Journal of Intelligent and Robotic Systems **16**: 315–385, 1996.
© 1996 *Kluwer Academic Publishers. Printed in the Netherlands.*

A Discrete Event Framework for Autonomous Observation Under Uncertainty

TAREK M. SOBH and RUZENA BAJCSY
169 University Avenue, Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT 06601, U.S.A. e-mail: sobh@cse.bridgeport.edu

(Received: 24 November 1994; in final form: 8 September 1995)

Abstract. In this work we establish a framework for the general problem of observation, which may be applied to different kinds of visual tasks. We construct 'intelligent' high-level control mechanisms for active visual recognition of different processes within a hybrid dynamic system. We address 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 manipulation system. The formulation utilizes the knowledge about the system and the different actions in order to solve the observer problem in an efficient, stable and practical manner. The model uses different tracking mechanisms so that the observer can 'see' the workspace of the manipulating robot. An automaton is developed for the hand/object interaction over time and a stabilizing observer is constructed. Low-level modules are developed for recognizing the visual 'events' that causes state transitions within the dynamic manipulation system in real time. A coarse quantization of the manipulation actions is used in order to attain an active, adaptive and goal-directed sensing mechanism. The formulation provides high-level symbolic interpretations of the scene under observation. The discrete event framework is augmented with mechanisms for recovering the continuous parametric evolution of the scene under observation and for asserting the state of the manipulation agent. This work examines closely the possibilities for errors, mistakes and uncertainties in the manipulation system, observer construction process and event identification mechanisms. We identify and suggest techniques for modeling these uncertainties. Ambiguities are allowed to develop and are resolved after finite time. Error recovery mechanisms are also devised. The computed uncertainties are utilized for navigating the observer automaton state space, asserting state transitions and developing a suitable tracking mechanism.

Key words. Robotics, autonomous observation, observation, uncertainty, discrete event systems, observers, vision.

1. Introduction

Visual observers are the class of agents whose primary function is to provide visual information about the behaviour of the different elements in the scene under observation. There could be many components of such a dynamic scene that evolves over time. Environments under observation could include both dynamic and static agents, whether human or robotic. The definition of an environment could also include some properties of the different objects and/or agents in the scene and how they behave.

The kind of information that the visual observer is supposed to provide differs according to the nature of the *environment*, the structure and capabilities of the different *agents* in the scene and the set of *tasks* (if any) that in under consideration. The observer, ideally, should utilize all the pre-existing knowledge about the environment, agents and the tasks, in addition to the required kind of data it is expected to deliver and its own capabilities, be it in terms of processing capabilities, maneuvers and/or sensing mechanisms in order to achieve a robust and reliable observation mechanism.

The structure and behaviour of visual observers differs wildly from any given domain to another. Clearly, an observer for a manufacturing environment would be different from an observer for appreciating art or finding some material properties. The difference would exhibit itself in the both the kind of information that the observer is supposed to *sense* and *reason* about, possibly given some a priori knowledge, and in the kind of information that it is supposed to provide. The difference manifests itself in the processing capabilities and requirements of observers. For autonomous robotic observers, the difference would be in the kind of visual sensors, software and hardware to process the input sensory data, also in the mechanical capabilities of the robotic observer and in the decision making modules.

In this section, we formalize the problem statement and provide some background. We also define some requirements and justifications and outline our methodology for solving the problem. We then discuss the outline for the paper.

1.1. PROBLEM STATEMENT

The problem statement can be summarized by declaring that there is a need for an 'intelligent observer'. In particular, we need an observer that is able to determine the current state of the scene under observation. We are interested in an *autonomous* robotic observer. The observer should be able to utilize a priori information about the observation domain. The observer should also be an *active* one that is able to alter the parameters of its sensory apparatus in order to determine the evolution of the environment robustly [3, 8].

The intelligent observer should be able to recognize the visual tasks, understand the meaning of the scene evolution and successfully reports on the current visual state. It is obvious that there is a need for high-level interpretation of actions within the environment. The observer should be able to relocate itself adaptively in order to be in a better viewing position. It is important to have guarantees for observability and stability within the viewing mechanism. The framework should be a *predictable* one. The framework to be used should deliver under uncertainty and be able to recover from errors. In fact, we would like to construct an observer that actually *utilizes* uncertainties to assert visual states of the system.

We concentrate on the problem of observing a manipulation process in order to illustrate the ideas and motive behind our framework. The manipulation observer should recognize and report on different aspects of the tasks under observation actively and adaptively. We suggest that the framework we develop could be utilized for other kinds of visual observation domains.

1.2. REQUIREMENTS AND BACKGROUND

The problem of visually observing a moving agent was addressed in the literature. It was discussed in the work addressing tracking of targets and determination of the optic flow [4, 16, 20, 29, 46, 52], recovering 3D parameters of different kinds of surfaces [6, 13, 39, 40, 62, 80, 85, 87, 88], recently in the work addressing the determination and decoupling of visual shape and motion parameters [81, 83, 84] and also in the context of other problems [9, 27, 42, 47, 78].

Recovering the visual parameters of a scene under observation and using them to develop methods for tracking moving agents within a dynamic scene was discussed [21, 29, 41, 42, 59, 96]. 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 3D world events, as opposed to simple recovery of 3D parameters and the consequent tracking movements to compensate their variation over time. Thus, we need some kind of an *intelligent observer* to understand the actions of a dynamic agent. A number of frameworks for defining dynamic systems have been developed during the past few years. The work in [15, 35, 36, 64] addresses a variety of dynamic systems within the general hybrid system domain. The work in [19, 23, 43, 44, 61, 67, 70, 82, 94] addresses the problem within the discrete event paradigm. Other work in different contexts and general domains have been done too [12, 25, 26, 28, 48, 49, 55, 58]. Addressing the problem in the context of visual observation have not been studied. We feel that it is crucial to utilize, augment and fully develop a framework for a real-life and possibly real-time problem like visual observation. Attacking such a problem will help in identifying current deficiencies and requirements in terms of uncertainties, formulation, hardware and software.

We try to capture the scientific essence of understanding the process of observation. We hope to motivate the use of our framework for general modeling purposes for different complex hybrid dynamic systems. 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 establish 'intelligent' high-level control mechanisms for the observer in order to achieve an efficient approach to visually recognizing different processes within a dynamic system. Thus, we develop an observer to satisfy some general requirements as follows:

- Recognize visual tasks and events.

- Reposition itself 'intelligently'.
- Operate in real time.
- Assert and report on *distinct* and discrete visual states.
- Utilize the *continuous* parametric evolution of the visual system.
- Accommodate visual uncertainties.

The process of observing a robot hand manipulating an object is very crucial for many robotic and manufacturing tasks. It is important to know in an automated manufacturing environment whether the robot hand is doing the correct sequence of operations on an object (or more than one object), with the hope of fully or semi-automated correction of such actions. It might be a fact that the workspace of the robotic manipulator cannot be accessed by humans, as in the case of some space applications or some areas within a nuclear plant. In that case, having another robot 'look' at the process is a very good option. Thus, the observation process can be thought of as a stage in a closed-loop fully or semi-automated system where there are robots who perform the required manipulation task and some other robots who observe them and correct their actions when something goes wrong. Typical manipulation processes include grasping, pushing, pulling, lifting, squeezing, screwing and unscrewing. Visual information from the observing robots can be the only kind of feedback, or it can be supplemented by other kinds, like tactile sensing. In this work, we address the problem of observing a single hand manipulating a single object and 'knowing' what the hand is doing, no feedback will be supplied to the manipulating robot to correct its actions. Intelligence gathering in partially structured hostile environments is another possible use for autonomous intelligent observers.

1.3. RESEARCH GOALS AND METHODOLOGY

The objective of this research is to design, as discussed before, a predictable framework for intelligent observation, recognizing visual tasks and events, that allows a precise definition of the notion of observability. The observer should satisfy the requirements that were specified in the previous section. We identify a framework for the visual system state description and mechanisms for recovering the continuous scene evolution. Methods to assert the change of visual state are developed and uncertainty modeling is performed. The suggested system should use existing knowledge about the scene and tasks. We use a divide-and-conquer design approach. First we identify a high-level model for the visually observable discrete manipulation states, then we define low-level action recovery modules.

We use a discrete event dynamic system as a high-level structuring skeleton 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.

To be able to observe how a hand manipulates an object, we must be able to identify how the hand moves and how the hand/object physical relationship evolves over time. One way of doing this would be to identify the motion vectors as seen be the observer. In other words, identify the two-dimensional vectors in the observer's camera plane and use these as a cue to know how the objects under consideration moves in the three-dimensional space. The problems of recovering the image flow vectors (the two-dimensional motion vectors in the camera plane), and identifying the scene structure and motion have been key problems in computer vision. Many techniques have been developed for estimating the image flow [4, 17, 52, 91, 96], and to recover the three-dimensional world structure/shape and motion [13, 34, 63, 78, 92, 93]. Those techniques are not problem-oriented, they are not restricted to a particular problem domain, as is the case with our observer construction problem.

Trying to use the above techniques directly to solve our observer problem is naive and inefficient. In fact, possibly not feasible to perform in a practical way using the current technology, as the complexity of the manipulation process increases. Due to the fact that we probably know a priori some information about the allowable (or useful) manipulation processes and the geometry of the robotic hand, posing the problem as a structure (or shape) and motion vision procedure is a very naive and simplistic way of modeling the observer system. It should also be noted that the observer will have to be an *active* one to be able to interact with the manipulation environment in such a way as to be able to 'see' at all times. The idea of an active observer was discussed in the literature [3, 8–10, 75, 76], and it was shown that an active observer can solve basic vision problems in a much more efficient way than a passive one.

The work examines closely the possibilities for errors, mistakes and uncertainties in the visual manipulation system, observer construction process and event identification mechanisms. We divide the problem into a number of major *levels* for developing uncertainty models in the observation process. The *sensor* level models deals with the problems in mapping 3D features to pixel coordinates and the errors incurred in that process. We identify these uncertainties and suggest a framework for modeling them. The next level is the *extraction strategy* level, in which we develop models for the possibility of errors in the low-level image processing modules used for identifying features that are to be used in computing the 2D evolution of the scene under consideration. In the following level, we utilize the geometric and mechanical properties of the hand and/or object to reject unrealistic estimates for 2D movements that might have been obtained from the first two levels.

After having obtained 2D models for the evolution of the hand/object relationship, we transform the 2D uncertainty models into 3D uncertainty models for the structure and motion of the entire scene. The next level uses the equations that govern the 2D to 3D relationship to perform the conversion. We then reject the improbable 3D uncertainty models for motion and structure estimates by using the existing information about the geometric and mechanical properties of the moving components in the scene. The highest level is the DEDS formulation with uncertainties, in which state transitions and event identification is asserted according to the 3D models of uncertainty that were developed in the previous levels.

1.4. OUTLINE OF THE PAPER

The remainder of this document is organized as follows. We describe the automaton model of a discrete event dynamic system for some hybrid representations in the next section and then proceed in Section 3 to formulate the high-level skeleton framework for modeling the manipulation state space and the observer construction process. In Section 4 we discuss the low-level event identification mechanisms in 2D and 3D and also tracking strategies as controllable events.

We then develop the uncertainty levels and their different models in Section 5. The uncertainty models are constructed for the different event descriptions. The event definition with uncertainty is utilized in Section 6 in order to navigate the observer state space and assert different state transitions. We argue that our representation is a sound approach for modeling hybrid systems and discuss some aspects of our formulation in Section 6. We also discuss a mapping strategy for the automatic generation of visual observers. Experimental results and the description of the hardware and software of the experimental system, which was used to verify the design, are reported in Section 7. The concluding remarks, the contribution and applications of this work, and the future research directions are summarized in Section 8. An appendix discusses a 3D recovery algorithm.

2. Hybrid and Discrete Event Dynamic Systems

Hybrid systems, in which digital and analogue devices and sensors interact over time, is attracting the attention of researchers [15, 35, 49, 55, 64]. 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* 2D and 3D evolution of parameters. Also, there should be a *symbolic* description of the current state of the system, especially in the manipulation domain.

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 systems (DES) 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, 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), Queueing Theory, and Recursive Functions.

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, 61, 67, 68, 70, 90], 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.

2.1. WHAT IS A DISCRETE EVENT DYNAMIC SYSTEM?

An example of a high-level DEDS controller for part inspection can be seen in Figure 1. 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*.

