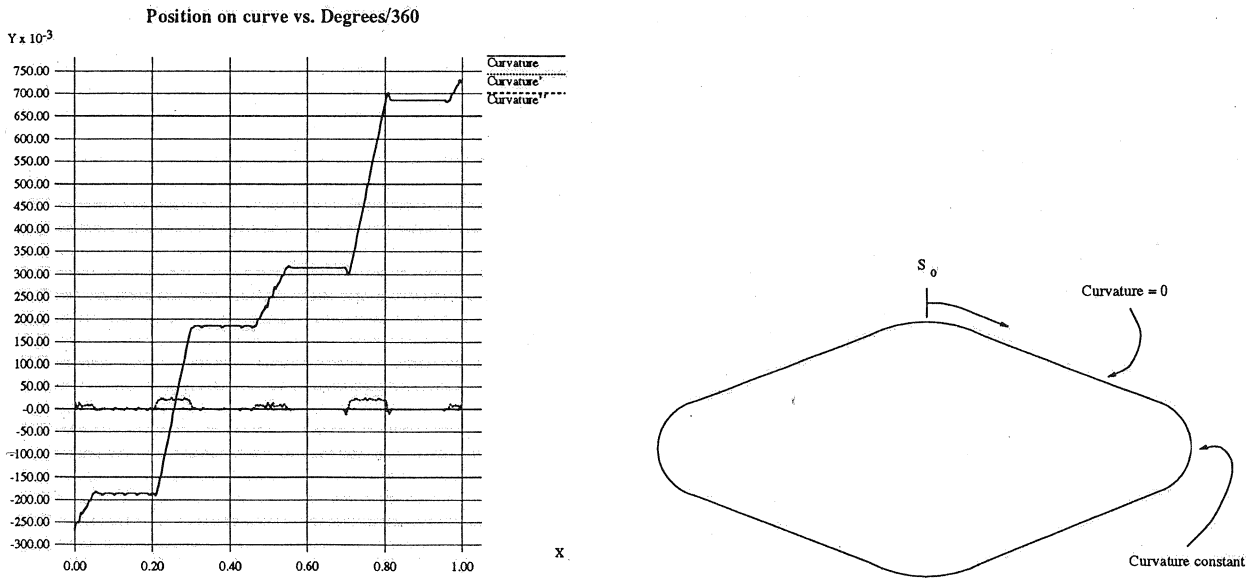


FIG. 1. Curvature and first and second derivatives.

3.4. *Arcs and Lines to Machined Features*

In our limited domain of machined features, machining constraints can be exploited to identify curves as machined features. For instance, if a negative feature contains a positive feature, then it must be an island within a profilePocket. If necessary, the island may be trimmed to the sensed height with a negative feature such as the profileGroove. If a negative feature contains no positive feature and is composed of a 360° arc segment, then it may be represented by a hole. If a hole contains no other features, and the interior of the raw image is below a threshold, it may be considered to be through.

Some aspects of machined features are difficult to measure accurately using our current image processing algorithms. For example, fillets between line segments in a profilePocket may not be within the accuracy of our vision, but are necessary for machinability. In cases such as these, default values are selected so as to have minimal deviation from the reverse engineered model, yet allow the model to be machined. It is anticipated that by recognizing likely locations for such difficult aspects, they may be explored effectively using a CMM (coordinate measuring machine) or another probing device.

4. EXPERIMENTS

Three experiments were conducted. The purpose of the first experiment was to demonstrate the functionality of the DRFSM controller and to test the effectiveness of the 2-D sensing module at determining the state of a sensing run. The second tested a new method for DRFSM design. The third experiment tested the 3-D sensing module and the sensing → CAD interface.

4.1. *Control Process Experiments*

This experiment was performed to integrate the visual system with the state machine. An appropriate DRFSM was generated by observing the part and generating the feature information. A mechanical part was put on a black velvet background to simplify the vision algorithms. The camera was placed on a stationary tripod such that the part was always in view. A simulated CMM probe was moved in response to the textual output of the controlling machine.

