A Graphical Environment and Applications for Discrete Event and Hybrid Systems in Robotics and Automation.

Tarek M. Sobh, Peter-Pike Sloan, and Mohamed Dekhil[1]

UUCS-94-030

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

November 10, 1994

Abstract

In this paper we present an overview for the development of a graphical environment for simulating, analyzing, synthesizing, monitoring, and controlling complex discrete event and hybrid systems within the robotics, automation, and intelligent system domain. We start by presenting an overview of discrete event and hybrid systems, and then discuss the proposed framework. We also present two applications within the robotics and automation domain for such complex systems. The first is for formulating an observer for manipulating agents, and the second is for designing sensing strategies for the inspection of machine parts.

[1]This work was supported in part by 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. This report was submitted as a paper to the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '95).

A Graphical Environment and Applications for Discrete Event and Hybrid Systems in Robotics and Automation

Tarek M. Sobh, Peter-Pike Sloan, and Mohamed Dekhil*

Abstract

In this paper we present an overview for the development of a graphical environment for simulating, analyzing, synthesizing, monitoring, and controlling complex discrete event and hybrid systems within the robotics, automation, and intelligent system domain. We start by presenting an overview of discrete event and hybrid systems, and then discuss the proposed framework. We also present two applications within the robotics and automation domain for such complex systems. The first is for formulating an observer for manipulating agents, and the second is for designing sensing strategies for the inspection of machine parts.

1 Introduction

Hybrid systems, in which digital and analogue devices and sensors interact over time, is attracting the attention of researchers. Representation of states and the physical system condition includes continuous and discrete numerics, in addition to symbols and logical parameters. Most of the current robotics, automation, and intelligent systems problems, as well as problems in other domains, fall within the description of hybrid systems. There are many issues that need to be resolved, among them, definitions for observability, stability and stabilizability, controllability in general, uncertainty of state transitions and identification of the system.

The underlying mathematical representation of complex computer-controlled systems is still insufficient to create a set of models which accurately captures the dynamics of the systems over the entire range of system operation. We remain in a situation where we must tradeoff the accuracy of our models with the manageability of the models. Closed-form solutions of mathematical models are almost exclusively limited to linear system models. Computer simulation of nonlinear and discrete-event models provide a means for off-line design of control systems. Guarantees of system performance are limited to those regions where the robustness conditions apply. These conditions may not apply during startup and shutdown or during periods of anomalous operation.

Recently, attempts have been made to model low and high-level system changes in automated and semi-automatic systems as discrete event dynamic systems (DEDS). Several attempts to improve the modeling capabilities are focused on mapping the continuous world into a discrete one. However, repeated results are available which indicate that large interactive systems evolve into states where minor events can lead to a catastrophe. Discrete event and hybrid system formulations have been used in many domains to model and control system state changes within a process. Some of the domains include: Manufacturing, Robotics, Autonomous Agent Modeling, Control Theory,

*This work was supported in part by 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.

Assembly and Planning, Concurrency Control, Distributed Systems, Hierarchical Control, Highway Traffic Control, Autonomous Observation Under Uncertainty, Operating Systems, Communication Protocols, Real-Time Systems, Scheduling, and Simulation.

A number of tools and modeling techniques are being used to model and control discrete event systems in the above domains. Some of the modeling strategies include: Timed, untimed and stochastic Petri Nets and State Automata, Markovian, Stochastic, and Perturbation models, State Machines, Hierarchical State Machines, Hybrid Systems Modeling, Probabilistic Modeling (Uncertainty Recovery and Representation), Queuing Theory, and Recursive Functions.

We next review the development of a graphical environment for simulating, analyzing, synthesizing, monitoring, and controlling complex discrete event and hybrid systems. Then we proceed to discuss two applications within robotics and automation.

2 The Proposed Environment

We have built a software environment to aid in the design, analysis and simulation of Discrete Event and Hybrid Systems. The environment allows the user to build a system using either Finite State Machines or Petri-Nets. The environment runs under X/Motif and supports a graphical DES (Discrete Event System) hybrid controller, simulator, and analysis framework. The framework allows for the control, simulation and monitoring of dynamic systems that exhibits a combination of symbolic, continuous, discrete, and chaotic behaviors, and includes stochastic timing descriptions (for events, states, and computation time), probabilistic transitions, controllability and observability definitions, temporal, timed, state space, petri-nets, and recursive representations, analysis, and synthesis algorithms. The environment allows not only the graphical construction and mathematical analysis of various timing paths and control structures, but also produces C code to be used as a controller for the system under consideration.

Using the environment is fairly simple. For finite state machines the designer uses the mouse to place states (represented by ovals) and connect them with events (represented by arrows). Transitions and states can be added, moved and deleted easily. Figure 1 is an example of a simple stochastically timed FSM, containing 4 states and 5 events.
The probabilities on the events (that is, which path to navigate in the automaton) is designated using the mark field in the status dialog box. The different timings (on event and state times) and distribution function type, mean and variance can be assigned through the status dialog box too. The allowable distributions are currently restricted to Gaussian and exponential functions, but can be easily extended to arbitrary discrete or continuous distributions. A window shows the distribution function at a state or event, and also allows the user to do queries. For example: queries on whether a path time probability is greater or less than a give time, or combined timing distributions to reach a goal state through various paths, etc. The dialog box allows the user to perform queries of various kinds. The currently selected state/event is drawn with a dashed line, and the information in the status window pertains to it. Optimizing paths based on stochastic timing can also be performed, in that case, windows will pop out with the event path, and the status window will have the combined distribution function. Figure 2 presents an automaton model in the environment. The environment also produces C code for controlling the system under consideration.
In our PN model we have extended the definition of stochastic timed Petri Nets, to have additional timings. Our model has three times associated with it, a place time, a delay time, and an event

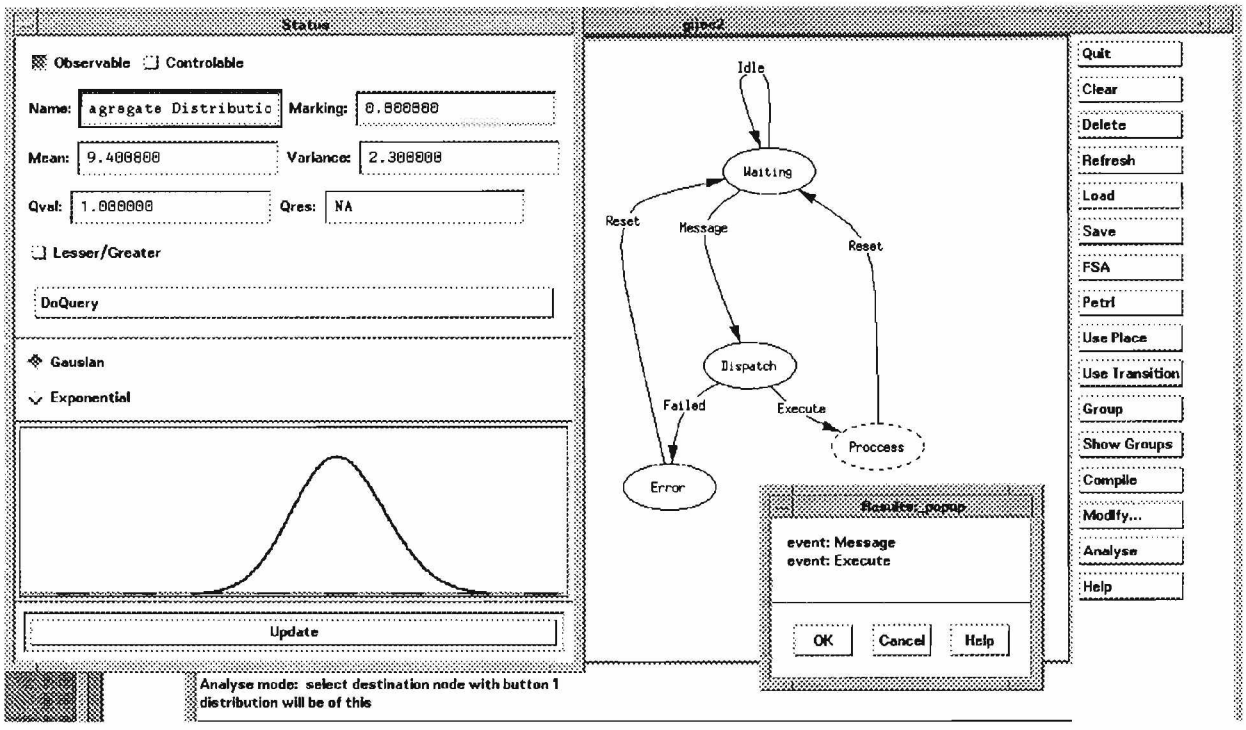


Figure 1: A stochastically timed FSM window during analysis

time (see Figure 3). The place time is a time where the token is held back, and delays the enabling of the transition, this represents the computation time of that place. The delay time is a time associated with the input arcs to a transition, it represents the time to leave the corresponding place. The event time is analogous to the single time in stochastic timed Petri Nets which is called *firing time*. We believe that this lends to a more intuitive representation of the times, and simplifies the modeling task since it captures more details than the original timed Petri net model.

We can define the new model as:

$$PN = (P, T, A, W, x_0)$$

where,

- P = set of places with associated random variables
- T = set of transitions
- $A = A_{in} \cup A_{out}$ with
  - A_{in} set of elements from $\{P \times T\}$ with associated random variables
  - A_{out} set of elements from $\{T \times P\}$
- W = a weight function, $w : A \rightarrow \{1, 2, 3, \ldots\}$
- x_0 is an initial marking