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 3D 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

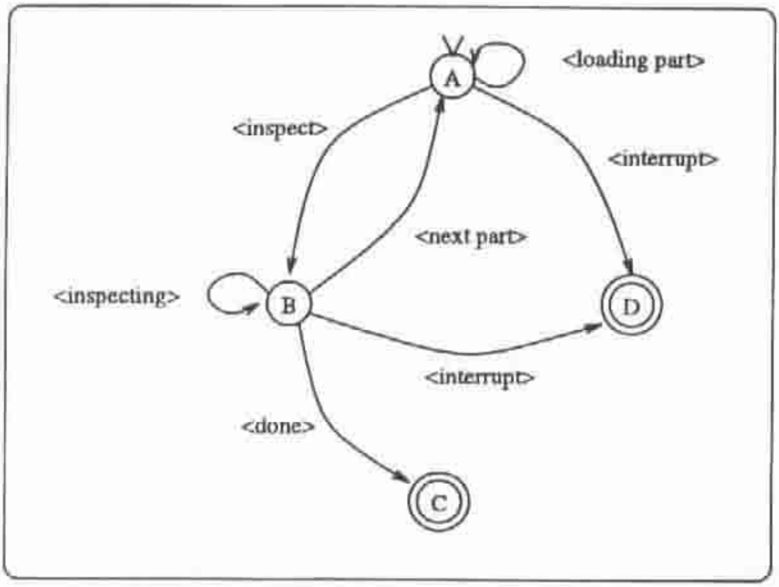


Fig. 1. A simple FSM.

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 2D visual cues for estimating 3D 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.

2.2. 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 2D and 3D 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 uncerttainty 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. 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 3D world motion and structure.

A bare-bone approach to solving the observation problem would have been to try and visually reconstruct the full 3D 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 3D, 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.

3.1. 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 automation 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 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 thorugh 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. A discussion of the mapping module is provided in Section 6.

We next discuss building the m,anipulation 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. 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 techniques 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 2D or 3D 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 sections and will be addressed in Section 6. 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.

3.2.1. 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 2, 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 3D 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 it. The frontal upright

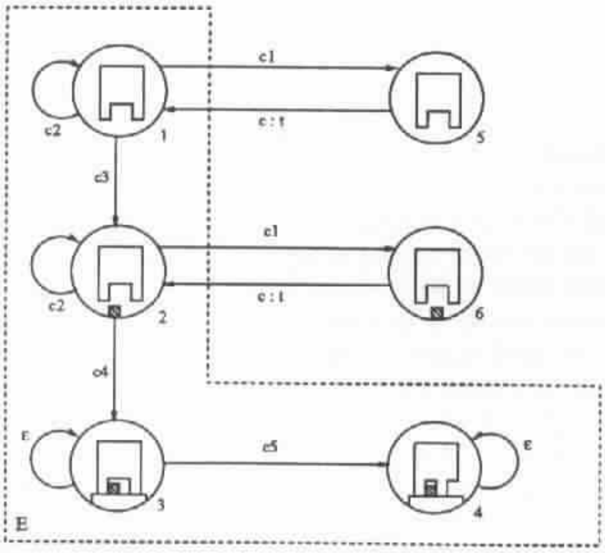


Fig. 2. A model for a grasping task.

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 $X = \{1, 2, 3, 5, 6\}$ 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 discuss 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 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.

3.2.2. A Screwing Task

The next model we present is one for a simple screwing task. The task is that of a gripppper screwing an object (a screw nail for example). It is assumed that the claws of the gripper already encloses the nail and that contact is maintained throughout theprocess, the rotation is allowed to be either clockwise or anticlockwise.

As shown in Figure 3 the model represents a frontal view of the hand at state 1, with the object between the claws, the hand starts to rotate at state 2 and 3 with some view of the claws and the object still in sight and the claws are occluded at state 4 which represents a side view of the gripper. This specific visual representation was chosen because of the fact that transitions between states 1 and 3 and the self loop at 3 cannot be compensated by a tracking action due to the physical limitations of the tracking arm, in other words, the observing robot might not be able to do 360 degrees rotations around the manipulating hand, especially if the workspaces of both robots do not intersect and both are fixed, non-mobile robots. As mentioned before, the frontal upright view with respect to the camera imaging plane in state one was chosen only to facilitate drawing the automaton. The hand can assume any 3D orientation as so long as the claws in states 1, 2 and 3 are within sight of the observer, for example, in the case of screwing a nail into a tilted wall. As shown by our model, the automaton tends to keep the frontal view of the hand as long as possible (as far as the observer robot can rotate), after that the observer will just have to sit idle until rotation of the hand is trackable again. If one define the stable visual state as state 1, then, obviously, the system cannot be made stable with respect to that state, however, one can think of a screwing action on the whole as a stable set, since the robot hand is always within sight of the observer and it does not tend to escape the viewing field. In that case the set of 'good' states E is the same as the set $X = \{1, 2, 3, 4\}$, the finite set of states. The goal of the observer in that case would basically be trying to keep a frontal view as long as it can.

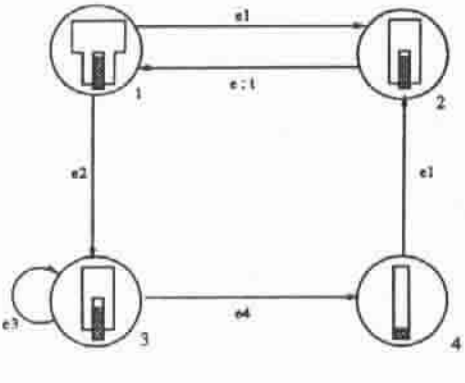


Fig. 3. A model for a screwing task.

The event can be defined as rotations that the observer robot can track and keep a frontal position of the hand, while e_2 is the one that makes the observable robot reach its 'limit' position where it cannot rotate around the hand in the same direction any longer. The rotations e_3 are the untrackable rotations, which lie beyond the reachable workspace of the observable robot. The event e_4 can be defined as the event that causes the visual scene to be a side view of the gripper.

3.3. 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 4. It can be easily seen that the system can be made stable with respect to the set E_0 as discussed in the previous section.

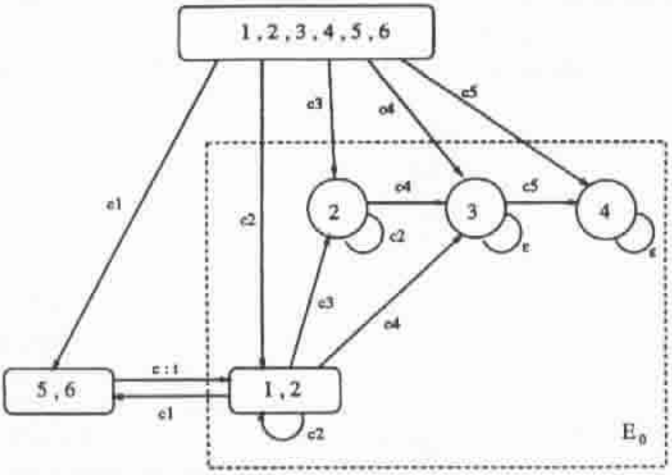


Fig. 4. An observer for the grasping system.

At the beginning, the state of the system is totally ambiguous, however, the observer can be 'guided' to the set E_0 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_0. 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 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.

3.4. 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 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 6. 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 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 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 noncentered 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

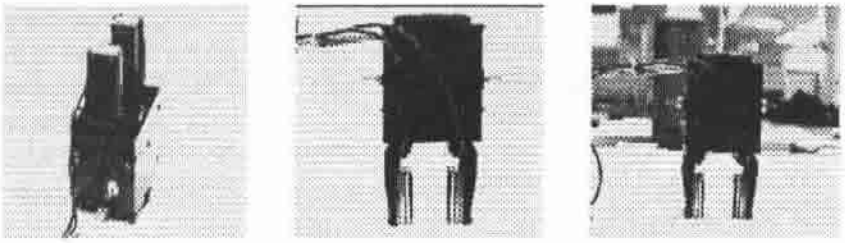


Fig. 5. Different views of the lord gripper.

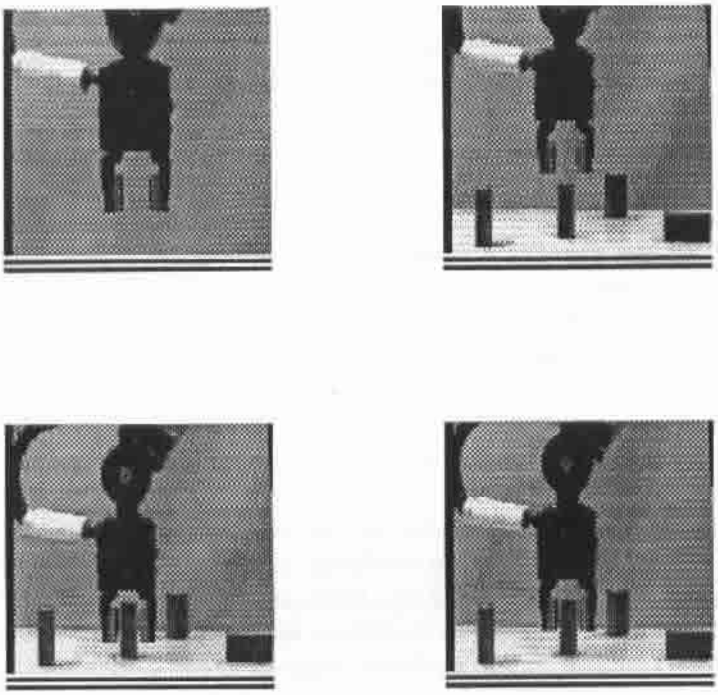


Fig. 6. A grasping task: As seen by the observer's camera.

are controllable tracking events, where e_4 always compensates for e_2 in order to keep a fixed 3D 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, can be defined as any motion $V_Z \geqslant d_z$. Event e_1 is defined as any motion such that

$$-\epsilon_x \leqslant V_X \leqslant \epsilon_x \quad \wedge \quad -\epsilon_y \leqslant V_Y \leqslant \epsilon_y \quad \wedge \quad -\epsilon_z \leqslant V_Z \leqslant \epsilon_z.$$

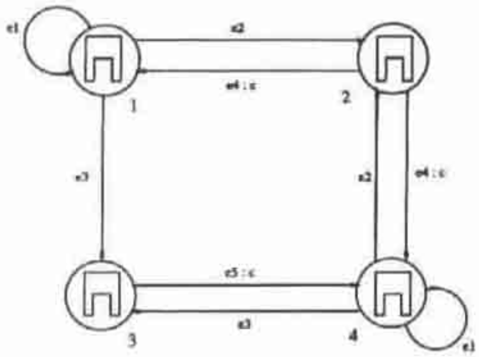


Fig. 7. A model for a simple visual sequence.

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 3D 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.

4. Event Identification and Recovery

In this section we discuss different techniques for calculating the 'events' that cause state transitions within the DEDS state model. Events are defined to be internal actions that happen in the system. Some of the events are uncontrollable ones that the observer role would be to identify and compute from visual cues, another set of events are controllable events which are precisely actions or decisions that the observer agent will have to undertake. The controllable actions are performed by the observer in order to attain some *observability* criteria pertaining to the task under supervision. Typical controllable actions would be some tracking or repositioning movements that would place the observer in a 'better' viewing positions. The controllable actions should be ones that guide the observer into the set of 'good' states as defined by the state space of a particular task. The goal is to be able to stabilize the observer agent DEDS automata.

Uncontrollable events are actual 3D actions that are performed by the manipulating robot. The visual observer role is to recover possible 3D cues from 2D data that is provided by the observer agent camera. Using the formulation in the previous section, it can be shown from some examples used in modeling manipulation processes, that uncontrollable events are mostly primitives like specific 3D movements of the manipulating hand and/or events like "there is an object now in view", "true hand has enclosed the object" and so on. The events that are supposed to be identified and recovered at different states of the observer automaton are highly dependent on the *current* state in the observation process. Thus the observer tends to 'look' for specific actions at different instances of time.

We next discuss techniques to be used in identifying the 3D actions of the manipulation hand and/or the object, which are events that are always important to recover in order to enable the observer to navigate in the automaton. The process is started by identifying the manipulating hand and the object (if it exists) within the observer's viewing window. We then proceed to develop an algorithm for detecting two-dimensional motion vectors on the observer's camera plane. Overall motion estimation and different tracking strategies are then developed in order to be able to stabilize the observer in the most efficient way.