Once the first level of the DRFSM was created, the experiment proceeded as follows: First, an image was captured from the camera. Next, the image processing modules found the position of the part, the number of features observed, the recursive string, and the location of the probe. A program using this information produced a state signal appropriate for the scene. The signal was read by the state machine and the next state was determined. Each closed feature was treated as a recursive problem, and as the probe entered a closed region, a new level of the DRFSM was generated with a new transition vector.

The specific dynamic recursive DEDS automaton generated for the test was a state machine G (shown in Fig. 2). Where the set of states X = {Initial, EOF, Error, A, B, C, D} and the set of transitional events Σ = {1, 2, 3, 4, 5, 6, 7, 8, 9, eof}. Here, states A through D correspond to:

- "no probe"—the probe is not within the scene
- "probe far"—the probe is in the scene, but not near the feature
- "probe close"—the probe is near the feature (i.e., move with caution)
- "probe on feature"—contact is pending.

The transitional events correspond to tests such as "distance from probe to feature is less than 2 in.," where the distance is variable, depending on the depth of recursion. This adds robustness to the system, allowing ranges to be specified as opposed to exact thresholds.

State transitions were controlled by the input signals supplied by intermediate vision programs. As described above, there are four stable states (A, B, C, and D) that describe the state of the probe and part in the scene. Three other states, Initial, Error, and EOF specified the state of the system in special cases.

In one sequence, the probe was introduced into the scene and moved in a legal way (accepted by stable states in the machine) until contact was made. Next, the probe backed off and again approached until the probe and part were in proximity. The automaton was forced into an error state by approaching too quickly. The probe was not seen until it was close to the object body. Because a transition from state A to C is invalid, an error state is reached. Images recognized during the experiment as

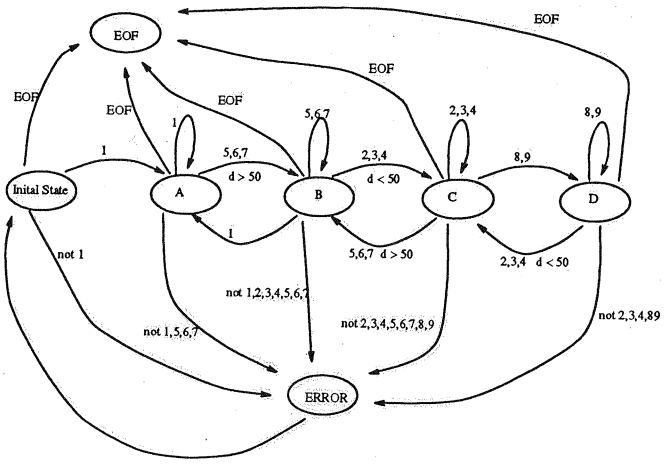


FIG. 2. State machine used in test.

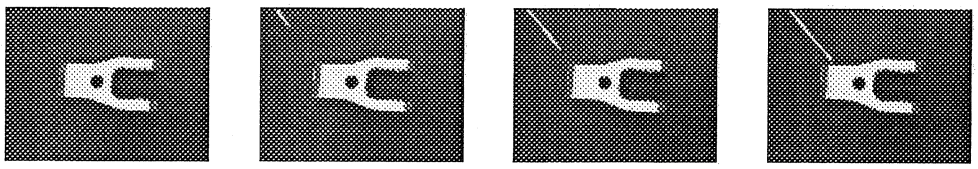


FIG. 3. States during the exploration of the outside curve.

these states are shown in Fig. 3. As shown in that figure, the part used was a simple one with only one hole. Its part string was recovered to be the following:

Outside(Hole()).

In another sequence, the part was more complex. Exploration of a hole in this part is shown in Fig. 4. Its part string was recovered to be the following:

Outside(Hole(),Pocket(Hole()), Hole()).

As before, the probe was introduced into the scene and moved toward the part. Next, the probe backed off and again approached until the probe and the part were in proximity. The automaton was forced into an error state by the sudden disappearance of the probe after it was very close to the part. Because a transition from state C to state A is invalid, an error state is reported.