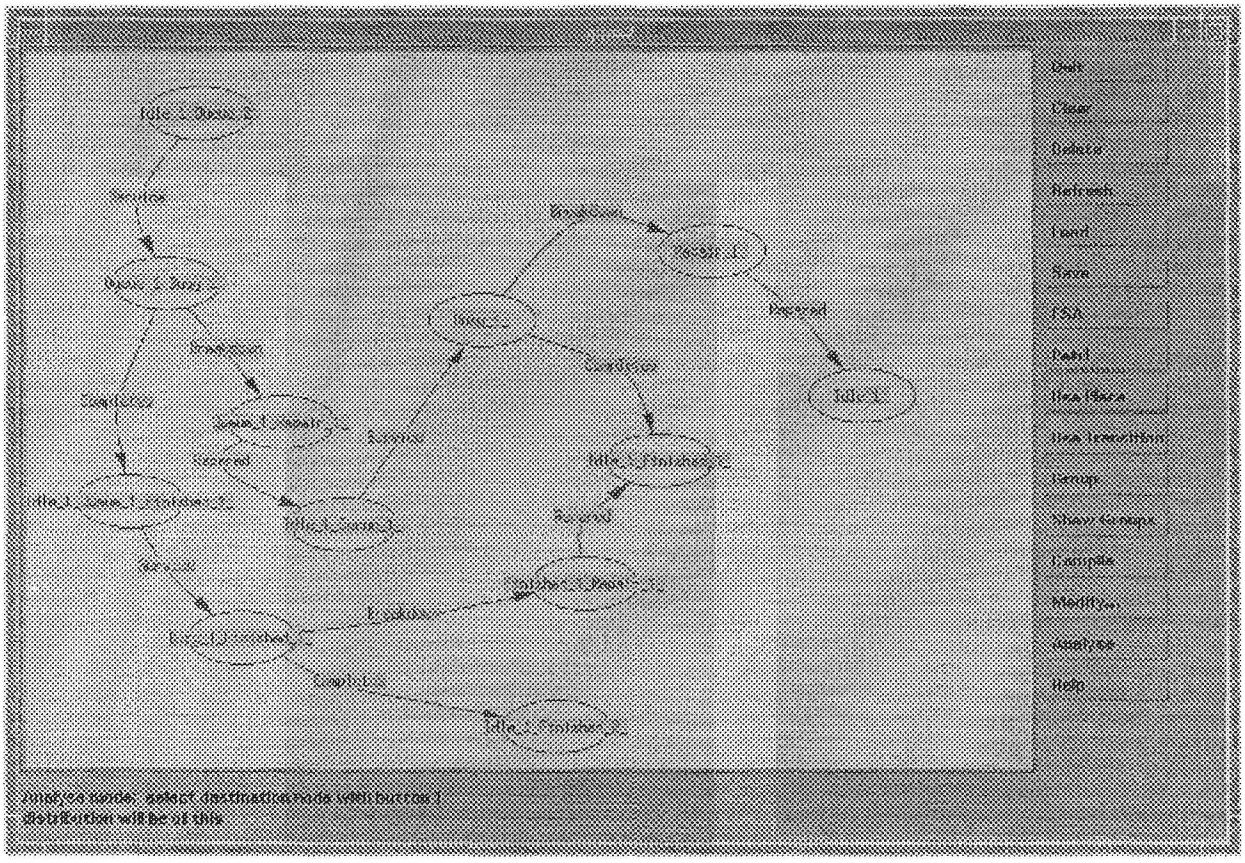


Figure 2: A snap shot of the FSM environment

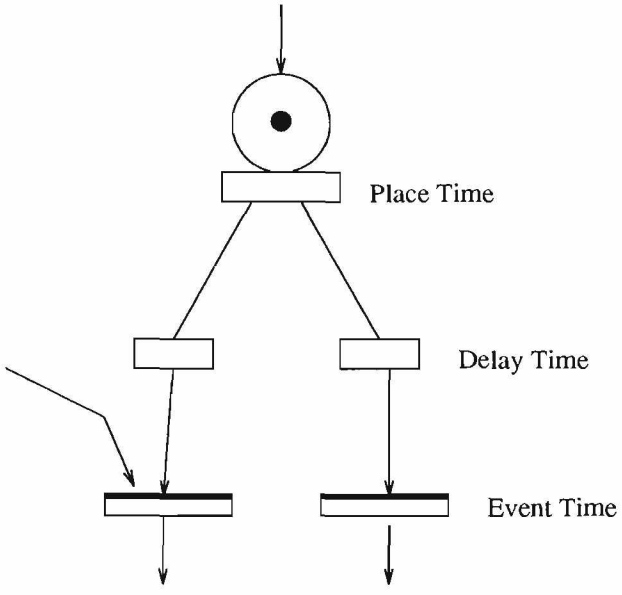


Figure 3: The proposed three time zones for a timed Petri net.

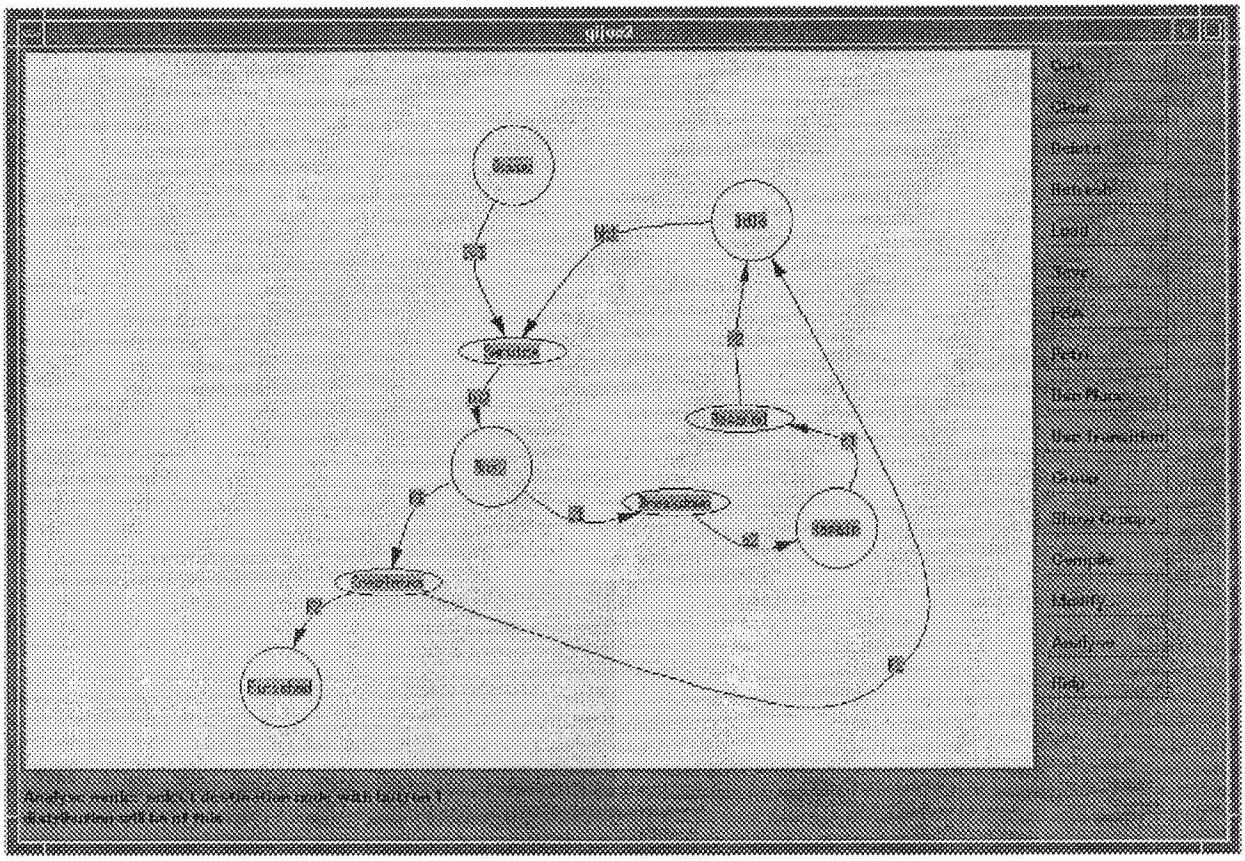


Figure 4: A snap shot of the Petri-net environment

The environment for Petri-Nets is similar. Places are represented graphically by circles, transitions by ellipses, and arcs by arrows. As mentioned above, there are three locations where one can place timing information, on the events - the event time, which is the time the actual event takes, place time - when a token is moved, through a transition firing, there is a place time, which hides the token until it has expired, the final time is a delay time, this comes into effect when a transition fires, it is the time for the event to reach the transition, the event time will not start until all of its input tokens delay time has expired. Figure 4 depicts a snap shot of the Petri-Net environment in action.

The system generates C code for the user hybrid system, so one can simulate and control an actual system using the code. The C code is currently generated for FSMs (soon code will be generated for PN's too). A Petri Net will be converted to a FSM before code is generated, all of the timing is then placed on the events. The user has to select the initial state, and provide the function for simulating/generating the events, the code will keep track of the elapsed simulated time, and will return when it reaches a state with no transitions.

The environment allows conversion back and forth between the FSM and PN models. Conversion to a Petri net is straight forward, but one looses the event probabilities. The only thing that's needed is to create a transition for every event. Conversion from a Petri-net to a FSM is only possible if the PN is k-bounded, which means no place can ever have more than k tokens. The system generates a state for every possible marking of that net. The states are represented as the

marking, the events are just the transitions. Three 3 times are pushed into the events, The system convolves the maximum of the input delays, with the event, and the maximum of the place times. The maximum function is a standard convolution, except that the maximum is used instead of multiplication.

The algorithm for generating all of the markings starts with some initial marking, then goes through all of the possible transitions, if it can fire, the firing is simulated, and the new marking is inserted in the set of states, if it is already represented, the transition is kept; otherwise the transition is kept and recursion is done with the new marking. This process is repeated till no transitions can be fired.

Our system serves as much-needed graphical simulator, analyzer, synthesizer, monitor, and controller for complex hybrid systems models using either Petri nets or FSMs high-level frameworks.

3 Discrete Event Observation Under Uncertainty

We present a new framework and representation for the general problem of observation. The system being studied can be considered as a "hybrid" one, due to the fact that we need to report on *distinct* and *discrete* visual states that occur in the *continuous*, *asynchronous* and three-dimensional world, from two-dimensional observations that are sampled periodically. In other word, the system being observed and reported on consists of a number of continuous, discrete and symbolic parameters that vary over time in a manner that might not be "smooth" enough for the observer, due to visual obscurities and other perceptual uncertainties.

The problem of observing a moving agent was addressed in the literature extensively. It was discussed in the work addressing tracking of targets and, determination of the optic flow [2,7,15,33], recovering 3-D parameters of different kinds of surfaces [6,20,31,32], and also in the context of other problems [1,3,8,11]. However, the need to *recognize*, *understand* and *report* on different visual steps within a dynamic task was not sufficiently addressed. In particular, there is a need for high-level symbolic interpretations of the actions of an agent that attaches meaning to the 3-D world events, as opposed to simple recovery of 3-D parameters and the consequent tracking movements to compensate their variation over time.

In this work we establish a framework for the general problem of observation, recognition and understanding of dynamic visual systems, which may be applied to different kinds of visual tasks. We concentrate on the problem of observing a manipulation process in order to illustrate the ideas and motive behind our framework. We use a discrete event dynamic system as a high-level structuring technique to model the visual manipulation system. Our formulation uses the knowledge about the system and the different actions in order to solve the observer problem in an efficient, stable and practical way. The model incorporates different hand/object relationships and the possible errors in the manipulation actions. It also uses different tracking mechanisms so that the observer can keep track of the workspace of the manipulating robot. A framework is developed for the hand/object interaction over time and a stabilizing observer is constructed. Low-level modules are developed for recognizing the "events" that causes state transitions within the dynamic manipulation system. The process uses a coarse quantization of the manipulation actions in order to attain an active, adaptive and goal-directed sensing mechanism.

The work examines closely the possibilities for errors, mistakes and uncertainties in the visual manipulation system, observer construction process and event identification mechanisms, leading to a DEDS formulation with uncertainties, in which state transitions and event identification is asserted according to a computed set of 3-D uncertainty models.

We motivate and describe a DEDS automaton model for visual observation in the next section and then proceed to formulate our framework for the manipulation process and the observer construction. Then we develop efficient low-level event-identification mechanisms for determining different manipulation movements in the system and for moving the observer. Next, the uncertainty levels are discussed. Some results from testing the system are enclosed.

3.1 Hybrid and Discrete Event Dynamic Systems for Robotic Observation

Hybrid systems, in which digital and analogue devices and sensors interact over time, is attracting the attention of researchers. Representation of states and the physical system condition includes continuous and discrete numerics, in addition to symbols and logical parameters. Most of the current vision and robotics problems, as well as problems in other domains, fall within the description of hybrid systems. There as many issues that need to be resolved, among them, definitions for observability, stability and stabilizability, controllability in general, uncertainty of state transitions and identification of the system. The general observation problem falls within the hybrid system domain, as there is a need to report, observe and control *distinct* and *discrete* system states. There is also a need for recognizing *continuous* 2-D and 3-D evolution of parameters. Also, there should be a *symbolic* description of the current state of the system, especially in the manipulation domain.