4.1. IMAGE MOTION

In order to be able to identify how the manipulating hand is moving within a grasping task, we use the image motion to estimate the hand movement. This task can be accomplished by either feature tracking or by computing the optic flow. Feature tracking seems to be a good option for determining the hand motion, especially since the same hand will probably be used throughout the manipulation process, and if the system is to be ported to another manufacturing environment, then the interface that tracks specific features can be changed while maintaining modularity. On the other hand, determining the full optic flow seems to be essential for computing the object motion, as we might not know in advance any shape or material information about the objects to be manipulated.

Many techniques were developed to estimate the optic flow (the 2D image motion vectors) [4, 46, 52], we propose an algorithm for calculating the image flow and then we discuss a simpler version of the same algorithm for real time detection of the 2D motion vectors. As a start, we can use a simple two-dimensional segmentation scheme in order to identify the hand and the manipulated object within the camera view. All the 'objects' within an image are identified. An object is simply characterized by a region with a space of at least one pixel surrounding it from everywhere (we assume good lightening control and materials which are easy to identify in this work, as complex visual processing is not the focus of this project), thus regions with holes can be easily recognized using this technique. An edge tracer can be used for this purpose. We can make our decision built on the knowledge we have regarding the geometry of the hand and/or the object. As mentioned before, specific features can be identified, for example: the corners, or we can have a piece of paper with specific features

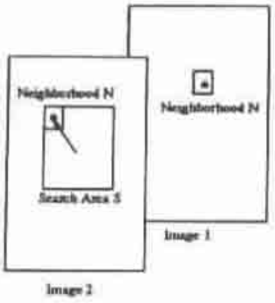


Fig. 8. Identifying the SSD optical flow.

stuck on the hand. It should be noticed that the initial visual sensing techniques used are not sophisticated or novel, the goal is to be able to provide the DEDS visual observer with reasonable sense data.

The image flow detection technique we use is based on the sum-of-squared-differences optic flow. We consider two images, 1 and 2 as shown in Figure 8. For every pixel (x, y) in image 1 we consider a pixel area N surrounding it and search a neighboring area S to seek a corresponding area in image 2 such that the sum of squared differences in the pixel gray levels is minimal as follows

$$SSD(x', y') = \min_{x', y' \in S} \sum_{\Delta x, \Delta y \in N} \left[E(x + \Delta x, y + \Delta y) - E'(x' + \Delta x, y' + \Delta y) \right]^2.$$

The image flow vector of pixel (x, y) then points from the center of N in the first image to the center of the best match in the second image. The search area S should be restricted for practicality measures. In the case of multiple best matches, we can use the one which implies minimum motion, as a heuristic favoring small movements. It should be noted that the accuracy of direction and magnitude of the optic flow determination depends on the sizes of the neighborhoods N and S.

There are three basic problems with this simple approach, one is that the sum of squared differences will be near zero for all directions wherever the graylevel is relatively uniform, the second is that it suffers from the so-called 'aperture problem' even if there is a significant graylevel variation. To illustrate this point, consider a vertical edge moving to the right by one pixel distance, and suppose the N window size is 3×3 pixels and the S window size is 5×5 pixels, the squared-differences at an edge point reaches its maximum for three directions as indicated by the vectors (in pixel displacements); $(1, 0), (1, -1)$ and $(1, 1)$. Figures 9 and 10 illustrates the aperture problem. The third problem is that the scheme will only determine the displacement to pixel accuracy.

We solve the first problem by estimating the motion only at the hand or object pixels (as determined by the two-dimensional segmentation scheme) where the

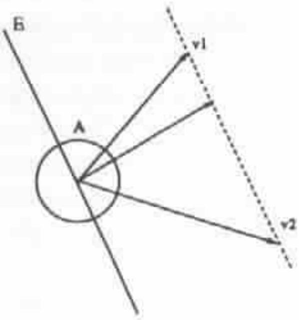


Fig. 9. The aperture problem: The direction of the edge E cannot be determined by viewing E through aperture A.

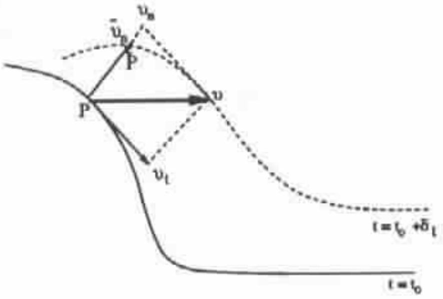


Fig. 10. Normal flow estimation.

intensity changes significantly. An edge detector is applied to the first image to estimate the edge magnitude $M(x, y)$ and direction $D(x, y)$ for every pixel:

$$M(x, y) \approx \sqrt{E_x^2 + E_y^2},$$
$$D(x, y) \approx \tan^{-1}\left(\frac{E_x}{E_y}\right)$$

where E_x and E_y are the partial derivatives of the first image with respect to x and y, respectively. The edge direction and magnitude is discretized depending on the size of the windows N and S. The motion is then estimated at only the pixels where the gradient magnitude exceeds the input threshold value. Motion ambiguity due to the aperture problem can be solved by estimating only the *normal* flow vector. It is well known that the motion along the direction of intensity gradient only can be recovered. Then we evaluate the SSD functions at only those locations that lie on the gradient directions and choose the one corresponding to the minimal SSD, if more than one minimal SSD exist we can choose the one corresponding to the smallest movement, as described above.

The full flow vector can then be estimated by using the following equation which relates the normal flow vector $\vec{v}_n$ to the full flow vector $\vec{v}$

$$\vec{v}_n = \vec{v} \cdot \vec{n}.$$

This method works under the asumption that the hand image motion is locally constant. Solving the over-determined linear system will result in a solution for the full flow. The least square error of the system can help us to decide whether the assumption is a reasonably valid one for determining the event that caused the transition in the DEDS. On the other hand, full flow determination can be performed for small clusters of points in the image and a number of full flow estimates is then used for 3D recovery.

To obtain sub-pixel accuracy, we can fit a one-dimensional curve along the direction of the gradient for all the SSD values obtained. A polynomial of the degree of the number of points used along the gradient can be used to obtain the best precision. However, for an S window of size 7×7 pixels or less and an N window of size 3×3 or so, a quadratic function can be used for efficiency and to avoid optimizational instabilities for higher order polynomials. Subpixel accuracy using a quadratic function is shown in Figure 11. The subpixel optimum can be obtained by finding the minimum of the function used and using the displacement at which it occurred as the image flow estimate. To avoid probable discontinuities in the SSD values, the image could be smoothed first using a gaussian with a small variance.

A simpler version of the above algorithm can be implemented in real-time using a multi-resolution approach [96]. We can restrict the window size of N to 3×3 and that of S to 5×5, and perform the algorithm on different levels of the gaussian image pyramid. A gaussian pyramid is constructed applications by the successive of gaussian low-pass filtering and decimation by half. The pyramid processor, PVM-1, is capable of producing complete gaussian pyramid from a 256 by 256 image in one video frame (1/30 of a second). Maxvideo boards can be used for the simultaneous estimation of image flow at all the levels of the pyramid for all the pixels. Image flow of 1 pixel at the second level would

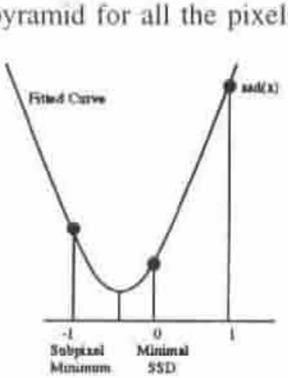


Fig. 11. Subpixel accuracy for optical flow.

correspond to 2 pixels in the original image, 1 pixel displacement at the third level would correspond to 4 pixels in the original image, and so on. The level with the smallest least square fitting error of the normal flow can be chosen to get the full flow and the motion vector is scaled accordingly. This method is crude in the sense that it only allow image flow values of 1, 2, 4 or 8 pixel displacement at each pixel, but it can be used for detecting fast movements of the hand. By either using a flow recovery algorithm or a feature idetification and tracking algorithm, we end up having a set of values for 2D displacements of a number of pixels. The problem is how can we model the uncertainty in those 2D estimates, which are to be used later for 3D parameter recovery. For example, if the estimate is – for a specific 3D feature – that pixel (x_i, y_j) has moved to pixel (x_m, y_n), then the problem reduces to finding space probability distributions for the four indices. 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. In the next section we discuss these problems in details.

4.2. RECOVERING 3D EVENTS

One can model an arbitrary 3D motion in terms of stationary-scene/moving-viewer as shown in Figure 12. The optical flow at the image plane can be related to the 3D world as indicated by the following pair of equations for each point (x, y) in the image plane [63]

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

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.

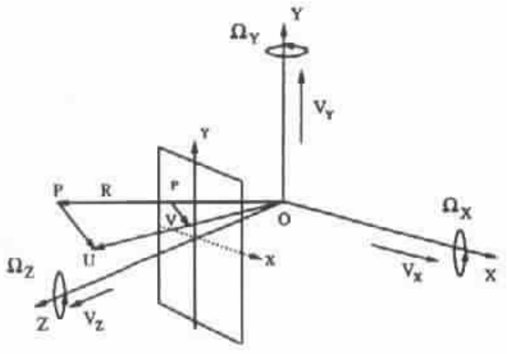


Fig. 12. 3D formulation for stationary scene moving viewer.

In this system of equations, the only knowns are the 2D vectors v_x and v_y, if we use the formulation with uncertainty then basically the 2D vectors are random variables with a known probability distribution. In case that the real 3D relationships between feature points (on the hand) are known, then recovering the absolute depth is a simple process. The equations can then be formalized, in case that the 3D features lie on a planar surface, as follows

$$v_x = (1 - px - qy)\left(x \frac{V_Z}{Z_o} - \frac{V_X}{Z_o}\right) + \left[xy\Omega_X - (1 + x^2)\Omega_Y + y\Omega_Z\right],$$
$$v_y = (1 - px - qy)\left(y \frac{V_Z}{Z_o} - \frac{V_Y}{Z_o}\right) + \left[(1 + y^2)\Omega_X - xy\Omega_Y - x\Omega_Z\right]$$

where Z_o is the absolute depth, p and q are the planar surface orientations. It should be noticed that the resulting system of equations is nonlinear, however, it has some linear properties. The rotational part, for example, is totally linear. In the next section we discuss different methods for solving the system of equations and thus recovering the 3D parameters in real time with and without uncertainty formulation.

A part of the events definition, as mentioned before, is the recognition of the existence of an object, for example. In other words, identifying objects in the visual scene and not only recovering 3D motion. Orientation of the object relative to the observer's camera and its shape can always be asserted by a simple 2D segmentation strategy as mentioned in the discussion about computing the 2D motion vectors. A data base of different shapes and orientations for different sized objects with the associated state that they may be manipulated in may be used and updated by the system. Correlation-based matching techniques can be used to compare 2D object representations, while moment computations are used to scale, shift and re-orient the shapes to be correlated. New objects can still be recognized and stored in this data base to facilitate future accesses.

4.3. THE CONTROLLABLE EVENTS

One kind of control inputs that can be supplied to the observer robot are the tracking actions. Depending on the nature of the manipulation process, the observer has to keep track of the hand and object within the camera image plane in such a way so as to be able to observe the process. The intelligent tracking control is supplied by the DEDS formulation. Simple-minded tracking ideas, like keeping fixed 3D relation between the camera and the manipulating agent are not to be used in our system. The manipulation action might be a simple one that does not require complex tracking, such as screwing and unscrewing, however, more complex events, where the hand may occlude the manipulation process, or when the hand starts moving away from the observer, might suggest the need for complex tracking mechanisms, including translations and rotations of the observing robot hand on which the camera is mounted.

A subset of the three-dimensional motion and structure parameters would have to be calculated using two or more frames [6, 34, 40, 78, 81, 83, 88, 93]. The size of the subset will depend on the *expected* kind of 3D motion, as the current state of the DEDS system will specify. Our system needs to track the object while using all the six degrees of freedom of the observer robot in order to position the observer at the best feasible position at different states of the automaton. Using rotations only to follow the end effector of the manipulating robot is not sufficient for the stabilizing observer.

