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

Abstract

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 $\longrightarrow$ CAD interface for the automatic reconstruction of parts from visual data. We briefly discuss the use of the dynamic recursive finite state machine (DRFSM) as a new discrete event dynamic system (DEDS) tool for controlling inpection 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 aid us in ensuring that those tolerances are met. This is the inspection problem, and entails the use of highly accurate (and most likely, expensive) 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 consistent software and hardware environment, and a flexible system that is capable of performing a variety of inspection and reverse engineering activities. In particular, we will concentrate on the adaptive automatic extraction of some properties of the world to be sensed and on the subsequent use of the sensed data for producing reliable descriptions of the sensed environments for manufacturing and/or description refinement purposes. We use an observer agent with some sensing capabilities (vision and touch) to actively gather data (measurements) of mechanical parts.

In our research, 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 about the objects which we are sensing which make our sensing methods more efficient. In addition, we are able to assume that sensing environment will be controllable.

This paper will give an overview of our research. Detailed technical information can be found in related technical reports (see [20, 21, 22]).

2 Sensing Strategy

The vision system provides the controlling agent with specific information about the current state of the inspection. This is done though several layers of vision processing and through the cooperation of 2D, $2\frac{1}{2}$D, and 3D vision processors.

The aspects of the image that need to be given to the state machine are:

1. Location of the probe with respect to the part.
2. Number of features.
3. Contour representation of each feature.
4. Relationships of the features.
5. Depth of features.
6. Approximate depth estimates between features.

Items 1 through 4 above 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 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 have no intersection with the image's border. The probe is assumed to intersect the border and have a distinctly different intensity than the part.

Edge detection, via a zero-crossing algorithm, is used to detect the visual discontinuities of a part. Edge responses which are sufficiently strong and of sufficient length are considered to be contours. These contours are stored as "chains", derived by edge following routines.

0-8186-5310-8/94 $03.00

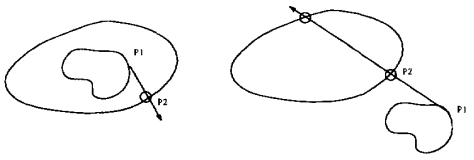

Figure 1: 2-d ray casting to determine feature relationships.

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 other factors such as lighting problems (shadows or reflections). We attempt to avoid occlusions by selecting views appropriately and lighting problems by providing a diffuse light source and a well buffed part.

2-d vision is also used to determine the relationship between these contours. This relationship is determined in either of two ways: by region growing or by casting two-dimensional rays. The first method should be self-explanatory. The second method works by selecting points (P_1 and P_2) on two contours, A and B (see Figure 1). To test if A is within B, a ray is cast from P_1 to P_2, counting the number of intersections with B. If the number is odd, then A is within B, otherwise it is not. This method works as long as the ray is not tangential to B, a condition that can be checked and avoided.

Once the part's contours and their relationships have been determined, and the probe located in the scene, the proximity (in image coordinates) 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. Although the sensing discussed thus far utilizes only two dimensional vision, our experiment (discussed below) demonstrated its ability to control a touch probe.

In order to transform our 2-d contours into three dimensions, we implemented a stereo vision algorithm by Aloimonos[2]. This algorithm can compute the depth of a planar contour without requiring correspondence between individual elements in a pair of images. Correspondence is 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.

While edges as detected in our images are not planar in the general case, they will be if:

- the stock from which the feature was milled is composed of planar faces
- interaction between features is limited so that feature outlines do not intersect[1]
- feature boundaries intersect at most one planar face of the stock

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, giving three equations and three unknowns. 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.

Camera calibration was performed using a technique developed by Tsai[25, 28]. The camera model used is based on the pinhole model of 3D-2D perspective projection with first order radial lens distortion. Tests on a simple cube produced an average error of approximately 2.3%.

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 attempt to determine depths for these smooth intervening areas, we experimented with an "illumination table". The basic idea of this that the intensity at a point decreases as the distance to that point or from a light source to that point increase. To be useful for our application, the point should be on a diffuse surface with no shadows, etc. If the intensity change is significant, it could provide a rough gauge for comparing the depths of two surfaces or if a table was built from known values it could be used for a quick table look-up. The results were somewhat mixed and other methods are being investigated such as depth from shading and depth from focus.