We do not intend to give a solution for the problem of defining, monitoring or controlling such hybrid systems in general. What we intend to present in this work is a framework that works for the class of hybrid systems encountered within the robotic observation paradigm. The representation we advocate allows for the symbolic and numeric, continuous and discrete aspects of the observation task. We conjecture that the framework could be explored further as a possible basis for providing solutions for general hybrid systems representation and analysis problems.

We suggest the use of a representation of discrete event dynamic systems, which is augmented by the use of a concrete definition for the events that causes state transitions, within the observation domain. We also use some uncertainty modeling to achieve robustness and smoothness in asserting state and continuous event variations over time.

Dynamic systems are sometimes modeled by finite state automata with partially observable events together with a mechanism for enabling and disabling a subset of state transitions [19,22,23], the reader is referred to those references for more information about this class of DEDS representation. We propose that such a DEDS skeleton is a suitable high-level framework for many vision and robotics tasks, in particular, we use the DEDS model as a high-level structuring technique for a system to observe a robot hand manipulating an object.

3.1.1 Discrete event dynamic systems for active visual sensing

An example of a high-level DEDS controller for part inspection can be seen in Figure 5. This finite state machine has some observable events that can be used to control the sequencing of the process. The machine remains in state A until a part is loaded. When the part is loaded, the machine transitions to state B where it remains until the part is inspected. If another part is

available for inspection, the machine transitions to state A to load it. Otherwise, state C, the ending state, is reached. If an interruption occurs, such as a misloaded part or inspection error, the machine goes to state D, the error state.

Our approach uses DEDS to drive a semi-autonomous visual sensing module that is capable of making decisions about the *visual state* of the manipulation process taking place. This module provides both symbolic and parametric descriptions which can be used to observe the process *intelligently* and *actively.*

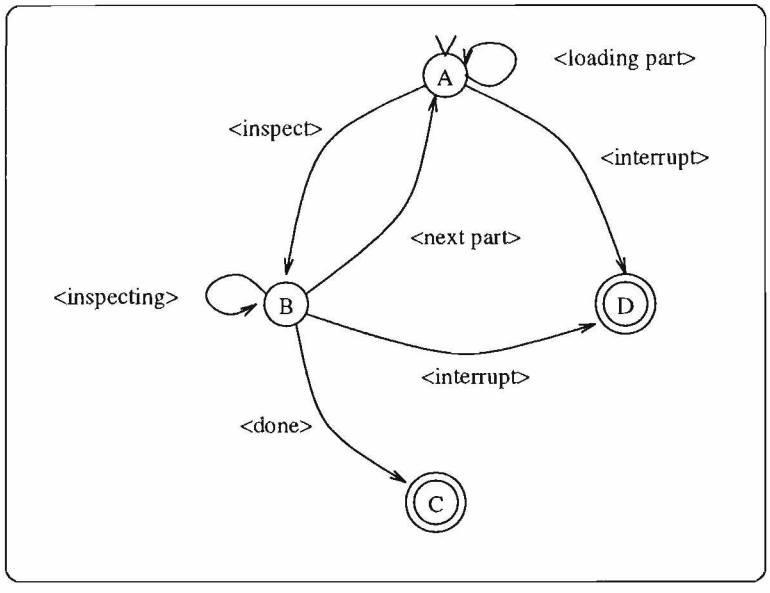


Figure 5: A Simple FSM

A DEDS framework is used to model the tasks that the autonomous observer system executes. This model is used as a high level structuring technique to preserve and make use of the information we know about the way in which a manipulation process should be performed. The state and event description is associated with different visual cues; for example, appearance of objects, specific 3-D movements and structures, interaction between the robot and objects, and occlusions. A DEDS observer serves as an intelligent sensing module that utilizes existing information about the tasks and the environment to make informed tracking and correction movements and autonomous decisions regarding the state of the system.

To be able to determine the current state of the system we need to observe the sequence of events occurring in the system and make decisions regarding the state of the automaton. State ambiguities are allowed to occur, however, they are required to be resolvable after a bounded interval of events. In a *strongly output stabilizable* system, the state of the system is known at bounded intervals and allowable events can be controlled (enabled or disabled) in a way that ensures return in a bounded interval to one of a desired and known set of states (visual states in our case).

One of the objectives is to make the system strongly output stabilizable and/or construct an observer to satisfy specific task-oriented visual requirements. Many 2-D visual cues for estimating 3-D world behavior can be used. Examples include: image motion, shadows, color and boundary information. The uncertainty in the sensor acquisition procedure and in the image processing mechanisms should be taken into consideration to compute the world uncertainty.

The observer framework can be utilized for recognizing error states and sequences. This recognition task will be used to report on *visually incorrect* sequences. In particular, if there is a pre-determined observer model of a particular manipulation task under observation, then it would be useful to determine if something goes wrong with the exploration actions. The goal of this reporting procedure is to alert the operator or autonomously supply feedback to the manipulating robot so that it can correct its actions.

3.1.2 DEDS for Modeling Observers

DEDS can be considered as very suitable tools for modeling observers. In particular, in the manipulation observer domain, there is a need to recognize and report on distinct and discrete visual states, which might represent manipulation tasks and/or sub-tasks. The observer should have the ability to state a symbolic description of the current manipulation agent action. The coarse definition of DEDS states provide a means for such symbolic state descriptions.
The definition for observers and the observer construction process for discrete event systems are very coherent with the requirements for an autonomous robotic observer. The purpose of DEDS observers is to be able to reconstruct the system state, which is exactly the requirements for a visual observer, which needs to recognize, report and possibly act, depending on the visual manipulation state. The notions of controllable actions is easily mapped to some tracking and repositioning procedures that the robotic observer will have to undertake in order to "see" the scene from the "best" viewing position as the agent under observation moves over time. The actions which the observer robot might need to perform, depends on the sequence of "observable" events and the reconstructed state path.
Event description in a visual observer is possibly a combination of different 2-D and 3-D visual data. The visual primitives used in an observer domain could be motion primitives, matching measures, object identification processes, structure and shape parameters and/or a number of other visual cues. The problem with the DEDS skeleton is that it does not allow for smooth state changes under uncertainty in recovering the events. We describe in the next sections techniques that make the transition from a DEDS skeleton into a working hybrid observer for a moving manipulation agent. Stability and stabilizability issues are resolved in the visual observer domain by supplying suitable control sequences to the observer robot at intermittent points in time in order to "guide" it into the "desirable" set of visual states.

3.2 State Modeling and Observer Construction

Manipulation actions can be modeled efficiently within a discrete event dynamic system framework. It should be noted that we do not intend to *discretize* the workspace of the manipulating robot hand or the movement of the hand, we are merely using the DEDS model as a high level structuring technique to preserve and make use of the information we know about the way in which each manipulation task should be performed, in addition to the knowledge about the physical limitations of both the observer and manipulating robots. The high-level state definition permits the observer recognize and report on symbolic descriptions of the task and the physical relationships under observation. We avoid the excessive use of decision structures and exhaustive searches when observing the 3-D world motion and structure.
A bare-bone approach to solving the observation problem would have been to try and visually reconstruct the full 3-D motion parameters of the robot hand, which would have more than six degrees of freedom, depending on the number of fingers and/or claws and how they move. The

motion and shape or structure of the different objects should also be recovered in 3-D, which is complicated especially if some of them are non-rigid bodies. That process should be done in real time while the task is being performed. A simple way of tracking might be to try and keep a fixed geometric relationship between the observer camera and the hand over time. However, the above formulation is inefficient, unnecessary and for all practical purposes infeasible to compute in real time. In addition, that formulation does not provide any kind of interpretation for the *meaning* of the scene evolution, nor does it allow for any symbolic recognition for the task under observation. The limitation of the observer reachability and the extensive computations required to perform the visual processing are motives behind formulating the problem as a hierarchy of task-oriented observation modules that exploits the higher-level knowledge about the existing system, in order to achieve a feasible mechanism of keeping the visual process under supervision.

3.2.1 State Space Modeling

We do a coarse quantization of the *visual manipulation actions* which allows modeling both continuous and discrete aspects of the manipulation dynamics. State transitions within the manipulation domain are asserted according to probabilistic models that determine at different instances of time whether the visual scene under inspection has changed its state within the discrete event dynamic system state space. Mapping the desired visual states to a DEDS skeleton is a straight forward procedure. We attach a DEDS automaton state to each meaningful visual state within a manipulation action. The quantization threshold depends on the application requirement. In other words, the state space can be expanded or contracted depending on the level of accuracy required in reporting and observing. A surgical operation step, performed by a robotic end effector, will obviously require an observer that reports (and possibly control the effector within a closed-loop visual system) with extreme precision. The observer for a robotic manipulator whose task is to pile up heaps of waste would, most likely, report in a crude fashion, thus needing a small number of states. The quantization threshold depends heavily on the nature of the task and the application requirements. The DEDS formulation is flexible, in the sense that it allows different precisions and/or state space models depending on the requirements.

The task of building DEDS automaton skeletons for observer agents can be performed either *manually* or *automatically*. In the manual formation case, the designer would have to draw the automaton model that best suits the task(s) under observation and depending on the application requirements and implement the code for the state machine. Automatic construction of the state machine could be done by having a *learning* stage [17,18] in which a mapping module would form the automaton. This is performed before the actual observation process is invoked. The idea is to supply the module with sets of possible sequences in the form of *strings* of a certain language that the DEDS automaton should minimally accept. The language could be either supplied by an operator, in which case, the resulting automaton performance depends on the relative skill of the operator, or through showing the module a sequence of visual actions and labeling those actions appropriately. The language strings should also be accompanied by a set of transitional conditions as event descriptions. The module would then produce the minimal DEDS automaton, complete with event and state descriptions that accepts the language.

We next discuss building the manipulation model for some simple tasks, then we proceed to develop the observer for these tasks. Formulating the models for the state transitions, the inter-state continuous dynamics and recovering uncertainty will be left for sections 4 and 5 which deal with the different uncertainty levels and event identification mechanisms.

3.2.2 Building the Model

The ultimate goal of the observation mechanism is to be able to know at all (or most) of the time what is the current manipulation process and what is the visual relationship between the hand and the object. The fact that the observer will have to move in order to keep track of the manipulation process, makes one think of the stabilizability principle for general DEDS as a model for the tracking technique that has to be performed by the observer's camera.
In real-world applications, many manipulation tasks are performed by robots, including, but not limited to, lifting, pushing, pulling, grasping, squeezing, screwing and unscrewing of machine parts. Modeling all the possible tasks and also the possible order in which they are to be performed is possible to do within a DEDS state model. The different hand/object visual relationships for different tasks can be modeled as the set of states X. Movements of the hand and object, either as 2-D or 3-D motion vectors, and the positions of the hand within the image frame of the observer's camera can be thought of as the events set Γ that causes state transitions within the manipulation process. Assuming, for the time being, that we have no direct control over the manipulation process itself, we can define the set of admissible control inputs U as the possible tracking actions that can be performed by the hand holding the camera, which actually can alter the visual configuration of the manipulation process (with respect to the observer's camera). Further, we can define a set of "good" states, where the visual configuration of the manipulation process enables the camera to keep track and to know the movements in the system. Thus, it can be seen that the problem of observing the robot reduces to the problem of forming an output stabilizing observer (an observer that can always return to a set of "good" visual states) for the system under consideration.
It should be noted that a DEDS representation for a manipulation task is by no means unique, in fact, the degree of efficiency depends on the designer who builds the model for the task, testing the optimality of a visual manipulation models is an issue that remains to be addressed. Automating the process of building a model was discussed in the previous section. As the observer identifies the current state of a manipulation task in a non ambiguous manner, it can then start using a practical and efficient way to determine the next state within a predefined set, and consequently perform necessary tracking actions to stabilize the observation process with respect to the set of good states. That is, the current state of the system tells the observer what to *look for* in the next step.

