8.7 Hybrid Systems and Control

Tarek M. Sobh
Computer Science Department
University of Utah

In this section we present an overview for the development of 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 illustrate the concept by an example from the robotics and automation domain. The application discussed is for formulating an observer for manipulating agents.

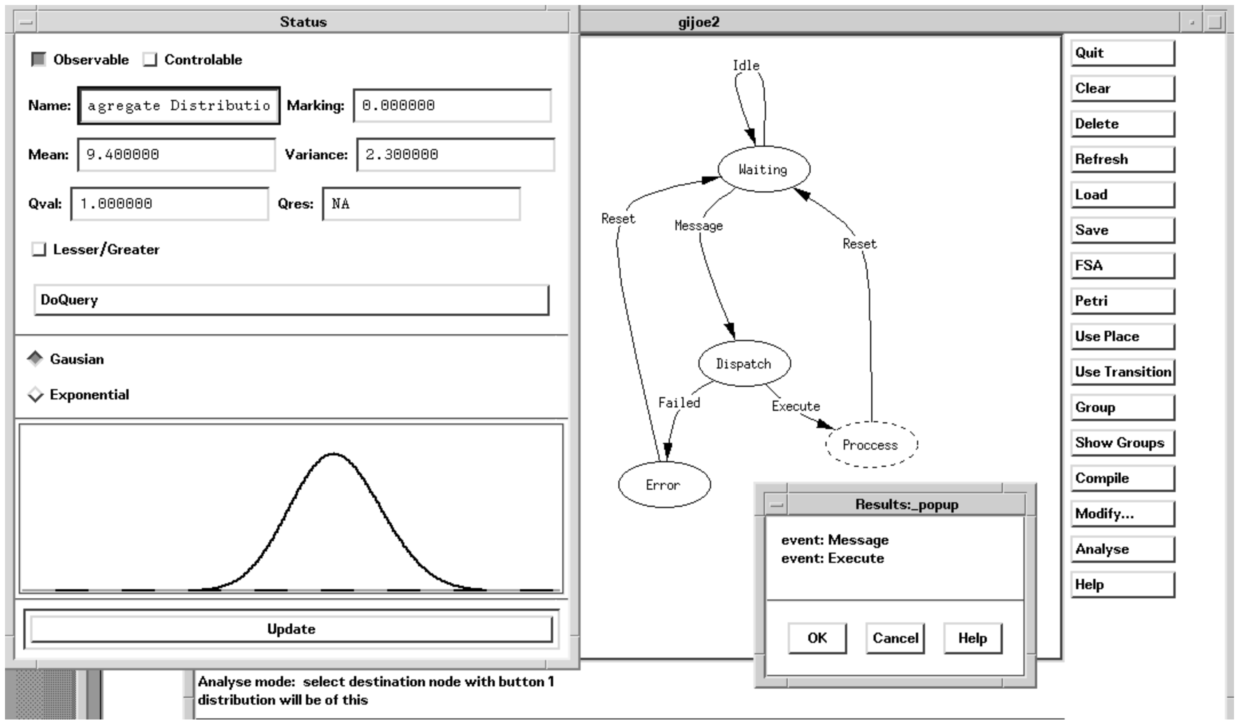
8.7.1 Introduction

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, 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,



[✓ Observable] Controlable

Name: agregate Distributio Marking: 0.000000

Mean: 9.400000 Variance: 2.300000

Qval: 1.000000 Qres: NA

Lesser/Greater

[✓ Gausian] Exponential

event: Message
event: Execute

Analyse mode: select destination node with button 1
distribution will be of this

Figure 8.1: A stochastically timed FSM window during analysis

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.

There is a continuous need for environments to support the development of hybrid systems. A number of tools and graphical environments for simulating, analyzing, synthesizing, monitoring, and controlling complex discrete event and hybrid systems have been developed. A snap shot of one such environment is shown in Figure 8.1.
As an example, this software environment aids 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. It also 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. Next, we proceed to introduce the concept of hybrid systems and control through an example application within robotics and automation.

8.7.2 Discrete Event and Hybrid Observation Under Uncertainty

We discuss a representation for the general problem of observation through a discrete event and hybrid system framework. The system being studied can be considered as a "hybrid" one. This is 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 words, 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 [2,7,10,19]; recovering 3-D parameters for different kinds of surfaces [6,14,17,18]; and also in the context of other problems [1,3,8,9]. 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. Those interpretations should attach meaning to the 3-D world events, as opposed to the simple recovery of 3-D parameters and the consequent tracking movements to compensate their variation over time.