Two kinds of tracking mechanisms can be used, in the first kind, the two images on which the motion estimation algorithms will be used, will be taken while the camera is stationary and then the camera will move and the process will be repeated after the camera stops. The observer movement will be a 'jerky' one. Another scheme can be used where the camera can grab images while the robot arm holding it is moving, in this case one should compensate for the moving arm before calculating the image flow of the hand and/or object. Thus, the problem reduces to finding the image flow due to the camera movement using the stationary-scene/moving-viewer 3D formulation. In the absence of translations, for example, we can compensate for the rotational part in a very fast and efficient way. Compensation will have to be performed before using the structure and motion recovery algorithms.

Some tracking actions would be performed for avoiding occlusions in order to be able to 'see' the workspace of the manipulating robot. Others would be performed for doing quick visual searches to locate the hand and/or objects if they go out of sight. The kind of mechanical control to be supplied is dependent on what the observer assert to be the current state in the DEDS automaton.

4.4. RECOVERING WORLD EVENTS

In this section we discuss different techniques for recovering the 3D events. We utilize the refined 2D motion estimates in order to achieve a robust estimation of the three-dimensional motion and structure vectors of the scene under observation. We develop some techniques for finding estimates of the required parameters and discuss mathematical formulations that will enable us to determine the 3D event distributions.

We concentrate in our treatment of the subject on determining the manipulating hand parameters, as the hand configuration is well defined, we also continue using the assumption that the feature points lie on a planar surface. As argued before, the extension to arbitrary configurations is straightforward. The object behaviour can be asserted using similar techniques and/or by observing conveniently located surface patches under similar assumptions. We use the recovered hand depth from observing a set of features and make assumptions about the depth of the objects relative to the hand depth. Using 2D cues about the object location, orientation, size and configuration (using correlation and moments) and combining them with 3D cues about the hand structure and motion in the world, enable the observer to make decisions about the actions being performed on the objects in the scene under observation.

We start by discussing a deterministic method to recover 3D parameters, then we describe other approximate methods. The problem of recovering scene structure and the camera motion relative to the scene has been one of the key problems in computer vision. Many techniques have been developed for the estimation of structure and motion parameters [39, 62, 80, 85, 93]. A lot of existing algorithms depend on evaluating the motion parameters between two successive frames in a sequence. However, recent research on structure and motion has been directed towards using a large number of frames to exploit the history of parametric evolution for a more accurate estimation and noise reduction [34, 40, 83, 88].

We develop a method for recovering the 3D motion and orientation of the planar surface (on which lies the hand features) from an evolving image sequence. The algorithm utilizes the image flow velocities in order to recover the 3D parameters. First, an algorithm is developed which iteratively improves the solution given two successive image frames. The solution space is divided into three subspaces – the translational motion, the rotational motion and the surface slope. The solution of each subspace is updated by using the current solution of the other two subspaces. The updating process continues until the motion parameters converge, or until no significant improvement is achieved. Second, we further improve the solution progressively by using a large number of image frames and the ordinary differential equations which describe the evolution of motion and structure over time. Our algorithm uses a weighted average of the expected parameters and the calculated parameters using the 2-frame iterative algorithm as current solution and continues in the same way till the end of the frame sequence. Thus it keeps track of the past history of parametric evolution. The solution is further improved by exploiting the temporal coherence of 3D motion. We develop the ordinary differential equations which describe the evolution of motion and structure in terms of the current motion/structure and the measurements (the 2D motion vectors) in the image plane. As an initial step we assume that the 3D motion is piecewise uniform in time. The extended Kalman filter can then be used to update the solution of the differential equations. Details of the algorithm are discussed in Appendix.

There are other non-iterative techniques for recovering the 3D parameters resulting from 2D motion between two frames. The methods that will be mentioned here rely on specific assumption regarding the hand geometry and/or world manipulating actions. Assuming that the actual relations between feature points that lie on the hand plane is well defined than a closed form solution for the structure parameters and depth can be estimated by using a method like the one described by Fischler and Bolles [32]. The motion parameters can then be easily recovered by solving a linear system in six parameters.

It should be noticed that we try to use alternative methods in order to solve linear equations at different automaton states, the motive behind that is the fact that linear systems can be solved in a pseudo-real time framework for a relatively small number of feature points and in addition a closed form solution always results. Another idea is to assume that the surface of the manipulating hand is frontal at the time of capturing the frame to be processed with the previous one, thus p and q are equal to zero, and the problem reduces to solving a linear system in six parameters for the motion parameters, while the depth is easily computed by knowing the 3D distance between any two feature points, thus Z_o is equal to

$$\sqrt{\frac{l^2}{\left[\left(\frac{x_2}{f-px_2-qy_2}-\frac{x_1}{f-px_1-qy_1}\right)^2+\left(\frac{y_2}{f-px_2-qy_2}-\frac{y_1}{f-px_1-qy_1}\right)^2+\left(\frac{1}{1-\frac{px_2}{f}-\frac{qy_2}{f}}-\frac{1}{1-\frac{px_1}{f}-\frac{qy_1}{f}}\right)^2\right]}}$$

where f is the focal length of the lens, l is the real 3D distance between two feature points on the hand and (x_1, y_1) and (x_2, y_2) are the CCD coordinates of the two image points.

The assumption here being that the observer always locates itself to a position in which the hand is frontal with respect to the camera image plane, and that manipulating movements while the camera is moving and during computations is negligible. Other formulations may attempt to find pseudo-close form solution of the nonlinear second order system and other assumptions, like the absence of rotational and/or translational motion reduces the complexity significantly.

5. Uncertainty Modeling and Identification

In this section we discuss uncertainty modeling for the observer framework. In particular, we describe some techniques for measuring and computing the uncertainties in the event identification mechanisms. The purpose of this discussion is to present the sources of uncertainty in the two-dimensional visual data, which is the only kind of input that the observer can sense through its camera. Then we procesed to identify methods by which the 2D uncertainty could be transformed into meaningful 3D interpretations that the observer automata can use reliably in order to recover the world events.

Figure 13 depicts the sequence of steps that are to be performed in order to recover the full world uncertainty from 2D measurements on the image plane. We start by recognizing the sensor uncertainty, then we recover the uncertainty resulting from the image processing technique that is used, the resulting 2D uncertainties are then refired and used to determine the 3D models. In the following three sections we discuss this sequence.

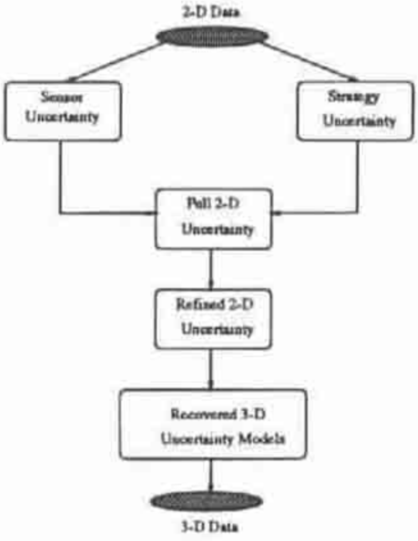


Fig. 13. Propagation of uncertainty.

5.1. SENSOR AND IMAGE PROCESSING UNCERTAINTIES

In this section we develop and discuss modeling the uncertainties in the recovered 2D displacement vectors. There are many sources of errors and ways to model uncertainties in image processing and sensing in general [22, 37, 95, 97]. As mentioned when describing techniques for recovering the image flow, the uncertainty in the recovered values results from sensor uncertainties and noise and from the image processing techniques used to extract and track features. When dealing with measurements of any sort, it is always the case that the measurements are accompanied by some error. Mistakes also occur, where mistakes are not large errors but failures of a system component or more. A description of errors, mistakes can be found in [5, 7].

5.1.1. Image Formation Errors

The observer robot uses a camera to grab and register images of the manipulation system, so we need to know errors in mapping from the 3D world features to the 2D domain which we use in forming 3D hypothesis about the task under supervision. The accuracy, precision and modeling uncertainty of the camera as our sensor is an important issue and the first step towards forming a full uncertainty model for recovering the 3D events in the observer automaton.

In Figure 14 (redrawn from [7]), a model of the image formation process is illustrated, which lists some salient features of each component. As a lot of the image processing algorithms compute derivatives of the intensity function, noise in the image will be amplified and propagated throughout the observation process. The goal of this treatment is to find a distribution for the uncertainty of mapping a specific 3D feature into a specific pixel value. In other words, if the feature 2D position was discovered to be (i, j), then the goal is to find a 2D distribution for i and j, assuming that there is no uncertainty in the technique used to extract the 2D feature.

The end product of modeling the sensor uncertainty is to be able to say a statement like: "The 3D feature F is located in the 2D pixel position (i, j) with probability p_1 or located in the 2D pixel position $(i, j+1)$ with probability p_2 or … *given* that the registered location is (l, m), such that $p_1 + p_2 + \cdots + p_n = 1$, and $\nexists$ error in the 2D feature recovery mechanism."

The errors in the image formation process are basically of two different kinds. The first type is a spatial error, the other type is a temporal error. The spatial error due to the noise characteristics of a CCD transducer can be due to many reasons, among which are dark signatures and illumination signatures. The technique to be used is to take a large number of images, we can denote the image intensity function as a 3D function $I(u, v, t)$, with spatial arguments u and v and temporal argument t. The sample mean of the image intensities over N time samples can be denoted by $\bar{I}(u, v)$

$$\bar{I}(u, v) = \frac{1}{N} \sum_{t=1}^{N} I(u, v, t).$$

The spatial variance in a 5×5 neighborhood of the means is computed by:

$$s^2(u, v) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} \left(\bar{I}(u+i, v+j) - \bar{I}(u, v)\right)^2.$$

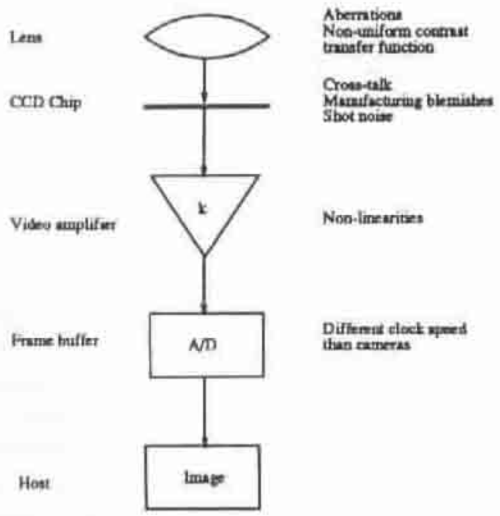


Fig. 14. Image formation.

The dark signature of the camera can be determined by computing $\bar{I}(u,v)$ of each pixel with the lens cap on. It will be found that a small number of pixels will have non-zero mean and non-zero variance. The specific pixel locations are blemished and should be registered. The uniform illumination is computed by placing a nylon diffuser over the lens and computing the mean and variance. It will be noticed that due to digitizing the CCD array into a pixel array of different size, and the difference in sample rates between the digitizer and camera, the border of the image will have different mean and variance from the interior of the image. Some 'stuck' pixels at the location of the blemished pixels will also be noted. The contrast transfer function will also be noted to vary at different distances from the center of the lens.

Temporal noise characteristics can also be identified by taking a number of experiments and notice the time dependency of the pixels intensity function. In our treatment and for our modeling purposes we concentrate on the spatial distribution of noise and its effect on finding the 2D uncertainty in recovering a 3D feature location in the pixel array.

5.1.2. Calibration and Modeling Uncertainties

Methods to compute the translation and rotation of the camera with respect to its coordinates, as well as the camera parameters, such as the focal length, radial distortion coefficients, scale factor and the image origin, have been developed and discussed in the literature [14, 50, 86]. In this section we use a static camera calibration technique to model the uncertainty in 3D to 2D feature locations. In particular we use the sequence of steps used to transform from 3D world coordinates to computer pixel coordinates in order to recover the pixel uncertainties, due to the sensor noise characteristics described previously.

A sequence of calibration steps is used for a coplanar set of points in order to obtain the rotation and translation matrices, in addition to the camera parameters. The input to the system are two sets of coordinates, (X_f, Y_f), which are the computer 2D pixel image coordinates in frame memory and (x_w, y_w, z_w), which are the 3D world coordinates of a set of coplanar points impressed on a piece of paper with known inter-point distances. A discussion of the exact mathematical formulation of the inter-step computations to find all the parameters can be found in [14].