- A Grasping Task

We present a simple model for a grasping task. The model is that of a gripper approaching an object and grasping it. The task domain was chosen for simplifying the idea of building a model for a manipulation task. It is obvious that more complicated models for grasping or other tasks can be built. The example shown here is for illustration purposes.
As shown in Figure 6, the model represents a view of the hand at state 1, with no object in sight, at state 2, the object starts to appear, at state 3, the object is in the claws of the gripper and at state 4, the claws of the gripper close on the object. The view as presented in the figure is a frontal view with respect to the camera image plane, however, the hand can assume any 3-D orientation as so long as the claws of the gripper are within sight of the observer, for example, in the case of grasping an object resting on a tilted planar surface. This demonstrates the continuous dynamics aspects of the system. In other words, different orientations for the approaching hand are allowable and observable. State changes occur only when the object appear in sight or when the hand encloses

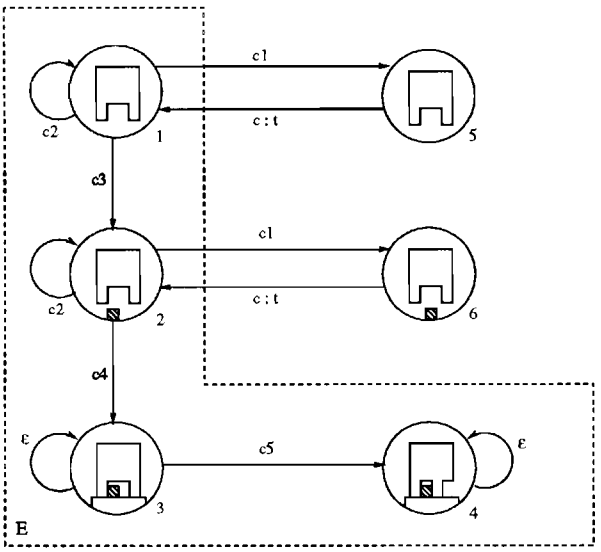


Figure 6: A Model for A Grasping Task

it. The frontal upright view is used to facilitate drawing the automaton only. It should be noted that these states can be considered as the set of good states E, since these states are the expected different visual configurations of a hand and object within a grasping task.
States 5 and 6 represent instability in the system as they describe the situation where the hand is not centered with respect to the camera imaging plane, in other words, the hand and/or object are not in a good visual position with respect to the observer as they tend to escape the camera view. These states are considered as "bad" states as the system will go into a non-visual state unless we correct the viewing position. The set $X = \{1, 2, 3, 4, 5, 6\}$ is the finite set of states, the set $E = \{1, 2, 3, 4\}$ is the set of "good" states. Some of the events are defined as motion vectors or motion vector probability distributions, as will be described later, that causes state transitions and as the appearance of the object into the viewed scene. The transition from state 1 to state 2 is caused by the appearance of the object. The transition from state 2 to state 3 is caused by the event that the hand has enclosed the object, while the transition from state 3 to state 4 is caused by the inward movement of the gripper claws. The transition from the set $\{1, 2\}$ to the set $\{5, 6\}$ is caused by movement of the hand as it escapes the camera view or by the increase in depth between the camera and the viewed scene, that is, the hand moving far away from the camera. The self loops are caused by either the stationarity of the scene with respect to the viewer or by the continuous movement of the hand as it changes orientation but without tending to escape a good viewing position of the observer. In the next section we discus different techniques to identify the events. The controllable events denoted by ": t" are the tracking actions required by the hand holding the camera to compensate for the observed motion. Tracking techniques will later be addressed in detail. All the events in this automaton are observable and thus the system can be represented by the triple $G = (X, \Sigma, T)$, where X is the finite set of states, Σ is the finite set of possible events and T is the set of admissible tracking actions or controllable events.
It should be mentioned that this model of a grasping task could be extended to allow for error detection and recovery. Also search states could be added in order to "look" for the hand if it is no where in sight. The purpose of constructing the system is to develop an observer for the automaton which will enable the determination of the current state of the system at intermittent points in time

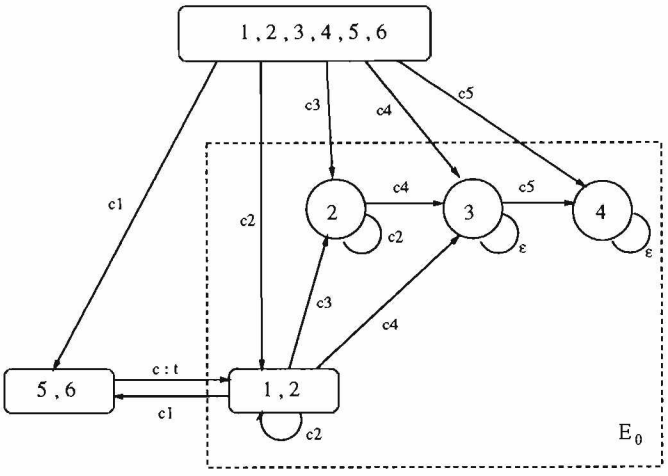


Figure 7: An Observer for the Grasping System

and further more, enable us to use the sequence of events and control to "guide" the observer into the set of good states E and thus stabilize the observation process. Disabling the tracking events will obviously make the system unstable with respect to the set $E = \{1, 2, 3, 4\}$ (can't get back to it), however, it should be noted that the subset $\{3, 4\}$ is already stable with respect to E regardless of the tracking actions, that is, once the system is in state 3 or 4, it will remain in E. The whole system is stabilizable with respect to E, enabling the tracking events will cause all the paths from any state to go through E in a finite number of transitions and then will visit E infinitely often.

3.2.3 Developing the Observer

In order to know the current state of the manipulation process we need to observe the sequence of events occurring in the system and make decisions regarding the state of the automaton, state ambiguities are allowed to occur, however, they are required to be resolvable after a *bounded* interval of events. An observer, have to be constructed according to the visual system for which we developed a DEDS model. The goal will be to make the system a stabilizable one and/or construct an observer to satisfy specific task-oriented visual requirements that the user may specify depending on the nature of the process. It should be noticed that events can be asserted with a specific probability as will be described in the sections to come and thus state transitions can be made according to pre-specified thresholds that compliments each state definition. In the case of developing ambiguities in determining current and future states, the history of evolution of past event probabilities can be used to navigate backwards in the observer automaton till a strong match is perceived, a fail state is reached or the initial ambiguity is asserted.
As an example, for the model of the grasping task, an observer can be formed for the system as shown in Figure 7. It can be easily seen that the system can be made stable with respect to the set E_O (The system always returns to that set).
At the beginning, the state of the system is totally ambiguous, however, the observer can be "guided" to the set E_O consisting of all the subsets of the good states E as defined on the visual system model. It can be seen that by enabling the tracking event from the state (5, 6) to the state (1, 2), all the system can be made stable with respect to E_O. The singleton states represent the instances in time where the observer will be able to determine without ambiguity the current state

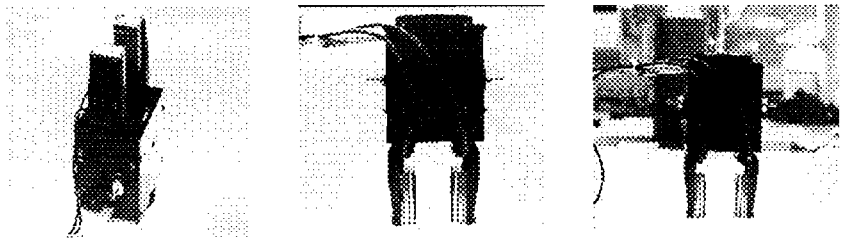


Figure 8: Different Views of the Lord Gripper

of the system.
In the next section we shall elaborate on defining the different events in the visual manipulation system and discuss different techniques for event and state identification. We shall also introduce a framework for computing the uncertainty in determining the observable visual events in the system and a method by which the uncertainty distribution in the system can be used to efficiently keep track of the different observer states and to navigate in the observer automaton.

3.2.4 Examples

Experiments were performed to observe the robot hand. The Lord experimental gripper is used as the manipulating hand. Different views of the gripper are shown in Figure 8. Tracking is performed for some features on the gripper in real time. The visual tracking system works in real time and a position control vector is supplied to the observer manipulator.
Some visual states for a grasping task using the Lord gripper, as seen by the observer camera, is shown in figure 9. The sequence is defined by our model, and the visual states correspond to the gripper movement as it approaches an object an then grasps it.
The full system is implemented and tested for some simple visual action sequences. One such example is shown in figure 10. The automaton encodes an observer which tracks the hand by keeping a fixed geometric relationship between the observer's camera and the hand as so long as the hand does not approach the observer's camera rapidly. In that case, the observer tends to move sideways, that is, dodge and start viewing and tracking from sideways. It can be thought of as an action to avoid collision, due to the fact that the intersection of the workspaces of both robots is not empty. State 1 represents the visual situation where the hand is in a centered viewing position with respect to the observer and viewed from a frontal position. State 2 represents the hand in a non-centered position and tending to escape the visual view, but not approaching the observer rapidly. State 3 represents a "dangerous" situation as the hand has approached the observer rapidly. State 4 represents the hand being viewed from sideways, and the hand is centered within the imaging plane.