Since touch sensing in this context requires contact of a precision instrument with a part, it is highly desirable to use a robust control mechanism. For both parts, the operation of camera and probe and detection of error conditions demonstrated the effectiveness of DRFSM in controlling disparate types of systems to achieve safe operation.

A second experiment was run, using a DRFSM generated by GIJoe. GIJoe is an interactive graphical tool for building FSMs which was modified to build DRFSMs [6]. The output of GIJoe was compiled and linked directly to the experimental sensing code used in the first experiment.

The controlling DEDS machine performed as expected.

4.2. *Model-Building Experiment*

This experiment was developed to test the system's model-building capabilities, necessary for reverse engineering. Setup was similar to the first experiment, but with the camera mounted on a robot arm to allow constrained motion between views.

The experiment started by taking images for the object from two positions, defined by the operator. These were used to generate two sets of contours which were fed into the stereo module for depth estimation. Using the stereo module, an initial set of world coordinates for these contours was generated and converted to an α_1 model.

A block diagram for the experiment is shown in Fig. 5. The resulting model for one of the two machined parts used is shown in Fig. 6.

5. SUMMARY

We propose a new strategy for inspection and/or reverse engineering. We use a recursive DEDS agent with some sensing capabilities to control the inspection and reverse engineering of mechanical parts. Geometric descriptions of the objects under analysis are generated and expressed in terms of a computer-aided design system. The geometric design can then be used to construct a prototype of the object. The manufactured prototypes are then to be inspected and compared with the original object using the sensing interface and refinements made as necessary.

We also describe a framework on which a full inspection and reverse engineering environment could be constructed. In it, the problem is divided into *sensing, design,* and *manufacturing* components with an underlying software and hardware backbone. The developed framework utilizes existing knowledge to formulate an adaptive and goal-directed strategy for exploring mechanical parts.

We consider the following tasks to constitute completed work in this project:

- Designed the DRFSM DEDS framework for recursive inspection.

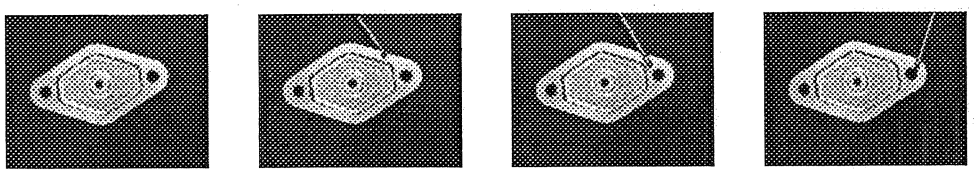


FIG. 4. States during the exploration of the rightmost hole.

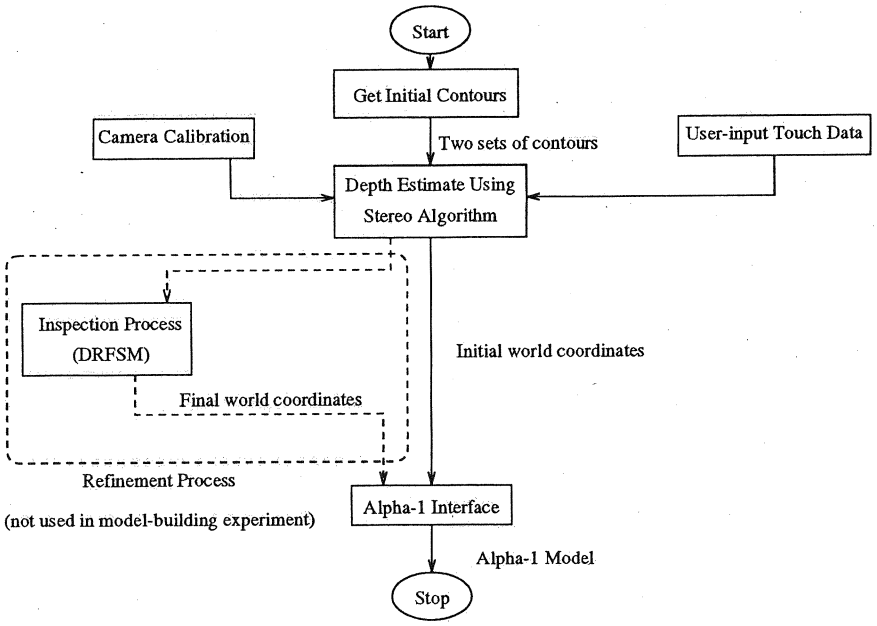


FIG. 5. Block diagram for the experiment.