Our approach is to treat the whole camera system as a black box and make input/output measurements and develop a model of its parametric behaviour. The next step is to utilize the recovered camera parameters and the number of 3D points which we created in order to formulate a distribution of the 2D uncertainty. The points used in calibration and later in recovering the distribution can be the actual features on the robot hand that are to be tracked and thus providing a similar experimental environment to the one that the observer will operate in. The strategy used to find the 2D uncertainty in the features 2D representation is to utilize the recovered camera parameters and the 3D world coordinates (x_w, y_w, z_w) of the 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 2D histograms for the displacements 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. Different histograms can be used for different 2D pixel positions distributed throughout the image plane. 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, thus modeling the sensor uncertainty. The black box approach is thus used to model errors in a statistical sense.

5.1.3. Image Processing Uncertainties

In this section we describe a technique by which developing uncertainties due to the image processing strategy can be modeled. In addition, we end the discussion by combining both the sensor uncertainties developed in the previous section and the models developed in this section to generate distribution models for the uncertainty in estimating the 2D motion vectors. These models are to be used for determining the full uncertainty in recovering the 3D events that causes state transitions between states of the observer automaton.

We start by identifying some basic measures and ideas that are used frequently to recognize the behaviour of basic image processing algorithms and then proceed to describe the technique we use in order to compute the error model in locating certain features from their 2D representation in the pixel array. We concentrate on modeling the error incurred in extracting edges, as edge extraction is a very popular mechanism that is used for both identifying feature points on the manipulating hand and also for computing 2D contours of the object under supervision. When we discussed flow recovery techniques before, it was discussed in details that the optic flow recovery algorithm using local matching works well for the intensity boundaries and not for the inside regions.

Edge extraction strategies and methods to evaluate their performance qualitatively and quantatively have been presented and discussed in the literature [1,

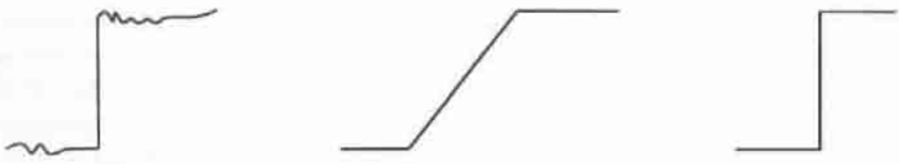


Fig. 15. Different types of edges.

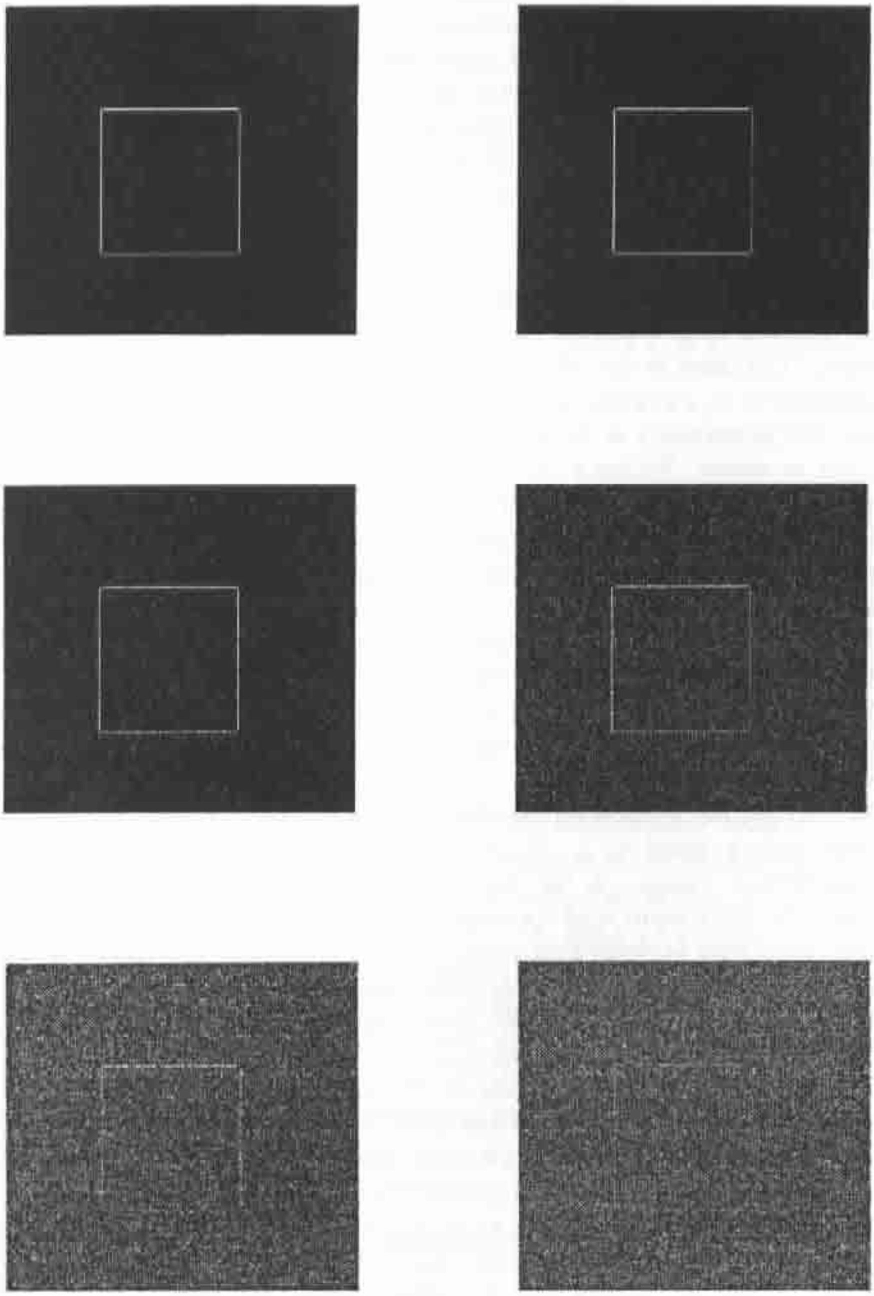


Fig. 16. Edge detection results for different noise levels.

31, 33, 38, 53, 56, 69]. There are many types of edges, ideal, ramp and noisy edges as shown in Figure 15 are only three of them. Different curvatures in the edges also constitute another dimension to be taken into consideration when it comes to asserting the types of edges that exist.

The goal of developing the error models for edge extraction to be able to say a statement like: "Given that the 2D feature recovered using the edge recovery S is in pixel position (x, y), then there is a probability that the feature was originally at pixel position $(x+1, y)$ with probability p_1 or … etc. due to the noise in the pixel image, such that $p_1 + p_2 + \cdots + p_n = 1$." The problem is to find the probabilities.

It should be obvious that there may be different types of noises and also different levels of those types that might vary at different locations in the sensor image plane. This adds to the different models that we might have to construct. Our approach is to use ideal, that is, synthesized edges of different types, locations and also orientations in image frames then corrupt them with different kinds and levels of noises. We know the ideal edge points from the ideal image, for which we shall use the edge detector that is to be used in the observer experiment. The corrupted images will then be operated upon by the detector and the edge points located. The edge points will differ from the ideal image edge points. The problem reduces to finding corresponding edge points in corrupted and ideal images then finding the error along a large number of edge points. A 2D histogram is then constructed for the number of points with specific displacement errors from the ideal point. The volume of the histogram is then normalized to be equal to 1, the resulting 3D function is the 2D probability density function of the error of displacements.

In Figure 16, an ideal box is drawn, then corrupted with an additive gaussian noise with σ equal to 3, 10, 20, 30 and 40 respectively and then the edges computed as shown. In the box there are four different kinds of ideal edges (different orientations with the object inside or outside of the background). The correspondence between edge points in the corrupted and ideal is established by choosing the point with the *minimal* distance from the ideal edge point, *such that* it does not correspond to another ideal edge point. The histogram is constructed for each edge and then normalized. For practicality measures, the process can be repeated for orientations differing by 15° and the set of distributions preserved. Whenever the observer automaton deals with a specific edge while extracting features, the corresponding distribution is referenced.

5.1.4. Computing 2D Motion Uncertainty

In this section we describe how to combine sensor and image processing strategy error models to compute models for the recovered image flow values. To simplify the idea, let's assume that we have recovered a specific feature point (x_1, y_1) in an image grabbed at time instant t and the corresponding point (x_2, y_2) at time

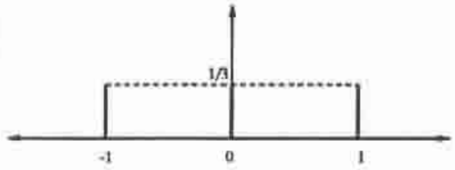


Fig. 17. Distribution of the x-coordinate displacement.

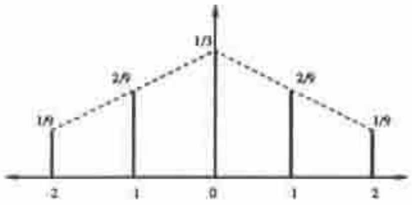


Fig. 18. Combined sensor and strategy distribution.

$t+1$. The problem is to figure out the distribution of v_x. As an example, to explain the procedure, let's assume that from the 3D sensor distribution we have have computed the marginal density function of the x coordinate of x_1 in the point

$$f_X(x) = \int_R f_{X,Y}(x,y)\,\mathrm{d}y,$$

where R is all the possible y values within the sensor uncertainty model.

The same process is applied for the strategy distribution and another function is recovered. To simplify things, lets assume that both distributions are identical to the distribution in Figure 17, that is, there is an equal probability equal to $1/3$ that the x coordinate is the same, or shifted one position to the left or the right. Combining the spatial information of both distributions as a convolution process would produce the distribution shown in Figure 18, which is the error probability density function of having the 3D feature x 2D coordinate in the recovered image 2D x position. Further more, assume that x_2 distribution is the same.

The problem reduces to finding the distribution of the optic flow x component, using these two combined distributions. As an example, if $x_1 = 10$ and $x_2 = 22$, then all probability statements can be easily computed, a set of some of these probability statement is shown

$$P(v_x = 8) = P\big((x_1 = 12) \wedge (x_2 = 20)\big) = \frac{1}{9} \times \frac{1}{9} = \frac{1}{81},$$

$$P(v_x = 9) = P\Big(\big((x_1 = 12) \wedge (x_2 = 21)\big) \vee \big((x_1 = 11) \wedge (x_2 = 20)\big)\Big)$$
$$= \left(\frac{1}{9} \times \frac{2}{9}\right) + \left(\frac{2}{9} \times \frac{1}{9}\right) = \frac{4}{81},$$

$$P(v_x = 10 \mid x_1 = 10) = \frac{P(x_1 = 10 \wedge x_2 = 20)}{P(x_1 = 10)} = \frac{\frac{3}{9} \times \frac{1}{9}}{\frac{3}{9}} = \frac{1}{9}.$$

Consequently, all distributions and expected values can be computed from the combination of the sensor level and strategy level uncertainty formulation. Those flow models are then passed to the higher levels for 3D recovery. In the next section we discuss a method for refining the measured 2D motion vectors and we then proceed to formulate the 3D modeling of events as defined by the observer automaton.

5.2. REFINING IMAGE MOTION

In this section we describe a method to refine the recovered 2D motion vectors on the image plane. Having obtained from the sensor and extraction strategy uncertainty levels distribution estimates for the image flow of the different features, we now try to eliminate the unrealistic ones. We concentrate on the flow estimates for the motion of the manipulating hand and develop a technique that is to be used during the observation process as a means to reject faulty estimates. Faulty estimates can results from noise, errors or mistakes in the sensor acquisition process, manipulation or visual problems like occlusion, modeling the uncertainties in the previous two levels may still leave room for such anomalies.

We assume that the features to be tracked on the hand lie on a planar surface or that segmenting the hand as a polyhedra object into planar surfaces is simple, although the modification would be very simple to allow for arbitrary 3D positions of the feature distribution. Since we know a priori some information about the mechanical capabilities and limitations and geometric properties of the hand, also about the rate of visual sampling for the observer, since we actualy control that, we might be able to assert some limits on some of the visual parameters in our system.

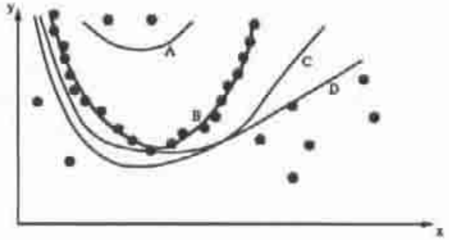


Fig. 19. Fitting parabolic curves.