After having defined the states, the events causing state transitions can be easily described. Event e_1 represents no hand movements, event e_2 represents all hand movements in which the hand does not approach the camera rapidly. Event e_3 represents a large movement towards the observer. Events e_4 and e_5 are controllable tracking events, where e_4 always compensates for e_2 in order to keep a fixed 3-D relationship and e_5 is the "dodging" action where the observer moves to start viewing from sideways, while keeping the hand in a centered position.
The events can thus be defined precisely as ranges on the recovered world motion parameters. For example, e_3 can be defined as any motion $V_Z \geq d_z$. Event e_1 is defined as any motion such that :

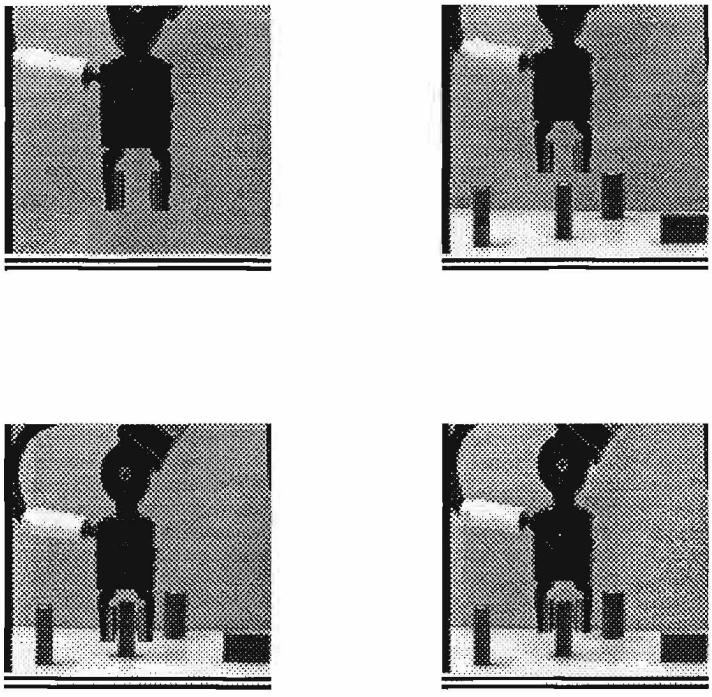


Figure 9: **A Grasping Task :** As seen by the observer's camera

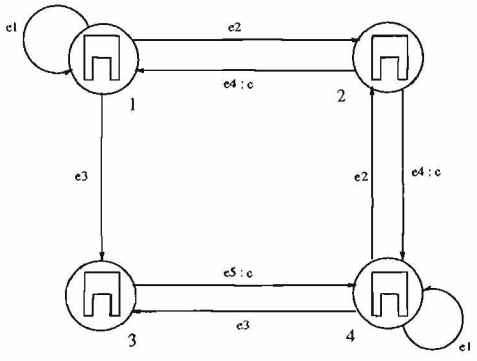


Figure 10: A Model for a Simple Visual Sequence

$$-\epsilon_x \leq V_X \leq \epsilon_x \wedge -\epsilon_y \leq V_Y \leq \epsilon_y \wedge -\epsilon_z \leq V_Z \leq \epsilon_z$$

It should be noted that defining e_1 in this manner helps a lot in suppressing noise. Having defined the events, the task reduces to computing the relevant areas under the distribution curves for the various 3-D motion parameters and computing the probabilities for the ranges of e_1, e_2 and e_3 at states 1 and 4. State transitions is asserted and reported when the probability value exceeds a preset threshold. States 1 and 4 are considered to be the set of stable states, by enabling the tracking events e_4 and e_5 the system can be made stable with respect to that set.

The low level visual feature acquisition is performed on the MaxVideo pipelined video processor at frame rate. The state machine resides on a Sun SparcStation 1. The Lord gripper is mounted on a PUMA 560 arm and the observer's camera is mounted on a second PUMA 560.

3.3 Identifying Motion Events

We use the image motion to estimate the hand movement. This task can be accomplished by either feature tracking or by computing the full optic flow. The image flow detection technique we use is based on the sum-of-squared-differences optic flow. The sensor acquisition procedure (grabbing images) and uncertainty in image processing mechanisms for determining features are factors that should be taken into consideration when we compute the uncertainty in the optic flow.

One can model an arbitrary 3-D motion in terms of stationary-scene/moving-viewer as shown in Figure 11. The optical flow at the image plane can be related to the 3-D world as indicated by the following pair of equations for each point (x, y) in the image plane [20] :

$$v_x = \left\{ x\frac{V_Z}{Z} - \frac{V_X}{Z} \right\} + \left[xy\Omega_X - \left(1 + x^2\right)\Omega_Y + y\Omega_Z \right]$$

$$v_y = \left\{ y\frac{V_Z}{Z} - \frac{V_Y}{Z} \right\} + \left[\left(1 + y^2\right)\Omega_X - xy\Omega_Y - x\Omega_Z \right]$$

where v_x and v_y are the image velocity at image location (x, y), (V_X, V_Y, V_Z) and $(\Omega_X, \Omega_Y, \Omega_Z)$ are the translational and rotational velocity vectors of the observer, and Z is the unknown distance from the camera to the object. In this system of equations, the only knowns are the 2-D vectors v_x and v_y, if we use the formulation with uncertainty then basically the 2-D vectors are random variables with a known probability distribution. A number of techniques can be used to linearize the system of equations and to solve for the motion and structure parameters as random variables [4,5,31].

3.4 Modeling and Recovering 3-D Uncertainties

The uncertainty in the recovered image flow values results from sensor uncertainties and noise and from the image processing techniques used to extract and track features. We use a static camera calibration technique to model the uncertainty in 3-D to 2-D feature locations. The strategy used to find the 2-D uncertainty in the features 2-D representation is to utilize the recovered camera parameters and the 3-D world coordinates (x_w, y_w, z_w) of a known set of points and compute the corresponding pixel coordinates, for points distributed throughout the image plane a number of times, find the actual feature pixel coordinates and construct 2-D histograms for the displacements

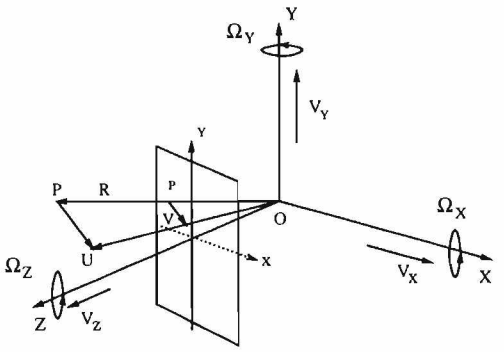


Figure 11: 3-D Formulation for Stationary Scene/Moving Viewer

from the recovered coordinates for the experiments performed. The number of the experiments giving a certain displacement error would be the z axis of this histogram, while the x and y axis are the displacement error. The three dimensional histogram functions are then normalized such that the volume under the histogram is equal to 1 unit volume and the resulting normalized function is used as the distribution of pixel displacement error.

The spatial uncertainty in the image processing technique can be modeled by using synthesized images and corrupting them, then applying the feature extraction mechanism to both images and computing the resulting spatial histogram for the error in finding features. The probability density function for the error in finding the flow vectors can thus be computed as a spatial convolution of the sensor and strategy uncertainties. We then eliminate the unrealistic motion estimates by using the physical (geometric and mechanical) limitations of the manipulating hand. Assuming that feature points lie on a planar surface on the hand, then we can develop bounds on the coefficients of the motion equations, which are second degree functions in x and y in three dimensions, $v_x = f_1(x, y)$ and $v_y = f_2(x, y)$.

The 2-D uncertainties are then used to recover the 3-D uncertainties in the motion and structure parameters. The system is linearized by either dividing the parameter space into three subspaces for the translational, rotational and structure parameters and solving iteratively or using other linearization techniques and/or assumptions to solve a linear system of random variables [4,5,6,31,32,34]. As an example, the recovered 3-D translational velocity cumulative density functions for an actual world motion, $V_X = 0\ cm$, $V_Y = 0\ cm$ and $V_Z = 13\ cm$, is shown in figure 12. It should be noted that the recovered distributions represents a fairly accurate estimation of the actual 3-D motion.

3.5 Utilizing the Discrete Event Observer

State transitions are asserted within the DEDS observer model according to the probability value of the occurrence of an event. Events are thus defined as ranges for the different parameters. The problem then reduces to computing the corresponding areas under the refined distribution curves. An obvious way of using those probability values is to establish some threshold values and assert transitions according to those thresholds. It might be the case that none of the obtained probability values exceeds the set threshold value and/or all values are very low. In that case, there is a good chance that we are at either the wrong automata state. The remedy to such problems can be

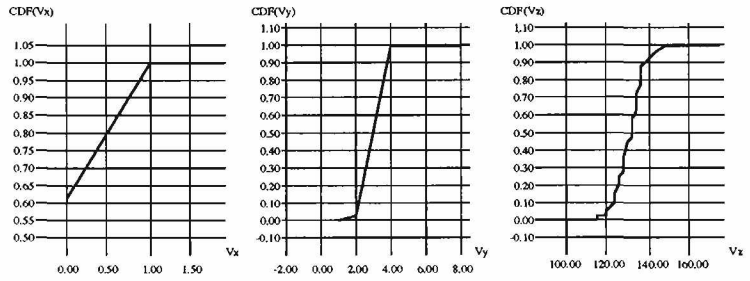


Figure 12: Cumulative Density Functions of the Translational Velocity

implemented through time proximity, that is, wait for a while (which is to be preset) till a strong probability value is registered and/or *backtrack* in the automaton model for the observer till a high enough probability value is asserted, a fail state is reached or the initial ambiguity is asserted. The backtracking strategy can be implemented using a stack-like structure associated with each state that has already been traversed, which includes a sorted list of the computed event probabilities and a father-state variable.

3.6 Experiments

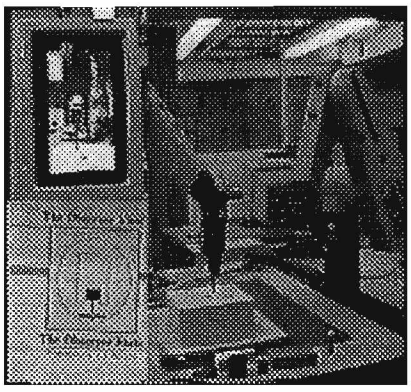
Experiments were performed to observe the robot hand. The low level visual feature acquisition is performed on the Datacube MaxVideo pipelined video processor at frame rate. The observer and manipulating robots are both PUMA 560's and the Lord experimental gripper is used as the manipulating hand.

The experiment was shot with three video camera. The right hand side of the images show the actual observer and manipulation workspace and the different configurations as the experiment proceed. The upper left corner shows the observer view, which is the set of images grabbed by the camera for processing. The lower left corner shows the observer state, that is, what the observer "thinks". A graphical representation of the different states and their change is used. Fail states are represented by an empty box. Figures 13 and 14 illustrate a manipulation experiment. In this sequence the hand tries to insert a peg in a hole. The observer approaches and focuses on the peg and hole when the peg gets nearer to the hole. State changes occur when the hole appears and when insertion is asserted.

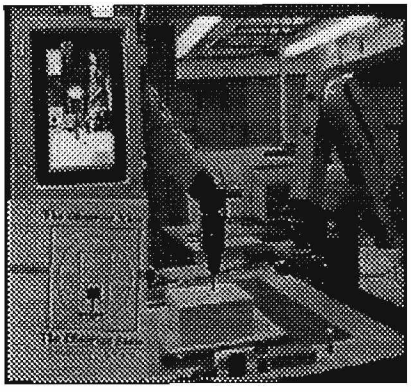
3.7 Conclusions

We described a system for observing a manipulation process. The proposed approach can be generalized for other hybrid systems involving different kinds of quantization requirements for dynamic systems, for sets of discrete, continuous and symbolic parameters. The use of discrete event dynamic systems with uncertainty modeling for the event description enables the observer to recognize tasks robustly. The proposed system also utilizes the a-priori knowledge about the task domain in order to achieve efficiency and practicality. The high level formulation allows for recognizing and reporting on the visual system state as a symbolic description of the observed tasks.