- Implemented image processing modules for recognizing features and probe position with respect to parts composed of them,
- Designed and implemented visual structure recovery techniques for machine parts and implemented calibration routines,
- Designed and implemented a sensing → CAD interface for generating α_1 code for bodies from depth, contour, and recursive feature relationships,
- Implemented the DRFSM DEDS automata for recursive inspection using a robot-held camera, probe, and actual parts.
- Designed and implemented a modification to an existing reactive behavior design tool (GIJoe) to accommodate the graphical design of DRFSM DEDS.

The application environment we eventually intend to develop consists of three major working elements: the sensing, design, and manufacturing modules. The ultimate goal is to establish a computational framework that is capable of deriving designs for machine parts or objects, inspecting and refining them, while creating a flexible and consistent engineering environment that is extensible. The control flow is from the sensing module to the design module and then to the manufacturing component. Feedback can be supplied to the sensing agent to inspect manufactured parts, compare them to the originals, and continue the flow in the loop until a certain tolerance is met. The system is ultimately intended to be as autonomous as possible.

6. CONCLUSIONS AND FUTURE WORK

The current implementation has allowed us to explore the constraints needed to make progress in the reverse engineering problem. We have come to the conclusion that a machine feature representation is a worthwhile target and may be useful in determining sensing strategy. Interacting features have been identified as a source of

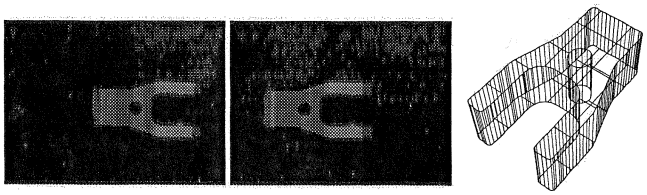
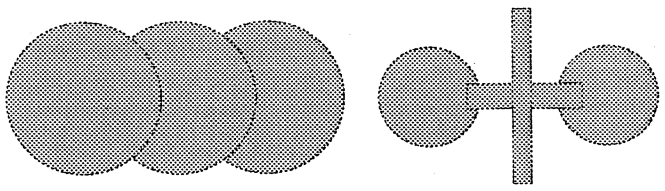


FIG. 6. Stereo image pair and the resulting α_1 model from the experiment.



Three Holes? profilePocket? Two Slots + Two Holes? profilePocket?

FIG. 7. Possible combinations.

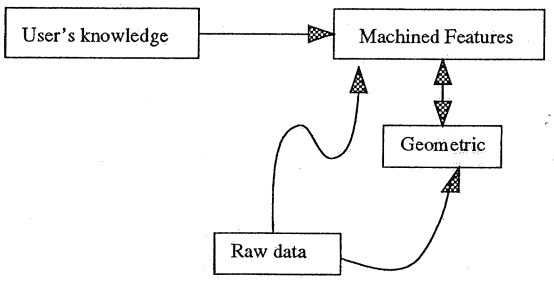


FIG. 8. Proposed, feature-first approach.

complexity. This information will be used to construct a new reverse engineering system.

In our current implementation, each contour is treated as at most one machined feature (some contours may be islands and therefore part of another contour's feature). Future work may allow contours to be made from multiple features if appropriate. For example, combinations of drilled holes, slots, and pockets may be produced (see Fig. 7), based on a machining/inspection time/cost analysis which will include such factors as time needed to select and load tools (operator), change tools, etc. This problem has some characteristics that may be best solved through artificial intelligence or optimization techniques.