To illustrate the idea behind the approach, consider Figure 19, assume all the curves are 2D parabolic functions $y = ax^2 + bx + c$, if the set of data points are as shown in the figure, then a least square error fit will produce the function D. However, if we know some upper and lower limits on the values of the coefficients a, b and c then we might be able to construct an upper and lower function parabolas A and C as an enclosing envelope, outside which we can reject all the data points. In that case, we can do a fit for the points that lie inside the envelope and obtain a more realistic function as shown by the curve B.

The situation for rejecting estimates for the image flow is not much different. We know equations that govern the behaviour of the image flow as a function of the structure and 3D motion parameters, as follows

$$v_x = (1 - px - qy)\left(x\,\frac{V_Z}{Z_o} - \frac{V_X}{Z_o}\right) + \left[xy\Omega_X - (1 + x^2)\Omega_Y + y\Omega_Z\right],$$
$$v_y = (1 - px - qy)\left(y\,\frac{V_Z}{Z_o} - \frac{V_Y}{Z_o}\right) + \left[(1 + y^2)\Omega_X - xy\Omega_Y - x\Omega_Z\right].$$

Which are second degree functions in x and y in three dimensions, $v_x = f_1(x, y)$ and $v_y = f_2(x, y)$. In addition, we know upper and lower limits on the coefficients $p, q, V_X, V_Y, V_Z, \Omega_X, \Omega_Y, \Omega_Z$ and Z_o, as we know that the mechanical abilities of the robot arm holding the hand will make the relative velocity and distance between the camera impossible to exceed specific values within visual sampling timing period. So the problem reduces to constructing the three-dimensional envelopes for v_x and v_y as the worst case estimates for the flow velocity and rejecting any measured values that lie outside that envelope. Figure 20 indicates the maximal and minimal v_x that can ever be registered on the CCD array of the camera, the x and y are in millimeters and the $x - y$ plane represents the CCD image plane, the depth Z is the maximal v_x in millimeters on the CCD array that can ever be registered. It can be noticed that they are symmetric due to the symmetry in the limits of the coefficients.

As an example, we write the equation governing the maximum v_x value in the first quadrant of the $x - y$ plane (x^+, y^+).

$$\begin{aligned} v_{x_{\max}} = {} & \left(-\frac{fV_{X_s}}{Z_{o_s}} - f\Omega_{Y_s}\right) + \left(\frac{V_{Z_l}}{Z_{o_s}} + \frac{\max(q_l V_{X_l}, p_s, V_{X_s})}{Z_{o_s}}\right)x + \\ & + \left(\frac{\max(q_l V_{X_l}, q_s, V_{X_s})}{Z_{o_s}} + \Omega_{Z_l}\right)y + \\ & + \left(\frac{\Omega_{X_l}}{f} - \frac{\min(q_l V_{Z_s}, q_s, V_{Z_l})}{fZ_{o_s}}\right)xy - \\ & - \left(\frac{\min(p_l V_{Z_s}, p_s, V_{Z_l})}{fZ_{o_s}} + \frac{\Omega_{Y_s}}{f}\right)x^2 \end{aligned}$$

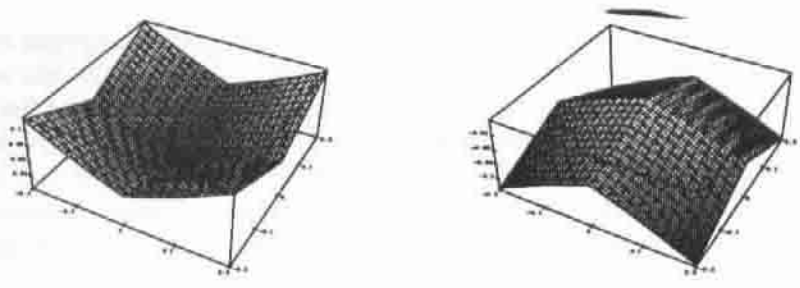


Fig. 20. Maximal and minimal v_x.

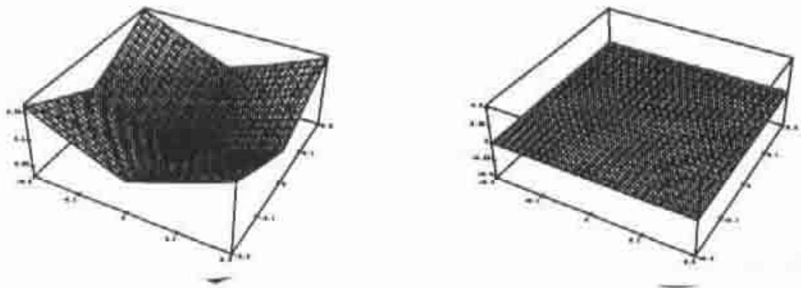


Fig. 21. Maximal and minimal flow magnitude.

where the subscripts s and l denote lower and upper limits, respectively. At first sight the problem of determining the maximum value of v_x seems to be a constrained nonlinear optimization problem, which is true, however, assuming that the upper and lower limits of the coefficients are equal in magnitude and opposite in directions (except for Z_o, which is used only as $Z_{o_s}^+$) makes the input to the max and min functions in the above equations always equal and thus providing one more degree of freedom in choosing the parameters and making the choice consistent throughout the equation. Thus the problem becomes simply to write eight equations as the above one for each of v_x and v_y, to draw the function in each of the four quadrants for maximum and minimum envelopes. We shall not rewrite the sixteen equations here, but we show the results for v_x in Figure 20, Figure 21 represents the maximum and minimum magnitude $m(x,y)$ for the image flow at any given point, where

$$m(x,y) = \sqrt{v_x^2 + v_y^2}.$$

It should be noted that the maximum absolute possible value of the image flow is minimal at the origin of the camera image plane and increases quadratically as the distance increases from the center.

The above envelopes are then used to reject unrealistic 2D velocity estimates at different pixel coordinates in the image. As a further note, it should be mentioned that some on-line elimination procedures can be implemented depending on the current positions in the observer automaton, for example, the image flow field tends to assume certain configurations in the image plane depending on the 3D motion, independent of the object or the hand structure, if the motion is only relative rotational velocities, the flow vectors all tend through pass from the same point. In other words, in addition to off-line a priori estimation of the envelopes and on-line testing of measurements, we can also develop custom rejection techniques for certain observer automata states.

5.3. RECOVERING 3D UNCERTAINTIES

Having discussed methods for computing the three-dimensional motion vectors and structure parameters between two image frames, we now use the same formulations described earlier for 3D recovery but using 2D error distributions as estimates for motion and/or feature coordinates in order to compute 3D uncertainty distributions for the real world motion vectors and structure instead of singular values for the world events.

As an example to illustrate the idea, let's assume that we have a linear system of equations as follows

$$x + 3y = z_1,$$
$$2x + y = z_2.$$

The solution of this system is very easily obtained as

$$x = \frac{3}{5}\, z_2 - \frac{1}{5}\, z_1,$$
$$y = \frac{2}{5}\, z_1 - \frac{1}{5}\, z_2.$$

That is, a linear combination of the right hand side parameters. If the parameters z_1 and z_2 were random variables of known probability distributions instead of constants, then the problem becomes slightly harder, which is, to find the linear combination of those random variables as another random variable. The obvious way of doing this would be to use convolutions and the formula

$$P_{X_1+X_2}(y) = \sum_{R} P_{X_1,X_2}(x, y - x)$$

for the sum of two random variables X_1, X_2 for any real number y and/or the formula for linear combinations over the region R, which is for all x such that $P_{X_1,X_2}(x, y-x) > 0$. Using the moment generating function or the characteristic function seems also to be a very attractive alternative. The moment generating function M of a linear combination of random variables, for example X_1, X_2 can be written as

$$M_{aX_1+bX_2+c}(t) = \mathrm{e}^{ct}\big(M_{X_1}(at)M_{X_2}(bt)\big).$$

for independent random variables X_1, X_2. That is, the problem of solving linear systems on the form $Ax = b$, where b is a vector of random variables, may be reduced to finding closed form solutions for x in terms of the random parameters (using any elimination technique) and then manipulating the results and finding different expectations using moment generating or characteristic functions. The solutions we suggest to this problem of finding the random variable solution of the 3D parameters utilize the techniques we described in the previous section. Using either the two-frame iterative technique or the closed form algorithms, it should be noticed that the problem reduces to either solving multi-linear systems or a single one. In that case, using elimination and characteristic functions for computing the required expectations and distributions is straight forward. As an example, the recovered 3D translational velocity cumulative density functions for an actual world motion of the Lord gripper equal to

$$V_X = 0 \text{ cm}, \quad V_Y = 0 \text{ cm} \quad \text{and} \quad V_Z = 13 \text{ cm}$$

is shown in Figure 22. It should be noted that the recovered distributions represents a fairly accurate estimation of the actual 3D motion.

Thus, we have suggested algorithms for the quick estimation of the 3D uncertainties in the structure and motion of the manipulation system. The next step would be to refine these estimates and use them for asserting the world events. Next, we describe techniques for eliminating and refining the 3D models of manipulation under observation, whose recovery was discussed in the previous sections. In particular, we discuss a strategy to reject improbable events that might have been computed due to noise and uncertainties that were not compensated for in the distribution formulation, also because of unsmooth visual artifacts. We employ both existing knowledge about the mechanical properties of the manipulation and also knowledge from the current state of the observer automaton.

We concentrate our treatment of the subject on the three-dimensional behaviour of the hand that is used in manipulation. The hand is assumed to be a well defined entity, and as we mentioned before, changing the hand and/or its characteristics can be modeled by simply plugging in a module that describes the new characteristics, the *same* hand is used through out the entire manipulation activities.

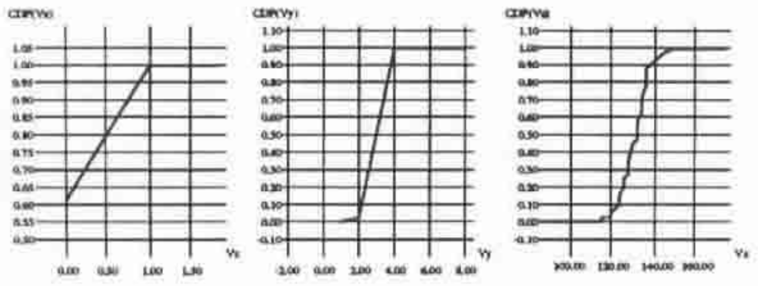


Fig. 22. Cumulative density functions of the translational velocity.

Knowing the joint limits of the manipulating robot will enable us to reject improbable recovered 3D motion vectors, that could not have occurred in the real 3D world. As an example, assuming that we use a gripper with two 'claws' having only one degree of freedom, then, obviously, any recovered 3D rotational velocities for the claws should be rejected. Unrealistic slope estimations should also be rejected, knowing the robotic reachability of the end effector, with respect to the viewer.

The current position in the observer automata will allow refining the recovered 3D event distributions, as it might well be the case that impossible manipulation actions at a specific manipulation stage are recovered. It is impossible, for example, due to the visual sampling rate, that the hand is in an upright position holding a nail in the center of the image plane at a time step, then having it disappear or hold another object at a dramatically distant 3D position in the next time step, unless, of course a manipulation or viewer system failure has happened. In that case, some designated fail state should be accessed, discarding the recovered parameters. Limits on $V_X, V_Y, V_Z, \Omega_X, \Omega_Y, \Omega_Z$ and Z are asserted for every observer subset of states, and used for refining the recovered 3D world events.

6. Utilizing the Framework

In this section we discuss how to use the framework of states, events and uncertainty that was developed in Sections 3, 4 and 5. In particular, we address a strategy for navigating the observer automaton and subsequently assert different manipulation states and perform the necessary tracking actions. We address some other aspects of our hybrid representation that could be explored to achieve a higher level of robustness and modularity. We discuss detection of error states and/or sequences which could have a number of applications in the manufacturing domain. We then proceed to discuss strategies for constructing a mapping module that transforms task descriptions into a design of a DEDS automaton. We also address some computational complexity issues for the developed framework.