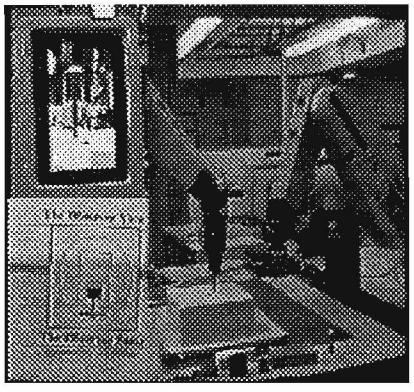
Thus, we have proposed a new approach to solving the problem of observing a moving agent. Our approach uses the formulation of discrete event dynamic systems as a high-level model for the framework of evolution of the visual relationship over time. The proposed formulation can be



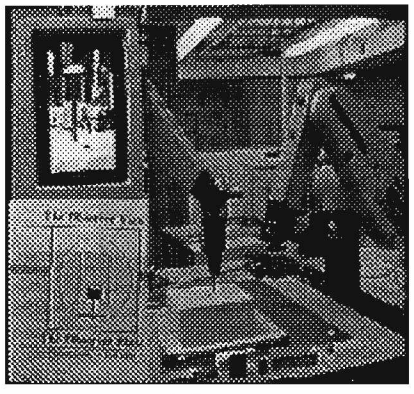
(1)



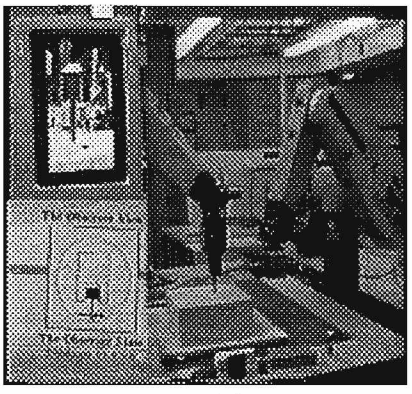
(2)



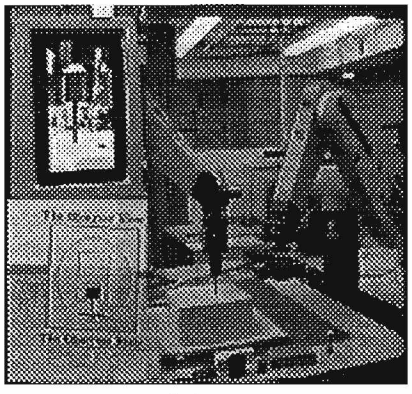
(3)



(4)

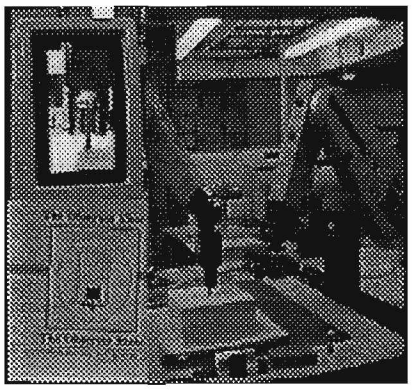


(5)

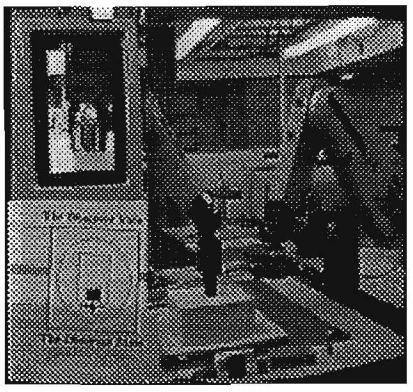


(6)

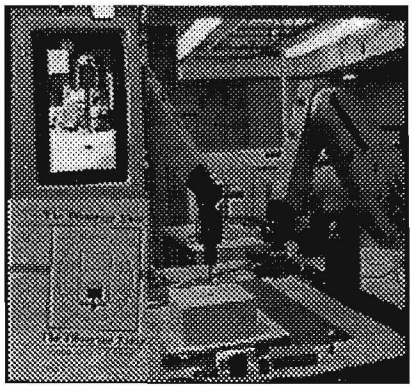
Figure 13: Observer State and View (1)



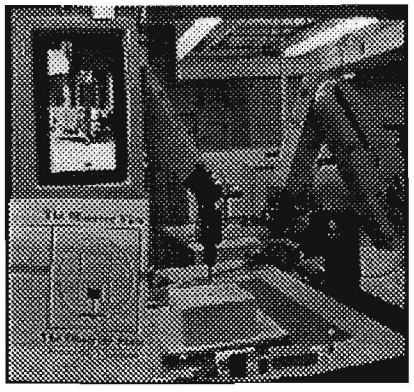
(7)



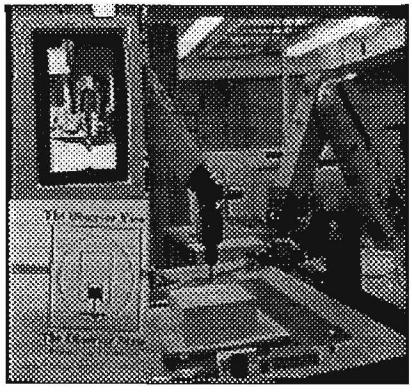
(8)



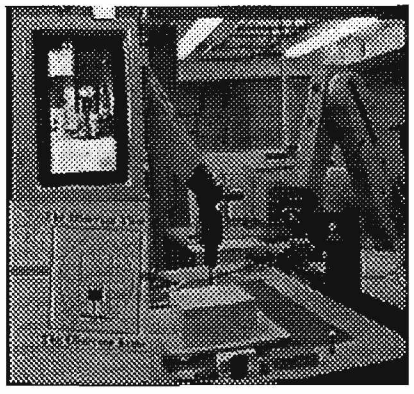
(9)



(10)



(11)



(12)

Figure 14: Observer State and View (2)

extended to accommodate for more manipulation processes. Increasing the number of states and expanding the events set would allow for a variety of manipulating actions.

4 Sensing for Inspection of Machine Parts

This work addresses the application of discrete event dynamic systems (DEDS) for autonomous sensing and inspection as part of the reverse engineering process. A dynamic recursive context for DEDS is presented and its usage for managing a complex hybrid system which has continuous, discrete and symbolic aspects is illustrated. We suggest that the dynamic recursive context is aptly suited to controlling and observing the active inspection of machined parts using such a hybrid system.
Reverse engineering is the process of constructing an accurate representation from sensed data. It can be represented by a closed loop system that consists of four main modules:

- Sensing
- CAD Modeling
- Manufacturing
- Inspection

This closed loop system is the framework we used to develop an integrated CAD/CAM/sensing system for inspection and reverse engineering. The process starts by constructing an initial CAD model using 2-d and 3-d vision, then the inspection module uses this model to drive a coordinate measuring machine (CMM). The results are used to increase the accuracy of the model. Additional sensing iterations could be made until the desired accuracy is obtained. Figure 15 shows this closed loop system.
Most research in reverse engineering ([24, 21, 12, 13, 14, 9, 10]) concentrates on the sensing and fitting techniques required. Hsieh[16] describes a system which does sculptured surface reconstruction with a CMM. The focus of the work is on path planning and surface fitting. If errors occur while gathering data, the system aborts and must be restarted. Van Thiel [35] describes an interactive CMM inspection system. The user is included as part of the control loop, and can abort inspections and call for explorations of particular features. Our work describes an approach that automatically gathers the sense data, processes it, and makes decisions based upon it for reverse engineering.
We use a recursive dynamic strategy for exploring machine parts. A discrete event dynamic system (DEDS) framework is designed for modeling and structuring the sensing and control problems. The dynamic recursive context for finite state machines (DRFSM) is a DEDS representation tailored to the recursive nature of the mechanical parts under consideration.
DRFSM is particularly useful for controlling the inspection module, and this has been an important aspect of our research.

4.1 Modeling and Constructing an Observer

A DEDS framework is used to model the tasks that the autonomous observer system executes. This model is used as a high level structuring technique to preserve and make use of the information we know about the way in which a mechanical part should be explored. The state and event description is associated with different visual cues; for example, appearance of objects, specific 3-D

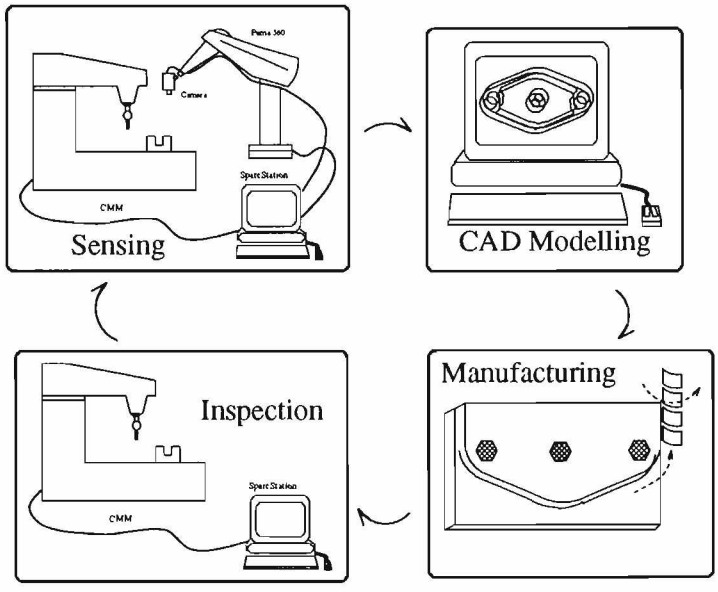


Figure 15: Closed loop system for reverse engineering

movements and structures, interaction between the touching probe and part, and occlusions. A DEDS observer serves as an intelligent sensing module that utilizes existing information about the tasks and the environment to make informed tracking and correction movements and autonomous decisions regarding the state of the system.

To be able to determine the current state of the system we need to observe the sequence of events occurring in the system and make decisions regarding the state of the automaton. State ambiguities are allowed to occur, however, they are required to be resolvable after a bounded interval of events. In a *strongly output stabilizable* system, the state of the system is known at bounded intervals and allowable events can be controlled (enabled or disabled) in a way that ensures return in a bounded interval to one of a desired and known set of states.

One of the objectives is to make the system strongly output stabilizable and/or construct an observer to satisfy specific task-oriented visual requirements. Many 2-D visual cues for estimating 3-D world behavior can be used. Examples include: image motion, shadows, color and boundary information. The uncertainty in the sensor acquisition procedure and in the image processing mechanisms are taken into consideration to compute the world uncertainty.

4.2 Experiments

In conducting our experiments, we use a B/W CCD camera mounted on a Puma 560 robot arm, that observe and guide the interaction between the CMM probe and the machined part (see Figure 16.) In order for the state machine to provide control, it must be aware of state changes in the system. As inspection takes place, the camera supplies images that are interpreted by a set of 2D and 3D vision processing algorithms and used to drive the DRFSM. These algorithms are described in greater detail in other publications [30, 25, 27, 29, 26, 28], but include thresholding, edge detection, region growing, stereo vision, etc. The robot arm is used to position the camera in the workplace and move in the case of occlusion problems.