3 Sensing $\longrightarrow$ CAD Interface

An important step in the reverse engineering process is the accurate description of the real-world object. We

[1] This condition also allows us to express contour relationships in a tree fashion. e.g. "Feature A is a child of Feature B".

generate an α_1 model from a combination of three types of scene information.

- Two dimensional images and feature contours.
- Stereo vision depth information.
- Touch information from the CMM (still to be implemented.)

By using each sensing method, we are able to gather enough data to construct the accurate CAD model necessary for reverse engineering. The two dimensional images provide feature detection that is in turn used by the stereo system to build feature models. Finally, the CMM drastically reduces uncertainty through the exploration of the features.

The state machine and the sensors are able to produce a set of data points and the respective enclosure relationships. Each feature is constructed in α_1 independently and the final model is a combination of these features. This combination is performed using the recursive structure of the object by forming the corresponding string for that object and generating the code by parsing this string recursively. The third dimension is retrieved from the stereo information and the illumination table as described before. An example for a reconstructed part is shown in figure 2.

This interface is one of the most important modules in this work, since it is the real link between inspection and reverse engineering. We have chosen α_1 as the CAD language since it has very powerful features, in addition to the fact that it has interfaces with some manufacturing machines, which allows us to actually regenerate a hard copy of the part. This will be our next step, so that, the output of our next experiment will be another part, hopefully identical to the original part.

3.1 Contours to Splines

Both closed and open contours are represented as ordered sets of points. The contour points are used as control points on a Spline curve in the α_1 system. It is important not to use all of the contour points while fitting the spline. In many cases, there are more than a thousand points in the original image. This gives an over-constrained solution.

3.2 Thinning Contours

We must supply a list of control points to α_1 that will fit a Spline accurately to the original real-world feature. Therefore, we must decide which points contribute to the actual shape of the feature and which points are simply redundant.

Obviously, regions of high curvature are important to the overall shape of the feature, while low curvature regions will not play as important a role. We fit lines to each contour and represent them as polyline segments in α_1. Each line only consists of its endpoints rather than all the image points along its length. All of the line segments and splines that make up a particular contour are combined together using the α_1 profile curve.

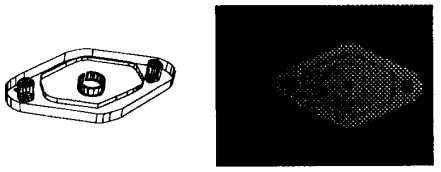

Figure 2: A rough α_1 surface model extracted from the machine vision

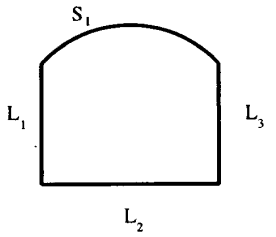

Figure 3: L_1, L_2, and L_3 are fit to lines before they are used in α_1. S_1 has too much curvature and all of its points are used to describe a piecewise spline.

An example is in figure 3. The straight lines in this closed contour are found and the corner points are used as "important" points to the α_1 model. Points along the top arc are all used so that a spline can be fit to them accurately. The final region is represented as the combination of the lines and curves that make up its length.

4 Experiments

Two 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 experiment tested the 3-d sensing module, the sensing→CAD interface, and a new method for DRFSM design.

4.1 Methodology for Inspection and Reverse Engineering

We use a B/W CCD camera mounted on a robot arm, and a coordinate measuring machine (CMM) to sense the mechanical part. This allows us to combine vision, which can be relatively quick, and touch, which is typically slow but produces high accuracy. Vision is used not only to sense the part, but also to observe the probe, which must be manoeuvred to contact the part, running the risk of breakage. This combination requires control of disparate systems.

DEDS are suitable for modeling robotic observers as they provide a means for tracking the *continuous*, *discrete* and *symbolic* aspects of the scene under consideration [4, 16, 17]. Thus the DEDS controller will be able to *model* and *report* the state evolution of the inspection process.

In inspection, the DEDS guides the sensing machines to the parts of the objects where discrepancies occur between the real object (or a CAD model of it) and the recovered structure data points and/or parameters. The DEDS formulation also compensates for noise in the sensor readings (both ambiguities and uncertainties) using a probabilistic approach for computing the 3-D world parameters [19]. The 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*; an exciting extension is to use a similar DEDS observer for moving agents and subsequently design *behaviors* through a learning stage.