In this work we establish a hybrid system framework for the general problem of observation, recognition, and understanding of dynamic visual systems. The framework 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 utilizes all existing knowledge about the system and the anticipated actions in order to solve the observer problem. The resulting observer is efficient, stable, and practical. 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. The work leads 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. We then 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.

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 are many issues that need to be resolved, among them: definitions for observability, stability, stabilizability, and 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 the *continuous* 2-D and 3-D evolution of parameters. In addition, 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 general problem of defining, monitoring, or controlling such hybrid systems in general. What we intend to present in this work is a suitable framework for the class of hybrid systems encountered within the robotic observation paradigm. The representation we advocate allows for the symbolic, 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 employing a representation of discrete event dynamic systems, which is augmented by the use of a concrete definition for events. We also implement 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 [13,15,16]. 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 a DEDS model as a high-level structuring technique for a system to observe a robot hand manipulating objects.

Discrete event dynamic systems for active visual sensing An example of a high-level DEDS controller for part inspection can be seen in Figure 8.2. 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. The module is capable of making decisions about the *visual state* of the manipulation process taking place. This module also provides both symbolic and parametric descriptions which can be used to observe the process *intelligently* and *actively.*

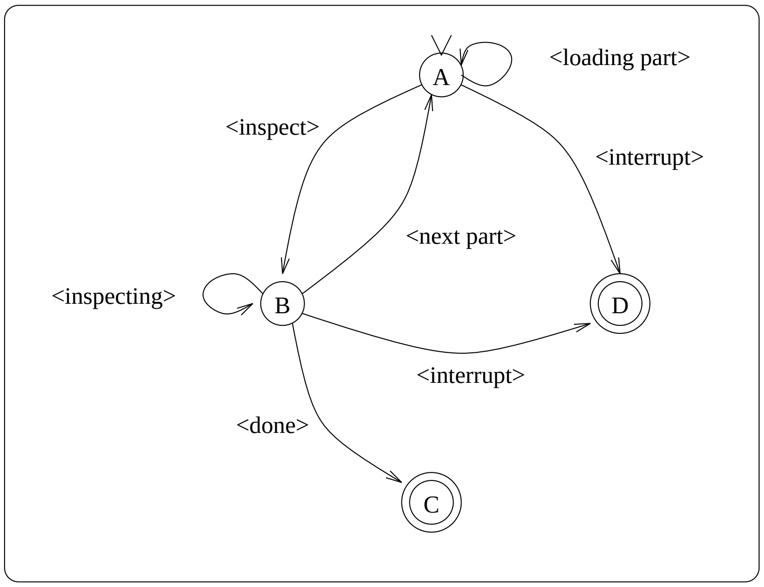


Figure 8.2: A Simple FSM

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 known information about a manipulation process. 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. It utilizes existing information about the tasks and the environment to make informed tracking movements and autonomous decisions regarding the state of the system.

For determining the current state of the system, the sequence of events ahould be observed. A decision should also be made 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. Allowable events can also be controlled (enabled or disabled) in a way that ensures the return - in a bounded time 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 while computing 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.

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. Those states 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 definitions 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 requirement for a visual observer. The observer 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. The actions which the observer robot needs to perform depends on the sequence of "observable" events and the reconstructed state path.
Event descriptions in a visual observer are possibly a combination of different 2-D and 3-D visual cues. 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 a classical 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 facilitate the transition from a DEDS skeleton to 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. Those control sequences are activated at intermittent points in time in order to "guide" the observer to the "desirable" set of visual states.

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 known information about manipulation tasks. Furthermore, we also use all existing knowledge about the physical limitations of both the observer and manipulating robots. The high-level state definition permits the observer to 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. The motion, shape, and structure of the different objects should also be recovered in 3-D. This process should be done in real time while the task is being performed. However, this formulation is inefficient, unnecessary, and for all practical purposes infeasible to compute in real time. In addition, the 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 attempting a different formulation. We view 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.

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. Those models 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, depending on the application requirements. The code for the state machine then needs to be implemented. Automatic construction of the state machine could be done by having a *learning* stage [11,12], in which a mapping module would form the automaton. The building phase 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 will depend 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 code, 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 the sections that deal with the different uncertainty levels and event identification mechanisms.