The object of these experiments was to test the operation of the visual system with the state

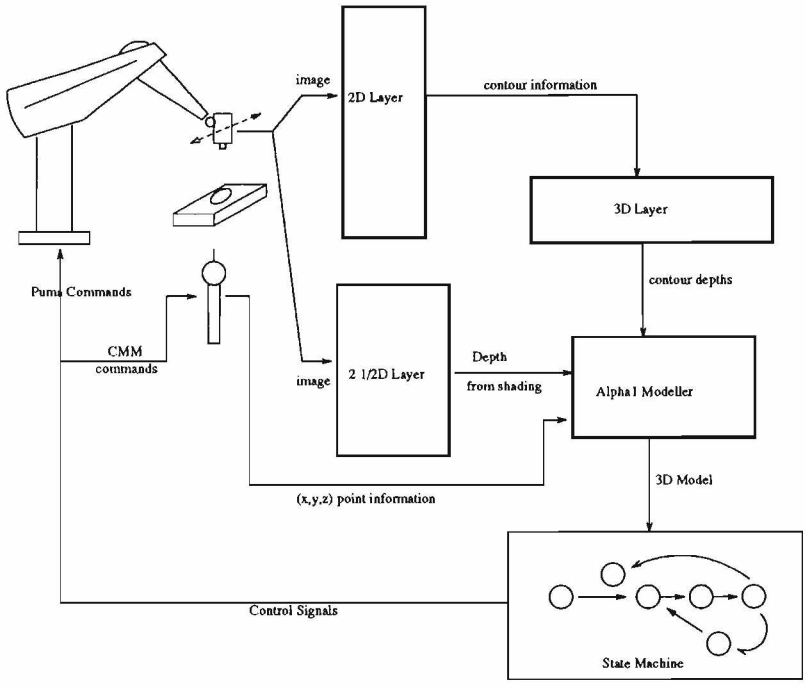


Figure 16: Inspection system overview

machine. Two facets of this were the generation of an initial model from stereo vision and the generation of events that describe a probe's relationship to features in that model.
This stereo process used the Puma arm to gather pairs of images. The resulting model was used to determine feature relationships used in the DEDS controller. The models shown are from this initial visual inspection.
The event generation method, consisting of 2-d image processing routines, was used to detect the relationship of a simulated (hand-held) CMM probe to the features in the initial model. These events were processed by the controller, which output text messages guiding the experimenter to move the probe or indicate that a touch had occurred.
The automaton used in the environment is shown in Figure /reffig:gijoe. This machine has the following states:

- **A:** The initial state, waiting for the probe to appear.
- **B:** The probe appears, and waiting for it to be close. Here, "close" is a measure of the distance between the probe and the current feature, since it depends on the level of the recursive structure. For example, the distance at the first level, which represents the outer contours or features, is larger than that of the lower levels.
- **C:** Probe is close, but not on feature.
- **D:** The probe appears to be on feature in the image, and waiting for physical touch indicated from the CMM machine.
- **E:** Physical touch has happened (and the CMM measurements for the feature parameters are recorded and saved for updating the CAD model.) If the current feature represents a closed

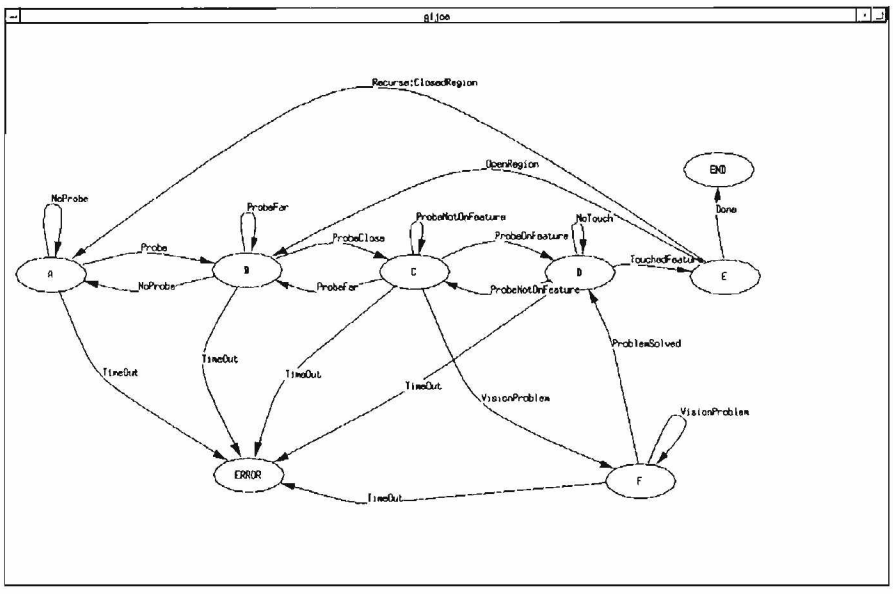


Figure 17: Inspection Environment Window

region, the machine goes one level deeper to get the inner features by a recursive call to the initial state after changing the variable transition parameters. If the current feature was an open region, then the machine finds any other features in the same level.

- **F:** This state is to solve any vision problem happens during the experiment. For example, if the probe is occluding one of the features, then the camera position can be changed to solve this problem.

- **ERROR:** There is a time limit for each part of this experiment. If for any reason, one of the modules doesn't finish in time, the machine will go to this state, which will report the error and terminate the experiment.

4.2.1 Experimental results, Automated Bracket Inspection

A metal bracket was used in the experiment to test the inspection automaton. The piece was placed on the inspection table within view of the camera (see Figure 18).
The machine was brought on line and execution begun in State A, the start state. After initiating the inspection process, the DRFSM transitioned through states until the probe reached the bracket boundary. The state machine then called for the closed region to be recursively inspected until finally, the hole was explored and the machine exited cleanly. The sequence is shown in Figure 21. The original part and the resulting reverse-engineered part are shown in Figures 19 (wireframes) and 20 (rendered images). Notice that the two side holes and a portion of the bracket were not sensed correctly, as a simple strategy was used to sense from only one direction. In the next experiment, a more complicated model is sensed with a more sophisticated sensing and modeling strategy.

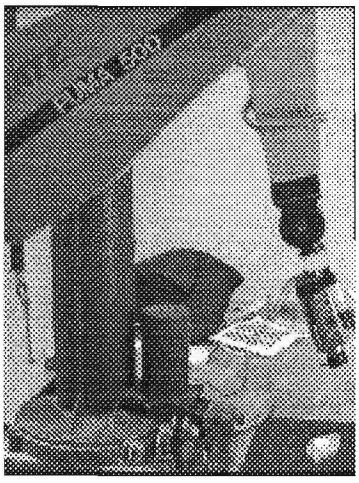


Figure 18: Experimental Setup

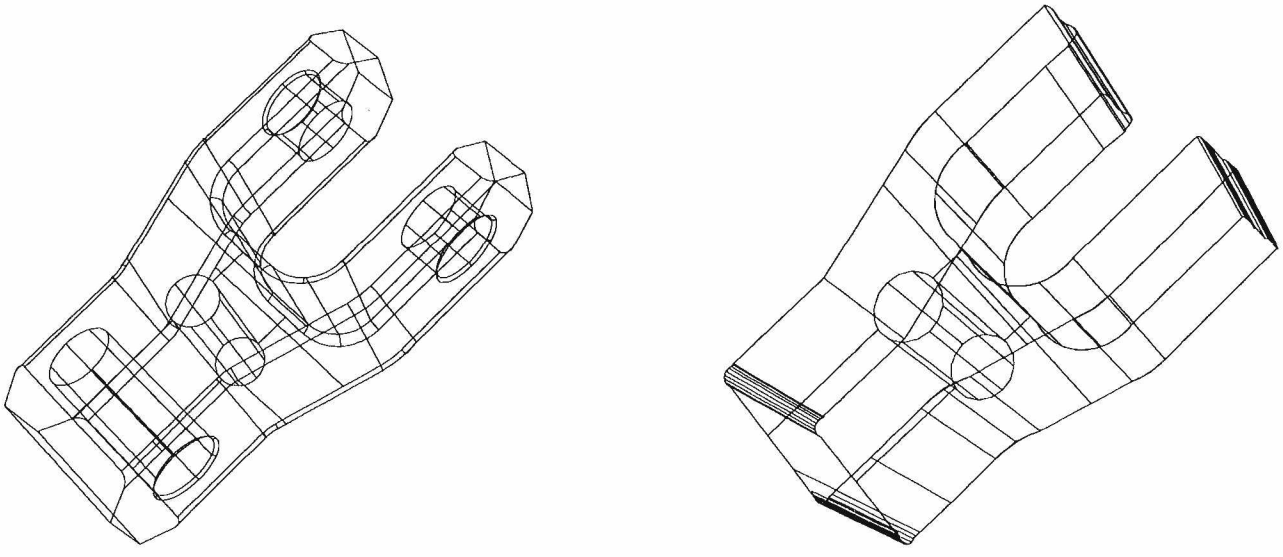


Figure 19: Original and Reverse-Engineered part models

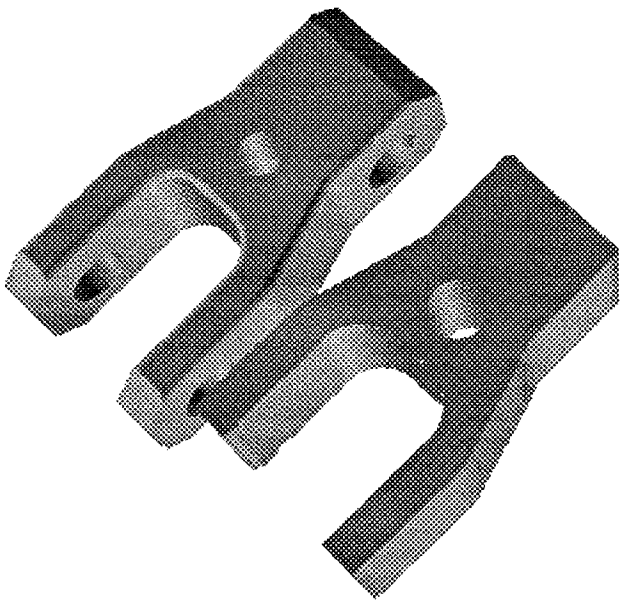


Figure 20: Original and reproduction

4.2.2 Experimental Results, Cover Plate

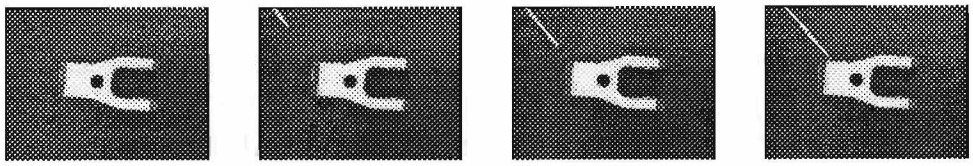
A second experiment was run in a similar fashion, using a part similar to the fuel pump cover from a Chevrolet engine. This piece offers interesting features and has a complex recursive structure which allowed us to test the recursive nature of the state machine.
The sensing strategy used here was more robust than in the previous experiment. Detected feature contours were sensed with stereo vision and used to build up a feature-based α_1 model. This model was then used to semi-automatically machine a reproduction of the part. The original and reverse-engineered wireframe models are shown in Figures 22. A photograph of the original and reproduction is shown in 23. For more detail on the sensing strategy, please see [30].
The inspection sequence corresponding to this experiment is shown in Figure 24. Shown there, the DRFSM transitions correctly through the inspection of the outside profile (depth of recursion=0), a hole (1), a profile pocket (1), a hole (2), and another hole (1).

5 Conclusions

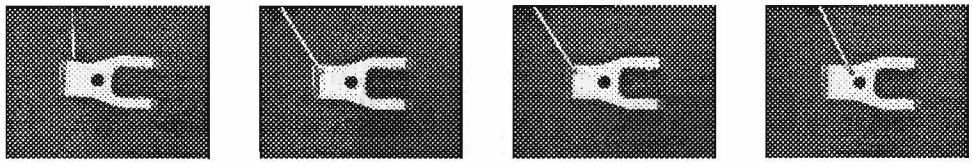
A software environment system was developed for simulating, analyzing, synthesizing, monitoring, and controlling complex discrete event and hybrid systems. We have also presented two problems related to robotics and automation for which discrete event and hybrid systems formulation play a significant role in the solution.