At this point in the hierarchy of recovery and uncertainty levels, we have established methods and algorithms for recovering the refined three-dimensional velocity and relevant structure parameters of the scene under observation. In addition, we have computed the distribution of the uncertainty in the numerical values of some of the parameters in real-time. For example, the computed value for the translational velocity V_X might be a random variable lying between two values V_1 and V_2 with a known probability distribution $\mathcal{F}$. The same applies for all the other parameters for the different components in the scene. The problem now is how to make use of these distribution values in order to be able to navigate in the observer automaton as defined in Section 2 and demonstrated by examples in Section 3. In the following section we describe one such navigation strategy.

6.1. NAVIGATING THE OBSERVER AUTOMATON

Having built the DEDS automaton model of the visual system and its observer, we have a set of events that are defined as ranges on the visual scene parameters that causes state transitions between the automaton states. As a simple example, there might be two different events branching from a state in some task observer automaton and causing state transition to two other states, and a self loop caused by the continuous dynamics within a coarse quantization of a DEDS state, as follows

$$
\begin{aligned}
&S_1 \xrightarrow{e_1 : \Omega_1 < \Omega_Y < \Omega_2} S_2,\\
&S_l \xrightarrow{e_2 : -\Omega_1 < \Omega_Y < \Omega_1} S_l,\\
&S_1 \xrightarrow{e_3 : -\Omega_2 < \Omega_Y < -\Omega_1} S_3.
\end{aligned}
$$

In addition to other limits on the other scene parameters. That is, if Ω_Y occurs within a specific range, then the corresponding state transition should be asserted according to the above set of event description.

The problem then reduces to computing the corresponding areas under the refined distribution curves obtained from the hierarchy levels. In the case of the presence of more than a single parameter in the transition event description, then the corresponding area under *each* parameter curve is computed and multiplied for each parameter in the event definition. The goal is to find the probability of the occurrence of each event. In the above example, the goal would be to find the probability of e_1, e_2 and e_3.

An obvious way of using those probability values is to establish some threshold values and assert transitions according to those thresholds. For example, if for any event in the set (e_1, e_2 and e_3), the computed probability of the range is > 0.7, then the corresponding state transition should be asserted. It should be noted that those threshold values are highly task and state-dependent, appropriate values for the thresholds can be determined by performing many experiments for different task descriptions. The thresholds can also be updated adaptively according to the current manipulation patterns under observation. Many problems may arise after having obtained the above probabilities at the current automaton state. 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, or that a gross error has occurred in manipulation or some system failure.

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 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 is implemented using a stack-like structure associated with each state that has already been traversed. A stack

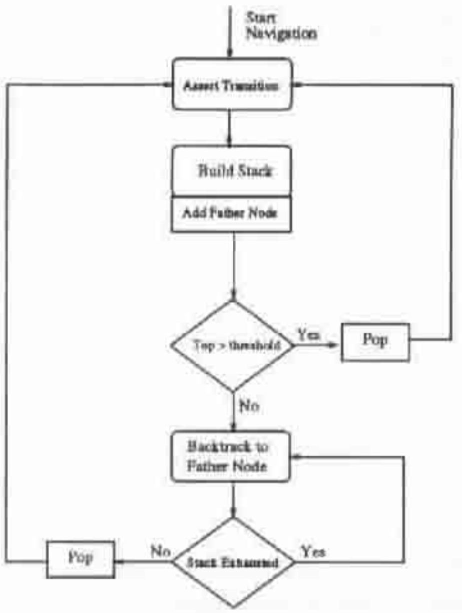


Fig. 23. Navigating the observer automata.

of the latest computed probability values sorted in descending order as an index to the corresponding event. As soon as a forward traversal is performed, the top value should be popped. Backtracking can be done by using the top of the stack value and do the corresponding transition and compute the new probabilities for the events. A father state parameter should also accompany each state that has been already been traversed. In case all the stack has been exhausted for a specific state, the father state should be accessed and a new route be accessed. Exhausted states are labeled and never revisited while backtracking. For states that have not been visited at all, new stacks and computations should be performed. Figure 23 depicts a flowchart for the traversal procedure.

Having established techniques for navigating the observer, the model description is now completed. The formulation uses uncertainties to assert current states of the manipulation system and attempts to recover from mistakes and errors. The model uses different intermediate levels for computing uncertainties, from the sensor level to the observer automaton level.

6.2. HYBRID REPRESENTATION FOR OBSERVERS

In this section we argue that our formulation for the observer system is a good solution for the general problem of defining, monitoring and controlling hybrid system as applied to the observation domain. Hybrid systems, as mentioned in Section 2 are systems in which digital and analogue devices and sensors interact over time and in which representation of states and the physical system condition includes continuous and discrete numerics, in addition to symbols and logical parameters. The problem of visual observation falls with in the description of hybrid systems, as there is a need to report, observe and control *distinct* and *discrete* system states. There is also a need for recognizing *continuous* 2D and 3D evolution of parameters. Also, there should be a *symbolic* monitoring of the current state of the system, especially in the manipulation domain. What we present in this work is a framework that works for the class of hybrid systems encountered within the robotic observation paradigm. The representation we discussed in the previous sections allows for the symbolic and numeric, continuous and discrete aspects of the observation task.

The framework we use also allows for the propagation of uncertainty in the world and asserts state transitions and events accordingly. The system also attempts to recover from mistakes that happen in the event recovery mechanism and/or the manipulation domain. The representation allows for flexibility in the coarse description of the system states. We have used 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. The approach used in our framework focuses on issues in visual observation that are decided by a suitable set of tracking actions which the observer can undertake to reposition itself autonomously and intelligently.

6.3. HIERARCHICAL REPRESENTATION

Figure 24 shows a hierarchy of three submodels. Motives behind establishing hierarchies in the DEDS modeling of different tasks includes reducing the search space of the observer and exhibiting modularity in the design. This is done through the designer, who subdivides the task space of the manipulating robot

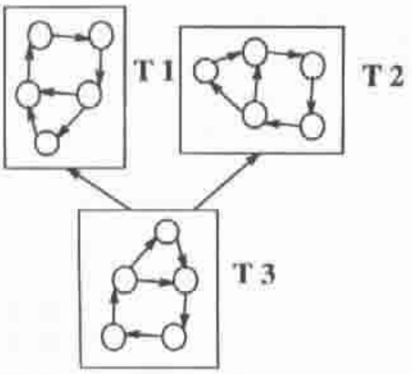


Fig. 24. A hierarchy of tasks.

into separate submodels that are inherently independent. Key events causes the transfer of the observer control to new submodels within the hierarchical description. For example, having a separate submodel for the grasping action might enable the DEDS observer automaton to concentrate on the part of the visual scene in which tha claws of the hand starts to enclose the object, while the observer remains in that submodel. An event like the object disappearance or the full enclosure and motion of the hand would then transfer the control in the observer agent to a different submodel. Transfer of control through the observer hierarchy of models allows coarse to fine shift of attention in recovering events and asserting state transitions.

6.4 ERROR STATES AND SEQUENCES

In this section we utilize the observer framework for recognizing error states and sequences. The idea behind this recognition task is to be able to report on *visually incorrect* sequences. In particular, if there a pre-determined observer model of a particular task under observation, then it would be useful to determine if something went wrong with the manipulation actions. The goal of this reporting procedure could be to alert operators or possibly to supply feedback to the manipulating robot so that it could correct its actions.

We do not consider supplying any sort of correction commands to the manipulating robot. In this treatment we consider only the recognition of such errors. Some examples of errors in manipulation include unexpected behaviour of the system, such as objects falling unexpectedly from the manipulating hand during a grasp and lift operation or some visual errors like unexpected occlusions between the observer camera and the manipulation environment.

There are a number of ways in which these problems could be reported. One such way can be to comply with the navigation strategy that was described in the first section of this section in order to capture the current state, if some event match occurred while navigating. However, if no match occurs than the error would have to be reported. Another quick method would be to report directly on any such inconsistencies.

A simple example for an inconsistent manipulation sequence is shown in Figure 25. If the automata model does not allow for the sudden disapearance of enclosed objects (i.e., if an object is enclosed, then the hand would have to put the object on a support surface before releasing its grip) then the transition from the visual scene (e) to (f) is clearly an illegal one.

The correct sequences of automata state transitions can be formulated as the set of strings that are *acceptable* by the observer automaton. This set of strings represents precisely the *language* describing all possible visual task evolution steps. Thinking of the acceptable DEDS observer sequences as the *language* accepted by the automaton model motivated describing a possible strategy for constructing a mapping module. Which is discussed in the next section.

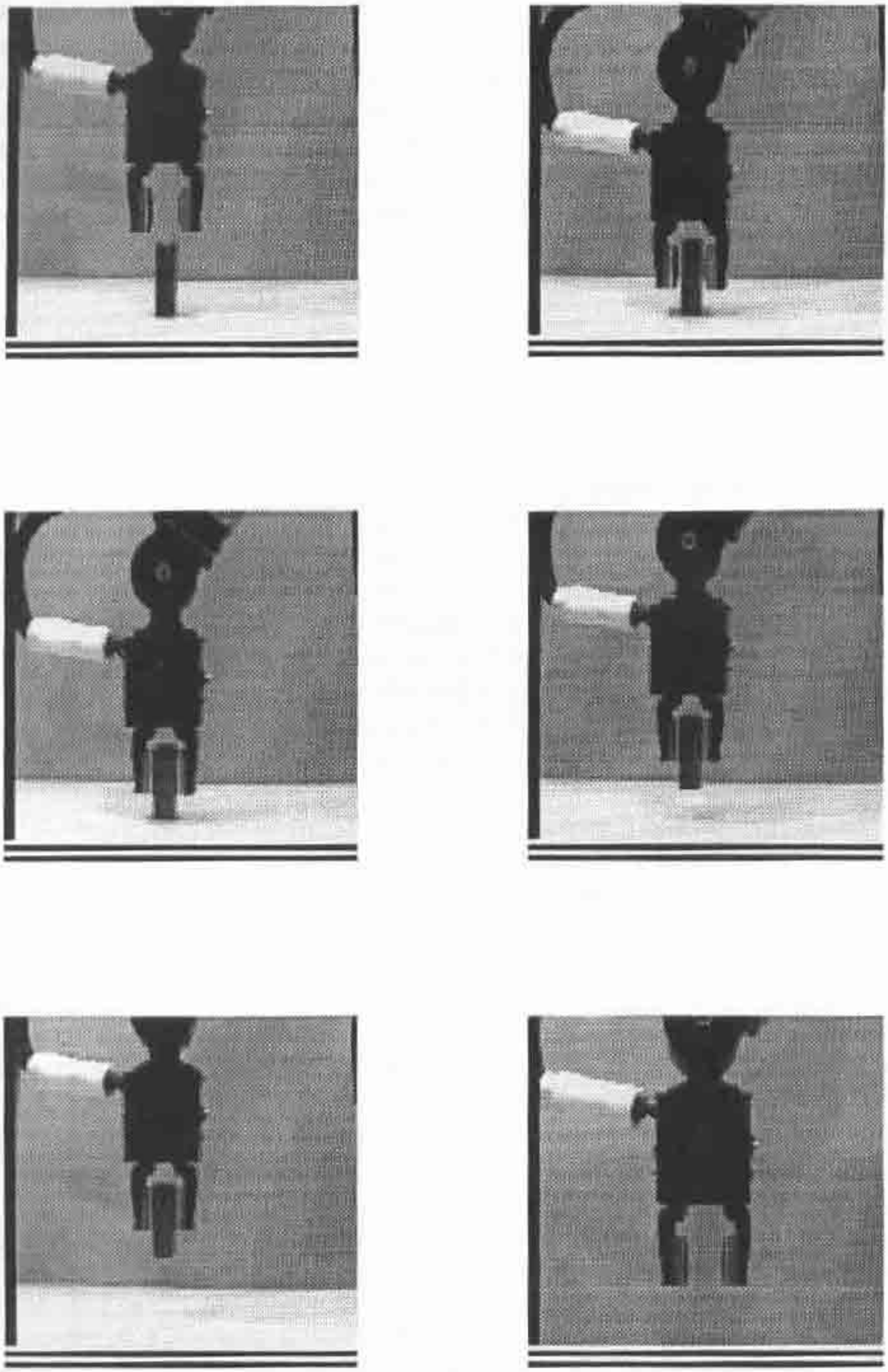


Fig. 25. An error lifting sequence.

6.5. MAPPING MODULE

The object of having a mapping module is to dispense with the need for the manual design of DEDS automata for various tasks. In particular, we would like to have an off line module which is to be supplied with some symbolic description of the task under observation and whose output would be the code for a DEDS automata that is to be executed as the observer agent. A graphical representation of the mapping module is shown in Figure 26.