A dynamic recursive finite state machines (DRFSM) implementation of a discrete event dynamic system (DEDS) algorithm is used to facilitate the state recovery of the inspection process. DRFSM are a particularly apt choice for the exploration of a set of machined parts which exhibit a recursive nature. As described below, we consider a part's visual features to be related as in tree structure. A feature which is contained within another feature will generally be of a smaller scale than its parent, even though it might be of the same 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. For more detailed information about experiments with DEDS and DRFSM, please see our related technical reports ([20, 21, 22]).

The CAD modeller we use is α_1, from the University of Utah. This modeller offers us a comprehensive representation for complex geometries, as well as a diverse set of machined features [2] and interfaces to automated milling and rendering tools. Among the machined features supported by α_1 which are useful to our research are:

[2] The reader must be careful to distinguish between *mechanical* or *machined* features and *visual* or *geometric* features.

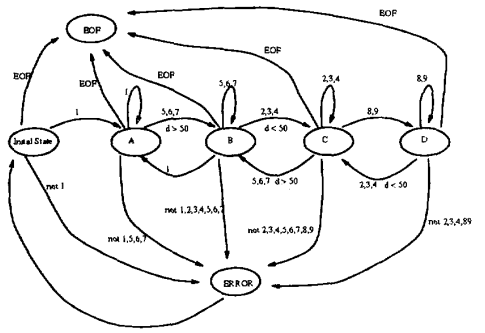

Figure 4: State Machine Used in Test

- holes
- slots
- profile pockets[3]
- profile grooves
- profile sides

All of these features are essentially planar curves which are extruded along a line.

4.2 Control Process Experiment

The state machine's viability was first demonstrated using a text-only interface. 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 on top of the coordinate measuring machine table to simplify the vision algorithms. The camera was placed on a stationary tripod at the base of the table so that the part was always in view.

Once the first level of the DRFSM was created, the experiment proceeded as follows: First, an image was captured from the camera. Next, the appropriate image processing takes place to find the position of the part, the number of features observed (and the recursive string), and the location of the probe. A program using this information produces a state signal that is appropriate for the scene. The signal is read by the state machine and the next state is produced and reported. Each closed feature is treated as a recursive problem, as the probe enters a closed region, a new level of the DRFSM is generated with a new transition vector. This new level then drives the inspection for the current closed region.

[3] "profile" features can be defined by arcs, lines, and piecewise parametric cubics, linked end to end to form complex curves.

The specific dynamic recursive DEDS automaton generated for the test was a state machine G (shown in figure 4.) 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}. The state transitions were controlled by the input signals supplied by intermediate vision programs. There are four stable states A,B,C, and D that describe the state of the probe and part in the scene. The three other states, Initial, Error, and EOF specify the actual 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) towards the part until contact was made. Next, the probe backed off and again approached until the probe and part overlapped. The automaton was forced into an error state by approaching from the other side of the part much too fast. The probe was not seen until it was too close to the object body. Because a transition from state A to C is invalid, an error state is reached. The part used was a simple one with only one hole, that is, it is represented by : C(C()).

In another sequence, the part was more complex. The representation was recovered to be the following string : C(C(),C(C()),C()). The probe was introduced into the scene and moved legally towards the part. Next, the probe backed off and again approached until the probe and the part overlapped. 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. Each image was displayed on a terminal window as it was captured along with the corresponding state of the automaton. The same state representations are displayed for different layers in the DRFSM (i.e., for different features.)

4.3 Model-Building Experiment

A DRFSM DEDS algorithm is used to coordinate the movement of the robot sensor and the probe. Feedback is provided to the robot arm, based on visual observations, so that the object under consideration can be explored. This DRFSM was generated by GIJoe, a graphical tool for building FSMs that was modified to build DRFSMs[5]. The output of GIJoe was compiled and linked with a text-only driver for testing and then linked directly to the experimental sensing code.

The DEDS control algorithm will also guide the probe to the relevant parts of the objects that need to be explored in more detail (curves, holes, complex structures, etc.) Thus, the DEDS controller will be able to *model, report,* and guide the robot and the probe to reposition *intelligently* in order to recover the structure and shape parameters. The data and parameters derived from the sensing agent are fed into the CAD system for designing the geometry of the part under inspection.