Building the Model The ultimate goal of the observation mechanism is to know at all (or most) of the time the current manipulation process. The fact that the observer will have

to move, makes one think of the stabilizability principle for general DEDS, as a model for the tracking technique 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 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. The events cause state transitions within the manipulation process. Assuming, for the time being, that we do not have 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. Those actions can alter the visual configuration of the manipulation process (with respect to the observer's camera). Furthermore, we can define a set of "good" states, where the visual configuration of the manipulation process enables the camera to keep track of, 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. Consequently, it can perform the 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 8.3, 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. An example of this would be 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 it. The

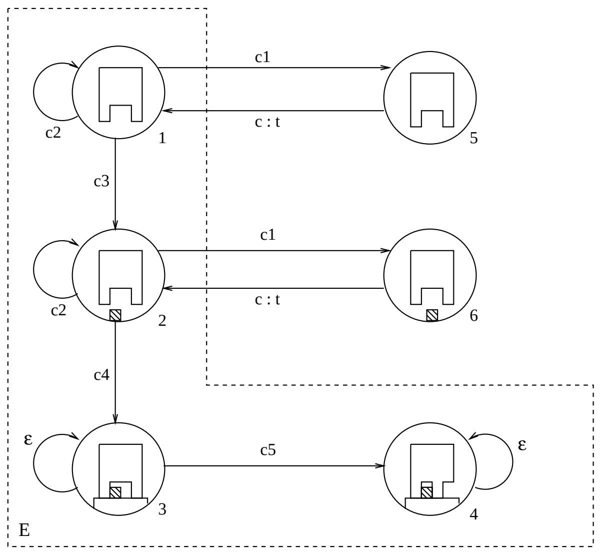


Figure 8.3: A Model for A Grasping Task

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. These states would occur when 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, because 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. Other events represent the appearance of the objects into the viewed scene. The transition from state 1 to state 2 is caused by the appearance of an object. The transition from state 2 to state 3 is caused by the hand enclosing an object. 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 (the hand moving 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 orientations.
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. Search states could be added in order to "look" for the hand

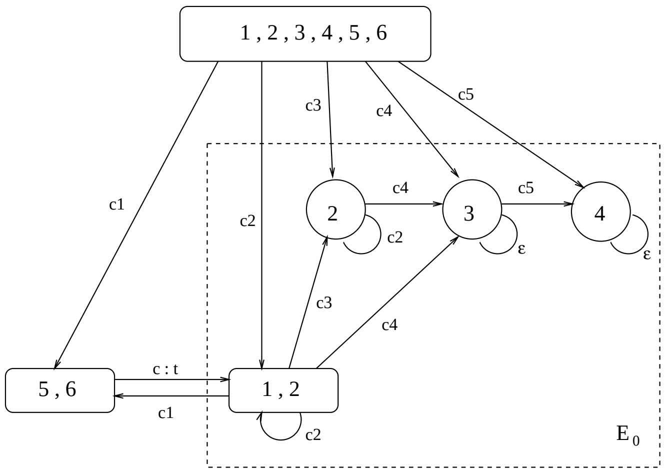


Figure 8.4: An Observer for the Grasping System

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. Furthermore, the model will enable using the sequence of events and control to "guide" the observer into the set of good states E, and thus stabilizing 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.

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. Decisions must be made 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. It should be noticed that events can be asserted with a specific probability (as will be described in the sections to come). Therefore, 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 8.4. 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 start, the state of the system is totally ambiguous. 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

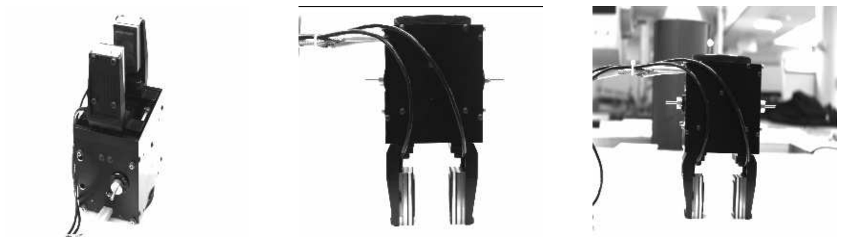


Figure 8.5: Different Views of the Lord Gripper

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 of the system.
In the next section we shall elaborate on defining the different events in the visual manipulation system. We also discuss different techniques for event and state identification. A framework for computing the event uncertainty will be introduced.