However, for the reverse engineering problem, autonomy may not be an important requirement. By allowing interaction with a machinist, we could overcome the problems of features interacting with each other and more complex geometries. A conceptual graph for such a system is shown in Fig. 8. Some support for this type of system is found in [1, 4].

Such system could provide valuable information about how to build reverse engineering systems in the future. For example, it could be used to perform protocol analysis to determine how an expert machinist solves the reverse engineering problem. Lessons learned would enable us to build a robust autonomous system.

ACKNOWLEDGMENTS

This work was supported in part by ARPA under ARO Grant 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.

REFERENCES

1. R. Abella, J. Daschbach, and L. Pawlicki, Human skill interface in reverse engineering, Proceedings of the *13th Annual Conference on Computers and Industrial Engineering, 1991,* Vol. 21, pp. 495–499, University of Toledo.
2. J. Aloimonos and D. Shulman, *Integration of Visual Modules: An Extension of the Marr Paradigm,* Academic Press, New York, 1989.
3. α_1. α_1 user's manual, Computer Science Department, University of Utah, 1993.
4. B. Batchelor, D. Hill, and D. Hodgson, Eds., *Interactive Image Processing for Problem Evaluation,* pp. 439–458, North-Holland, Amsterdam, 1984.
5. A. Benveniste and P. L. Guernic, Hybrid dynamical systems theory and the signal language, *IEEE Trans. Autom. Control* **35**(5), 1990.
6. M. J. Bradakis, Reactive behavior design tool, Master's thesis, Computer Science Department, University of Utah, Jan. 1992.
7. C. Menq, H. Yau, and G. Lai, Automated precision measurement of surface profile in CAD-directed inspection, *IEEE Trans. Robot. Autom.* **8**(2), 1985, 268–278.
8. A. Nerode and J. B. Remmel, A model for hybrid systems, in *Proceedings Hybrid Systems Workshop, Mathematical Sciences Institute, May 1991,* Cornell University.
9. C. M. Özveren, Analysis and control of discrete event dynamic systems: A state space approach, Ph.D. thesis, Massachusetts Institute of Technology, Aug. 1989.
10. V. Pratt, Direct least-squares fitting of algebraic surfaces, in *SIGGRAPH '87,* pp. 145–152.
11. B. Sarkar and C. Menq, Smooth-surface approximation and reverse engineering, *Comput. Aided Design* **23**(9), 623–628.
12. R. J. Schalkoff, *Digital Image Processing and Computer Vision,* Wiley, New York, 1989.
13. T. Sobh, J. Owen, C. Jaynes, M. Dekhil, and T. Henderson, Intermediate results in active inspection and reverse engineering, Computer Science Department UUCS-93-014, University of Utah, Salt Lake City, June 1993.
14. T. M. Sobh and R. Bajcsy, A model for observing a moving agent, in *Proceedings of the Fourth International Workshop on Intelligent Robots and Systems (IROS '91), Osaka, Japan, Nov. 1991.*
15. T. M. Sobh, M. Dekhil, C. Jaynes, and T. C. Henderson, A dynamic recursive structure for intelligent exploration, Technical Report UUCS-92-036, Department of Computer Science, University of Utah, Oct. 1992.
16. T. M. Sobh, M. Dekhil, and J. C. Owen, Discrete event control for inspection and reverse engineering, in *IEEE International Conference on Robotics and Automation, San Diego,* May 1994.
17. T. M. Sobh, J. C. Owen, M. Dekhil, C. Jaynes, and T. Henderson, Industrial inspection and reverse engineering, in *1993 IEEE 2nd CAD-Based Vision Workshop, Pittsburgh, Feb. 1994.*
18. T. M. Sobh, J. C. Owen, C. Jaynes, M. Dekhil, and T. C. Henderson, Active inspection and reverse engineering, Technical Report UUCS-93-007, Department of Computer Science, University of Utah, Mar. 1993.
19. G. Stix, Computed tomography is a boon to reverse engineering, *Sci. Am.* **268** Jan. 1993, 147.