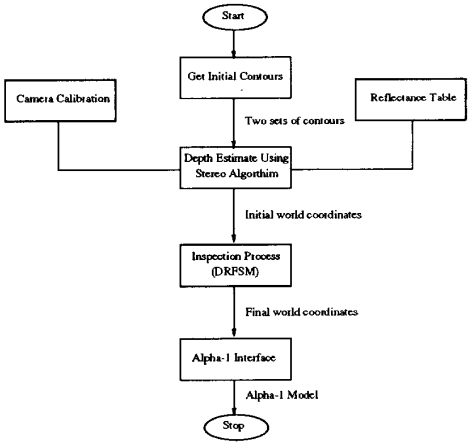

Figure 5: Block Diagram For the Experiment

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

The first step in running this experiment was setting the lighting conditions as desired (same conditions when constructing the reflectance map table), then initializing the robot and the camera and set them to initial positions. The experiment starts by taking images for the object from two positions, to generate two sets of contours to be fed into the stereo module for depth estimation. Using the stereo module with the assistance of the reflectance map table and the camera calibration module, an initial set of world coordinates for these contours is generated. Next, the DRFSM DEDS machine drives the probe and the robot arm holding the camera to inspect the object using the information generated from the stereo module and the relation between the object's contours.

The controlling DEDS machine performed as expected. Results for one of the two models used is shown in Figure 6.

5 Summary

We use an observer agent with some sensing capabilities (vision and touch) to actively gather data (measurements) 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 is then 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.

5.1 Completed activities

- Designed the DRFSM DEDS framework for recursive inspection.
- Implemented image processing modules for recognizing features and probe position on the parts.
- Designed and implemented visual structure recovery techniques for machine parts (using stereo, contour and illumination map data,) and implemented calibration routines.
- Designed and implemented a sensing $\longrightarrow$ CAD interface for generating α_1 code for bodies from depth, contour (and data reduction,) illumination map, and the recursive feature relationships.
- Implemented the DRFSM DEDS automata for recursive inspection (using robot-held camera, probe and actual parts.)
- Designed sensor and strategy-based uncertainty modelling techniques for the robot-held camera, for recovering the DEDS transitional "events" with uncertainty.
- Designed and implemented a modification to an existing reactive behavior design tool (GIJoe) to accommodate "dumping" the code of DRFSM DEDS from a graphical interface (used to draw the inspection control automaton.)

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, inspect and refine 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 re-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 intended to be ultimately as autonomous as possible. We study what parts of the system can be implemented in hardware. Some parts seem to be inherently suited to hardware, while some other parts of the system may be possible to put in hardware, but experimentation will provide the basis for making that decision. Providing language interfaces between the different components in the inspection and reverse engineering control loop is an integral part of the project.

The software and hardware requirements of the environment are the backbone for this project. We selected parts of the system for possible hardware implementation. The DEDS model, as an automaton controller, is very suitable for Path Programmable Logic (PPL) implementation. A number of the visual sensing algorithms could be successfully implemented in PPL, saving considerable computing time. There is a lot of interfacing involved in constructing the inspection and reverse engineering environments under consideration. Using multi-language object-based communication and control methodology between the three major components (Sensing, CAD and CAM) is essential.

We intend to develop the CAD interface to be more accurate and to accept more complicated models. The goal is to enhance the automatic programming interface between the data obtained in the sensing module to the α_1 programming environment. The parametric and 3-D point descriptions are to be integrated to provide consistent and efficient surface descriptions for the CAD tool. For pure inspection purposes the computer aided geometric description of parts could be used as a *driver* for guiding both the robotic manipulator and the coordinate measuring machine for exploring the object and recognizing discrepancies between the real part and the model.

The computer aided design parameters are then to be used for manufacturing the prototypes. Considerable effort has been made for automatically moving from a

computer aided geometric model to a process plan for making the parts on the appropriate NC machines and then to automatically generate the appropriate machine instructions [9]. We intend to use the Monarch VMC-45' milling machine as the manufacturing host. The α_1 system will produce the NC code for manufacturing the parts.

6 Conclusions

We propose a new strategy for inspection and/or reverse engineering. We concentrate on the inspection of machine parts. We also describe a framework for constructing a full environment for generic inspection and reverse engineering. The problem is divided into *sensing*, *design*, and *manufacturing* components with an underlying software and hardware backbone. This project aims at developing control strategies for sensing the world and coordinating the different activities between the phases. We use a recursive DEDS DRFSM framework to construct an intelligent observer module for inspection. The developed framework utilizes existing knowledge to formulate an adaptive and goal-directed strategy for exploring mechanical parts.