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 8.5. 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 8.6. The sequence is defined by our model, and the visual states correspond to the gripper movements 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 8.7. The automaton encodes an observer which tracks the hand by keeping a fixed geometric relationship between the observer's camera and the hand, as 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. 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.

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. Event 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

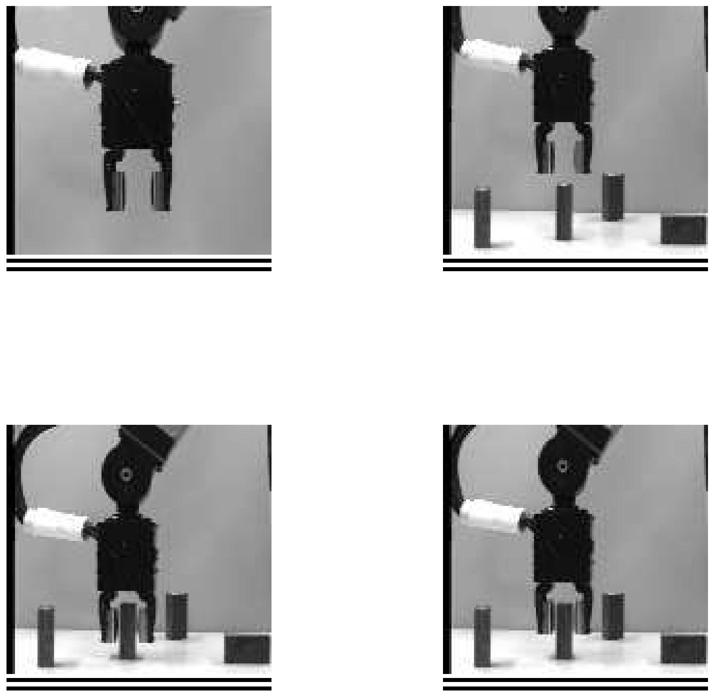


Figure 8.6: **A Grasping Task :** As seen by the observer's camera

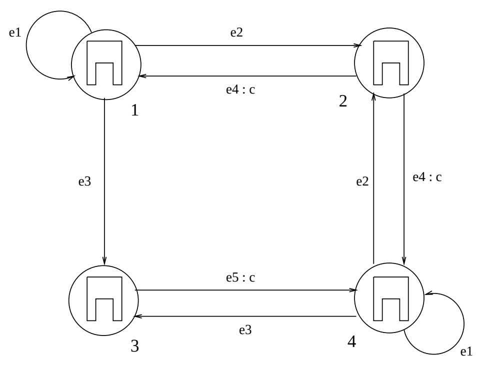


Figure 8.7: A Model for a Simple Visual Sequence

such that :

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

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 probability distribution curves for the various 3-D motion parameters. State transitions are 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.

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 the 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 8.8. 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 [14] :

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

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

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 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 [4,5,17].

Modeling and Recovering 3-D Uncertainties

The uncertainty in the recovered image flow values results from sensor uncertainties, noise, and 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

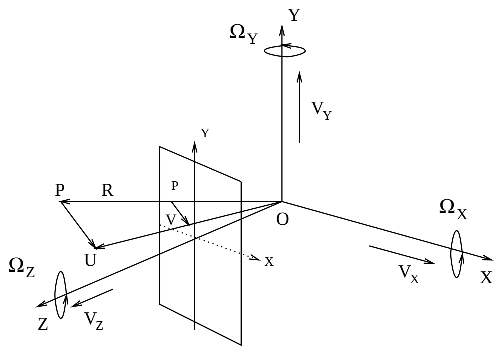


Figure 8.8: 3-D Formulation for Stationary Scene/Moving Viewer

strategy used to find the 2-D uncertainty in the features 2-D representation utilizes the recovered camera parameters, and the 3-D world coordinates (x_w, y_w, z_w) of a known set of points. The corresponding pixel coordinates are then computed, for points distributed throughout the image plane a number of times. We then find the actual feature pixel coordinates and construct 2-D histograms for the displacements from the recovered coordinates. 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 is modeled by using synthesized images and corrupting them. We then apply the feature extraction mechanism to both kinds of images and compute 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. These 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 [4,5,6,17,18,20]. 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 8.9. It should be noted that the recovered distributions represents a fairly accurate estimation of the actual 3-D motion.