References

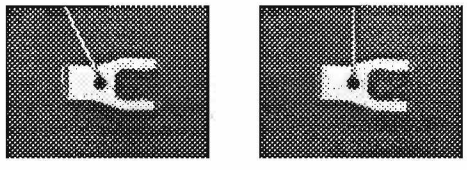
[1] J. Aloimonos and A. Bandyopadhyay, "Active Vision". In *Proceedings of the* 1^{st} *International*



State A: NoProbe State B: ProbeFar State C: ProbeClose State D: ProbeOnFeature



State E: TouchedFeature State A: NoProbe State B: ProbeFar State C: ProbeClose



State D: ProbeOnFeature State E: TouchedFeature

Figure 21: Bracket Sequence

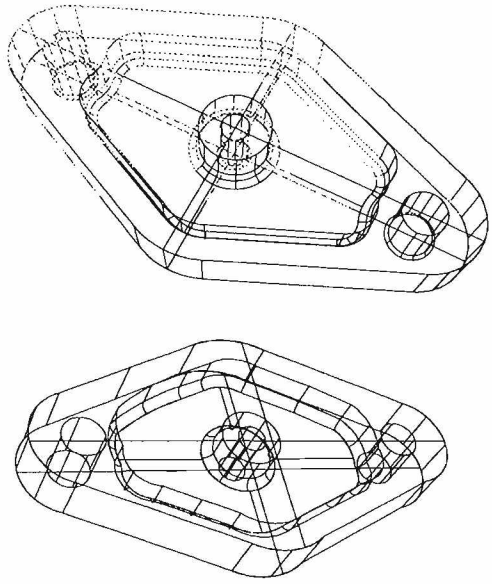


Figure 22: Original and Vision-Reverse Eng'd Models

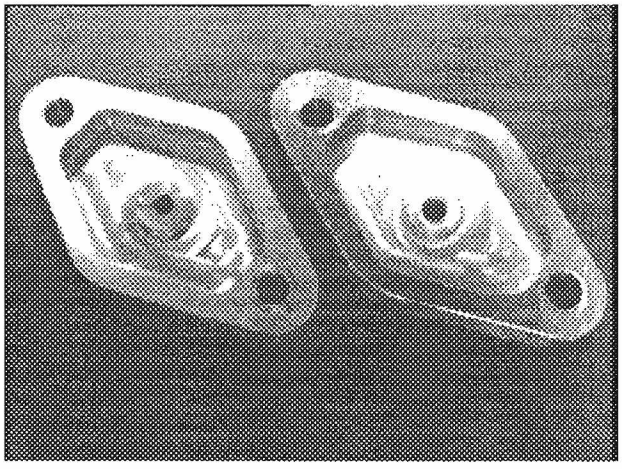
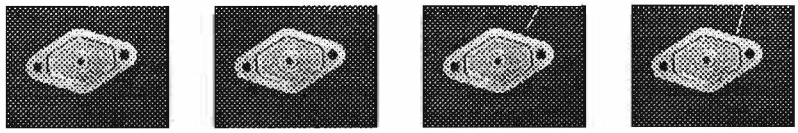
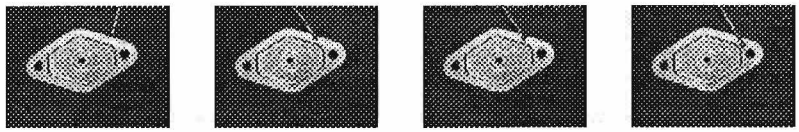


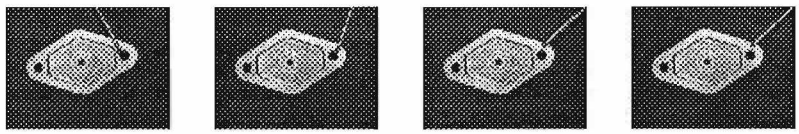
Figure 23: Original and Vision-Reverse Eng'd Parts



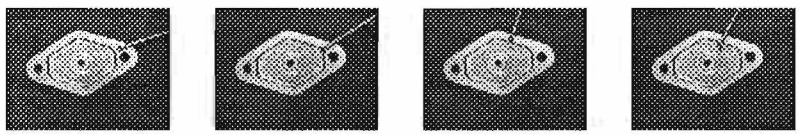
State A: NoProbe State B: ProbeFar State C: ProbeClose State D: ProbeOnFeature



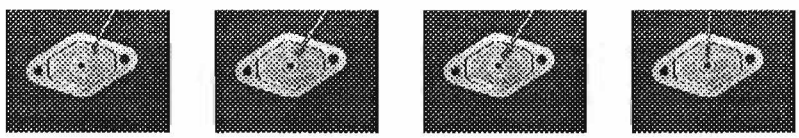
State E: TouchedFeature State A: NoProbe State B: ProbeFar State C: ProbeClose



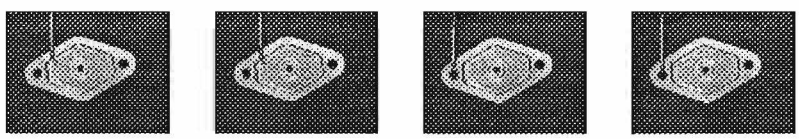
State D: ProbeOnFeature State E: TouchedFeature State A: NoProbe State B: ProbeFar



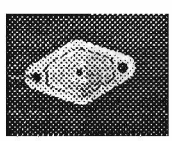
State C: ProbeClose State D: ProbeOnFeature State E: TouchedFeature State A: NoProbe



State B: ProbeFar State C: ProbeClose State D: ProbeOnFeature State E: TouchedFeature



State A: NoProbe State B: ProbeFar State C: ProbeClose State D: ProbeOnFeature



State E: TouchedFeature

Figure 24: Cover Sequence

Conference on Computer Vision, 1987.

[2] P. Anandan, "A Unified Perspective on Computational Techniques for the Measurement of Visual Motion". In *Proceedings of the* 1^{st} *International Conference on Computer Vision*, 1987.

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

[4] R. Bajcsy and T. M. Sobh, *A Framework for Observing a Manipulation Process.* Technical Report MS-CIS-90-34 and GRASP Lab. TR 216, University of Pennsylvania, June 1990.

[5] R. Bajcsy and T. M. Sobh, *Observing a Moving Agent.* Technical Report MS-CIS-91-01 and GRASP Lab. TR 247, Computer Science Dept., School of Engineering and Applied Science, University of Pennsylvania, January 1991.

[6] J. L. Barron, A. D. Jepson and J. K. Tsotsos, "The Feasibility of Motion and Structure from Noisy Time-Varying Image Velocity Information", *International Journal of Computer Vision*, December 1990.

[7] P. J. Burt, et al., "Object Tracking with a Moving Camera", *IEEE Workshop on Visual Motion*, March 1989.

[8] 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.

[9] CHEN, Y., AND MEDIONI, G. Object modelling by registration of multiple range images. *International Journal of Image and Vision Computing 10*, 3 (Apr. 1992), 145–155.

[10] CHEN, Y., AND MEDIONI, G. Integrating multiple range images using triangulation. In *Image Understanding Workshop* (April 1993), Defense Advanced Research Projects Agency, Software and Intelligent Systems Office, pp. 951–958.

[11] J. Hervé, P. Cucka and R. Sharma, "Qualitative Visual Control of a Robot Manipulator". In *Proceedings of the DARPA Image Understanding Workshop*, September 1990.

[12] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. Surface reconstruction from unorganized points. In *Computer Graphics, SIGGRAPH '92* (July 1992), vol. 26.

[13] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. Mesh optimization. In *Computer Graphics, SIGGRAPH '93* (Aug. 1993), vol. 27.

[14] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. Piecewise smooth surface reconstruction. In *Computer Graphics, SIGGRAPH '94* (1994). (to be published).

[15] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow", *Artificial Intelligence*, vol. 17, 1981, pp. 185-203.

[16] HSIEH, Y. C. Reconstruction of sculptured surfaces using coordinate measuring machines. Master's thesis, Mechanical Engineering Department, University of Utah, June 1993.

[17] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Teaching by showing : Generating robot programs by visual observation of human performance", 20^{th} ISIR, 1989.

[18] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Design and implementation of a system that generates assembly programs from visual recognition of human action sequences", IROS, 1990.

[19] Y. Li and W. M. Wonham, "Controllability and Observability in the State-Feedback Control of Discrete-Event Systems", *Proc.* 27^{th} *Conf. on Decision and Control*, 1988.

[20] H. C. Longuet-Higgins and K. Prazdny, *The interpretation of a moving Retinal Image*, Proc. Royal Society of London B, 208, 385-397.

[21] MOTAVALLI, S., AND BIDANDA, B. A part image reconstruction system for reverse engineering of design modifications. *J. Manufacturing Systems 10*, 5 (1991), 383–395.

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

[23] P. J. Ramadge and W. M. Wonham, "Modular Feedback Logic for Discrete Event Systems", *SIAM Journal of Control and Optimization*, September 1987.

[24] SARKAR, B., AND MENQ, C. Smooth-surface approximation and reverse engineering. *Computer Aided Design 23*, 9 (November 1991), 623–628.

[25] SOBH, T., JAYNES, C., DEKHIL, M., AND HENDERSON, T. *Intelligent Systems: Safety, Reliability.* Springer-Verlag, Berlin, 1993, ch. Automated Inspection and Reverse Engineering, pp. 95–122.

[26] SOBH, T. M., DEKHIL, M., JAYNES, C., AND HENDERSON, T. A perception framework for inspection and reverse engineering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '93)* (June 1993). New York City.

[27] SOBH, T. M., DEKHIL, M., AND OWEN, J. C. Discrete event control for inspection and reverse engineering. In *IEEE International Conference on Robotics and Automation* (May 1994). San Diego.

[28] SOBH, T. M., JAYNES, C., AND HENDERSON, T. A discrete event framework for intelligent inspection. In *IEEE International Conference on Robotics and Automation* (May 1993). Atlanta.

[29] SOBH, T. M., OWEN, J. C., DEKHIL, M., JAYNES, C., AND HENDERSON, T. Industrial inspection and reverse engineering. In *IEEE 2nd CAD-Based Vision Workshop* (February 1994). Pittsburgh.

[30] SOBH, T. M., OWEN, J. C., JAYNES, C., DEKHIL, M., AND HENDERSON, T. C. Active inspection and reverse engineering. Tech. Rep. UUCS-93-007, Department of Computer Science, University of Utah, March 1993.

[31] T. M. Sobh and K. Wohn, "Recovery of 3-D Motion and Structure by Temporal Fusion". In *Proceedings of the* 2^{nd} *SPIE Conference on Sensor Fusion*, November 1989.

[32] M. Subbarao and A. M. Waxman, *On The Uniqueness of Image Flow Solutions for Planar Surfaces in Motion*, CAR–TR–113, Center for Automation Research, University of Maryland, April 1985.

[33] S. Ullman, "Analysis of Visual Motion by Biological and Computer Systems", *IEEE Computer*, August 1981.

[34] S. Ullman, *Maximizing Rigidity: The incremental recovery of 3-D structure from rigid and rubbery motion*, AI Memo 721, MIT AI lab. 1983.

[35] VAN THIEL, M. Feature based automated part inspection. Master's thesis, University of Utah, 1993.