7 Bibliography

References

[1] P. K. Allen, *Robotic Object Recognition Using Vision and Touch*, 1987. Kluwer Academic Publishers, Norwell, MA.

[2] J. Aloimonos and David Shulman, *Integration of Visual Modules An Extension of the Marr Paradigm*. Academic Press, 1989.

[3] R. Bajcsy, "Active Perception," *Proceedings of the IEEE*, Vol. 76, No. 8, August 1988.

[4] A. Benveniste and P. L. Guernic, "Hybrid Dynamical Systems Theory and the SIGNAL Language," *IEEE Transactions on Automatic Control*, Vol. 35, No. 5, May 1990.

[5] M. J. Bradakis, "Reactive Behavior Design Tool," Master's Thesis, Computer Science Department, University of Utah, January 1992.

[6] T. M. Carter, K. F. Smith, S. R. Jacobs, and R. M. Neff, "Cell Matrix Methodologies for Integrated Circuit Design," *Integration, The VLSI Journal*, 9(1), 1990.

[7] F. Chaumette and P. Rives, "Vision-Based-Control for Robotic Tasks," In *Proceedings of the IEEE International Workshop on Intelligent Motion Control*, Vol. 2, pp. 395-400, August 1990.

[8] J. Craig, *Introduction To Robotics*, Addison-Wesley, 1989.

[9] S. Drake and S. Sela, "A Foundation for Features," *Mechanical Engineering*, 111(1), January 1989.

[10] J. Ens and P. Lawrence, "An Investigation of Methods for Determining Depth from Focus" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 2, February 1993.

[11] J. Gu and K. Smith, "A Structured Approach for VLSI Circuit Design," *IEEE Computer*, 22(11), 1989.

[12] C. D. Hansen and T. C. Henderson, "CAGD-Based Computer Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(10) : 1181-1193, 1989.

[13] L. Kitchen, A. Rosenfeld. "Grey Level Corner Detection", *Pattern Recognition Letters* vol. 1, pp. 95-102.

[14] G. Lindstrom, J. Maluszynski, and T. Ogi, "Using Types to Interface Functional and Logic Programming," July 1990, 10 pp. technical summary submitted to 1991 SIGPLAN Symposium on Principles of Programming Languages.

[15] D. Marr, E. Hildreth, "Theory of edge detection", *Proceedings Royal Society of London Buletin*, vol. 204, pp. 301-328, 1979.

[16] A. Nerode and J. B. Remmel, "A Model for Hybrid Systems," Presented at the Hybrid Systems Workshop, Mathematical Sciences Institute, Cornell University, May 1991.

[17] C. M. Özveren, *Analysis and Control of Discrete Event Dynamic Systems : A State Space Approach*, Ph.D. Thesis, Massachusetts Institute of Technology, August 1989.

[18] R. F. Riesenfeld, "Mathematical Methods in Computer Aided Geometric Design," chapter Design Tools for Shaping Spline, Academic Press 1989.

[19] T. M. Sobh and R. Bajcsy, "A Model for Observing a Moving Agent," *Proceedings of the Fourth International Workshop on Intelligent Robots and Systems (IROS '91)*, Osaka, Japan, November 1991.

[20] 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, October 1992.

[21] 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, March 1993.

[22] T. M. Sobh, J. C. Owen, C. Jaynes, M. Dekhil, and T. C. Henderson, *Intermediate Results in Active Inspection and Reverse Engineering*, Technical Report UUCS-93-014, Department of Computer Science, University of Utah, June 1993.

[23] M. Swanson, R. Kessler, "Domains : efficient mechanisms for specifying mutual exclusion and disciplined data sharing in concurrent scheme," *First U.S./Japan Workshop on Parallel*, August 1989.

[24] J. A. Thingvold and E. Cohen, "Physical Modeling with B-Spline Surfaces for Interactive Design and Animation," *Proceedings of the 1990 Symposium on Interactive 3-D graphics*, ACM, March 1990.

[25] R. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", *Radiometry - (Physics-Based Vision)*, L. Wolff, S. Shafer, G. Healey, eds., Jones and Bartlett, 1992

[26] Unimation,, *PUMA Mark II robot equipment manual*, Unimation Inc., Danbury, February 1983.

[27] Unimation,, *User's guide to VAL-II*, Unimation Inc., Danbury, April 1983.

[28] R. Willson, Personal Communication, 1993.