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

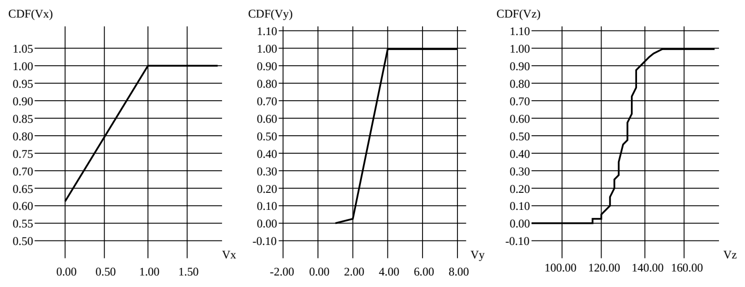


Figure 8.9: Cumulative Density Functions of the Translational Velocity

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 the wrong automata state. The remedy to such problems can be implemented through time proximity, that is, wait for a while (which is to be preset) till a strong probability value is registered. Another technique is to *backtrack* in the observer automaton model 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. The stack includes a sorted list of the computed event probabilities, and a father-state variable.

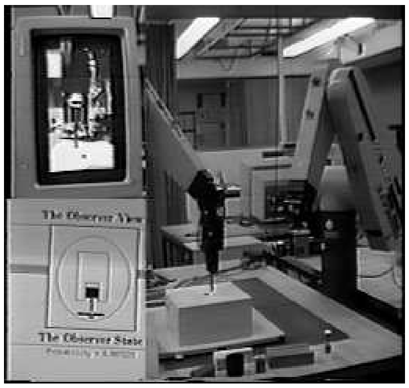
Experiments

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.

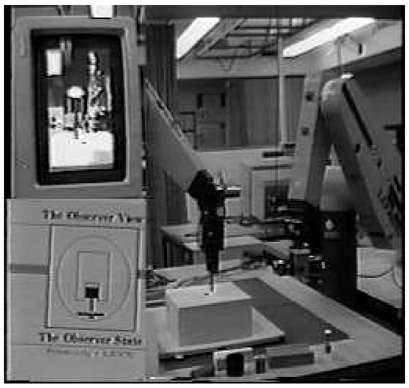
The experiments were shot with three video camera. The right hand side of the images shows the actual observer and manipulation workspaces, and the different configurations as the experiment proceeds. The upper left corner shows the observer view, which is the set of images grabbed by the observer camera 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 8.10 and 8.11 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.

Conclusions

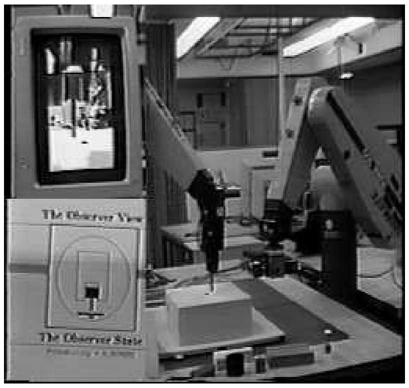
We described a system for observing a manipulation process. The proposed approach can be generalized for other hybrid systems involving different kinds of quantization requirements. The use of discrete event dynamic systems with uncertainty modeling enables the observer



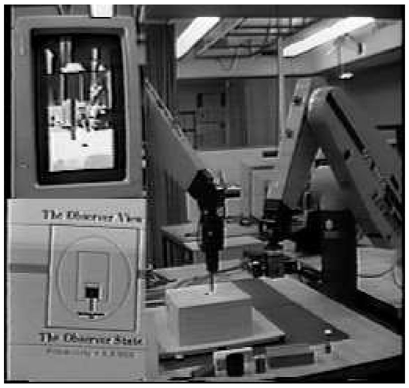
(1)



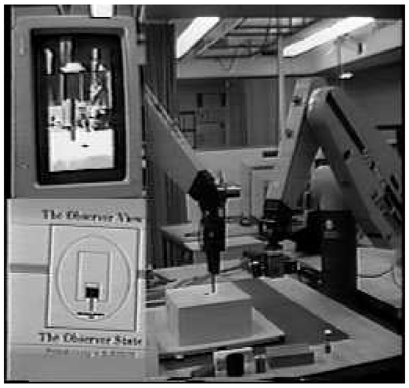
(2)



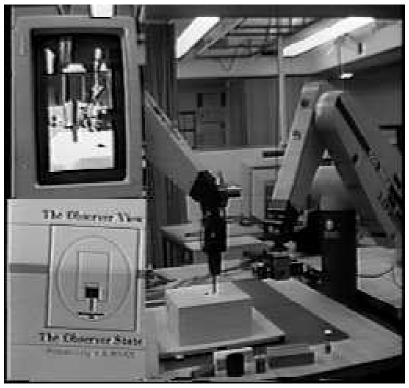
(3)



(4)

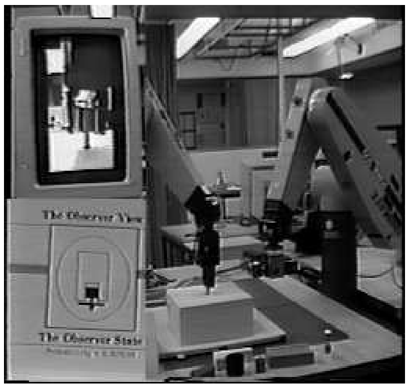


(5)

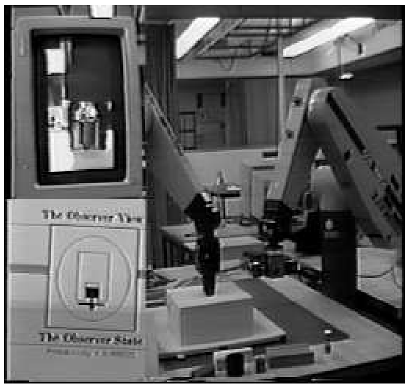


(6)

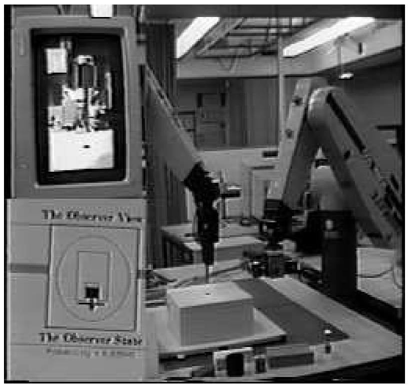
Figure 8.10: Observer State and View (1)



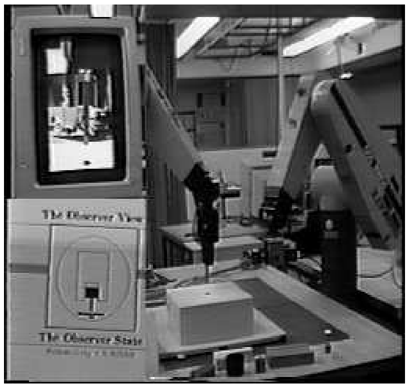
(7)



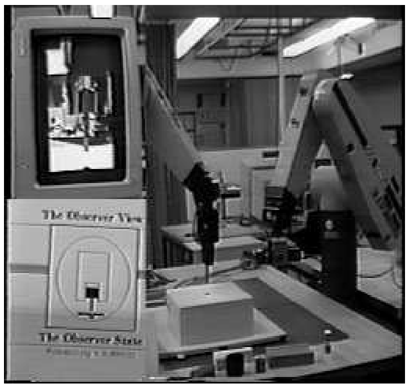
(8)



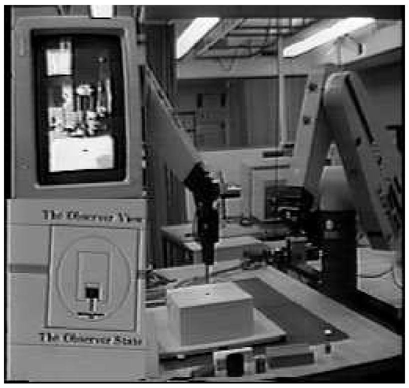
(9)



(10)



(11)



(12)

Figure 8.11: Observer State and View (2)

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 DEDS as a high-level model for the evolution of the visual relationship over time. The proposed formulation can be 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.

8.7.3 Conclusions

The control, analysis, modeling, synthesis, simulation, and monitoring of hybrid and discrete event systems are becoming more and more crucial in the current complex factory floor environments. We have discussed and presented hybrid systems through a problem related to robotics and automation for which discrete event and hybrid systems formulations play a significant role in the solution.

Acknowledgments:

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.

Bibliography

[1] J. Aloimonos and A. Bandyopadhyay, "Active Vision". In *Proceedings of the* 1^{st} *International 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] J. Hervé, P. Cucka and R. Sharma, "Qualitative Visual Control of a Robot Manipulator". In *Proceedings of the DARPA Image Understanding Workshop*, September 1990.

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

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

[12] 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.

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

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

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

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

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

[18] 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.

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

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