The problem reduces to figuring out what is an appropriate form for the task description. The discussion in the previous section motivated regarding this problem as the inverse problem of determining acceptable languages for a specific DEDS observer automaton. In particular, we suggest a skeleton for the mapping module that transform a collection of input strings into an automata model.

The idea is to supply the mapping module with a collection of strings that represents possible state transition sequences. The input highly depends on the task under observation, what is considered as relevant states and how coarse should the automata be. The sequences are input by an operator. It should be obvious that the 'Garbage-in-garbage-out' principle holds for the construction process, in particular if the set of input strings is not representative of all possible scene evolutions; then the automata would be a faulty one. The experience and knowledge that the operator have would influence the outcome of the resulting model. However, it should be noticed that the level of experience needed for providing these sets of strings is much lower than the level of experience needed for a designer to actually construct a DEDS automata manually. The description of the events that cause transitions between different symbols in the set of strings should be supplied to the module in the form of a list.

As an illustrative example, suppose that the task under is simple grasping of one object and that all we care to know is three configurations; whether the hand is alone in scene, whether there is an object in addition to the hand and whether enclosure has occurred. If we represent the configurations by three states h, h_o and h_c. Then the operator would have to supply the mapping module with a list of strings in a language, whose alphabet consists of those three symbols, and those strings should span the entire language, so that the resulting automata would accept all possible configuration sequences. The mapping from a set of

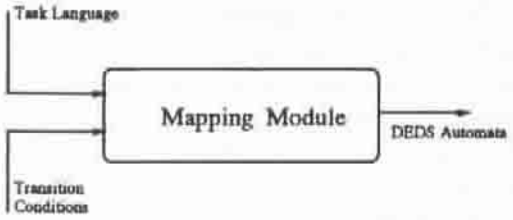


Fig. 26. The mapping module.

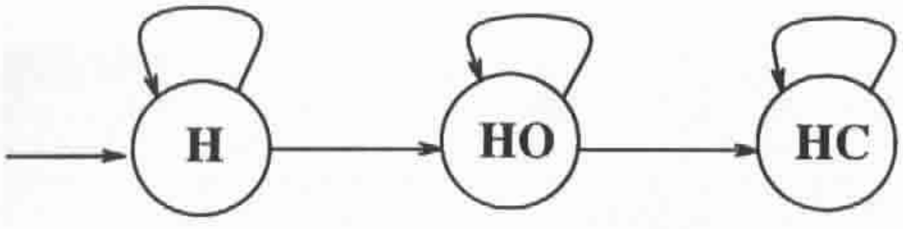


Fig. 27. An automaton for simple grasping.

strings in a regular language into a minimal equivalent automata is a solved problem in automata theory.

One possible language to describe this simple automata is

$$L = hh^* h_o h_o^* h_c h_o^*,$$

and a corresponding DEDS automata is shown in Figure 27.

The best-case scenario would have been for the operator to supply exactly the language L to the mapping module with the appropriate event definitions. However, it could be the case that the set of strings that the operator supply do not represent the task language correctly, and in that case some learning techniques would have to be implemented which, in effect, augment the input set of strings into a language that satisfies some pre-determined criteria. For example, y^* is substituted for any string of y's having a length greater than n, and so on. In that case the resulting automata would be correct up to a certain degree, depending on the operator's experience and the correctness of the learning strategy.

6.6. COMPLEXITY ANALYSIS

In this section we discuss complexity issues for some of the techniques used throughout the developed model. The issue of the computational complexity of *building* the DEDS model will not be discussed here, as the process is currently being done manually. The computational cost for traversing the automaton and asserting the events is the focus of the complexity issues under consideration.

Asserting state transitions within the DEDS observer model is one of the major sources of the computational burden associated with navigating the state space of the observer automata. There is a computational cost associated with converting 2D cues from the camera image plane into meaningful descriptions of the 3D world event uncertainty. As discussed in Section 5, recovering the 3D cumulative density function (CDF) distributions of world events involves solving linear systems of random variables, in particular performing a set of spatial convolutions for the different estimates in the image plane to propagate them into a set of 3D parameter distributions. The factors contributing to that kind of computation are the number of points and the number of spatial samples (in the 2D plane) taken to represent the spatial probability density function of having the 2D feature at a specific (x, y) location. The process involves using the moments generating function in the x and y directions for each point and multiplying them to solve each linear equation.

The number of coefficients in the equations is equal to: 2× number of points, as there is a factor involving the x and y coordinates for each point. The cost for computing the moment generating function (MGF) for each 3D value involves computing the product of all the MGF's of the 2D coefficients, which depends on the number of samples in each coefficient distribution and is exponential with respect to the number of coefficients. Thus the number of operations is of the order

$$O(m^{2n}).$$

Where m is the number of samples for each 2D distribution and n is the number of points. It is obvious that the process will work for only a small number of feature points and a low number of 2D distribution samples. The complexity of the traversing the DEDS automaton graph is essentially the complexity of doing a depth first search and thus is considered to be of an acceptable computational behaviour.

7. The Experimental System and Results

7.1. THE EXPERIMENTAL DESIGN

The design and the experiments for the proposed framework were performed on the architecture shown in Figure 28. The manipulating agent is the Lord experimental gripper and is mounted on a PUMA 560. The manipulating agent is essentially model by an external operator to perform some actions on a set of objects lying on a table. There is no coupling between the observer robot and the manipulation robot.

The observer agent is another PUMA 560 on which a camera is mounted. The low level visual feature acquisition is performed on the MaxVideo pipelined video processor at frame rate. In particular, there are two separate paths from the vision sensor. One path is for the computation of the hand 3D position and orientation and this is done through the MaxVideo. The other path (the inner loop) is done on a SparcStation, in which the image processing modules resides, those modules compute 2D cues from the scene under observation. Identification of objects, their location with respect to the hand and establishing contact, moments and correlation procedures are all performed within the inner loop.

The 2D to 3D conversion, probability computations, and the state machine transitions are performed on another SparcStation. All the 'thinking', uncertainty recovery and DEDS automaton updating is performed on that machine. The decision modules get their input data from the feature acquisition procedure

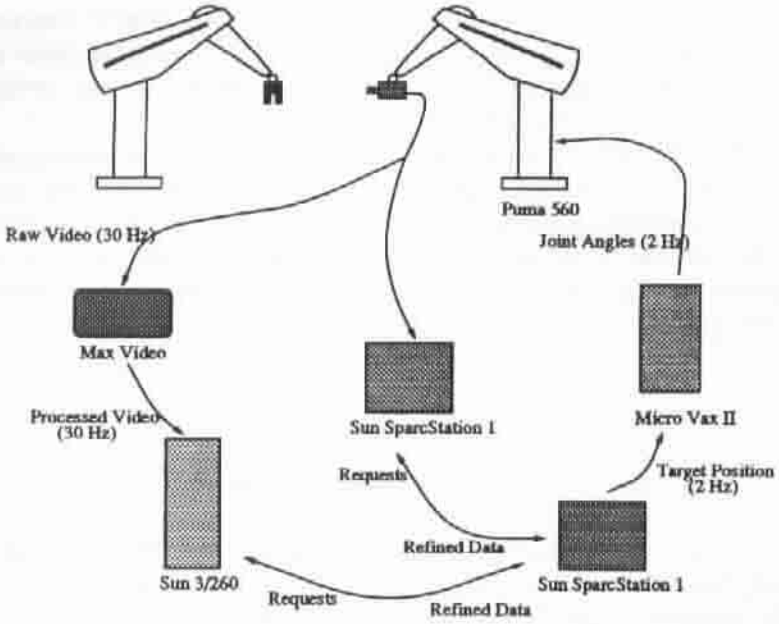
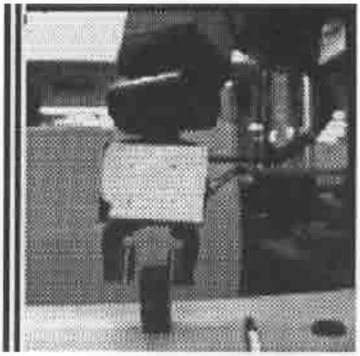


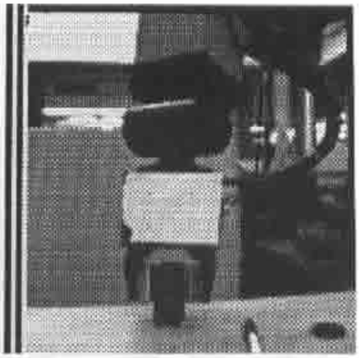
Fig. 28. The architecture of the system.

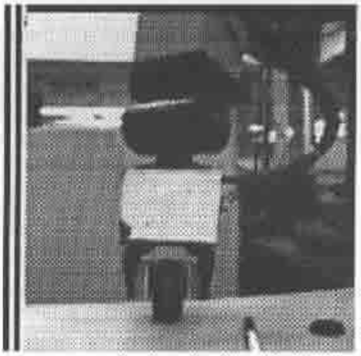
and the image understanding modules that reside on the other two machines. The output from the *thinking* modules is typicaly in the form of reporting states with the associated uncertainty and position control vectors to be supplied to the observer robot for relocation depending on the current state of the DEDS automata. The design exhibits modularity, the low-level event identification processes and the high-level 'thinker' and controller reside on separate entities. Thus futute modifications and enhancements could be coded and executed in a simple and modular fashion. Enhanced low-level modules for segmentation and 2D understanding of the image and to accommodate different kinds of hands could be coded within the inner-loop computer module. Different DEDS machines for different task descriptions are to coded within the 'thinker' module. Control vector generation could be modified within the procedure that supplies position control vectors to the observer manipulator.

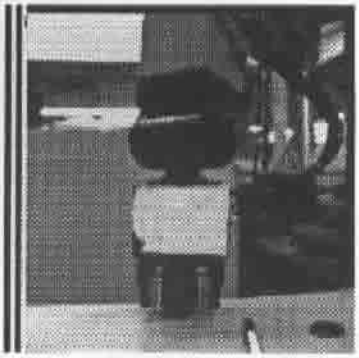
7.2. THE EXPERIMENTS

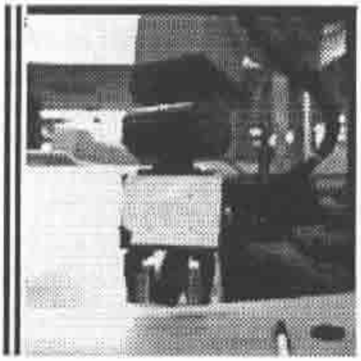
A number of experiments were performed with the lord gripper doing different manipulating action an a set of different objects. The whole system is tested by implementing automatons for recognizing the different actions under uncertainty and reporting on them, in addition to performing the necessary tracking movements, in real time. Thus testing both the low-level identification mechanisms and the high-level formulation.


Hand and Objects in Scene
Probability = 0.957878


Hand enclosing an Object
Probability = 0.962517


Hand enclosing an Object
Probability = 0.926735


Hand enclosing an Object
Probability = 0.994327


Hand is lifting an object
Probability = 0.918423

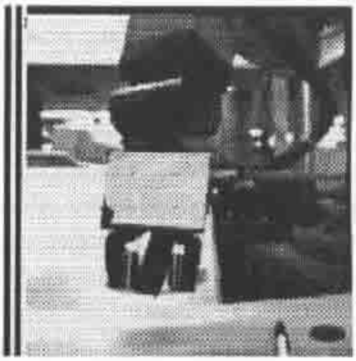
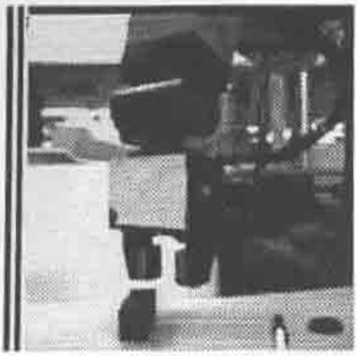
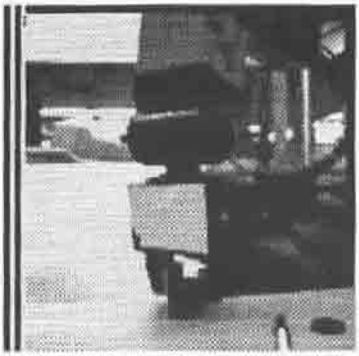
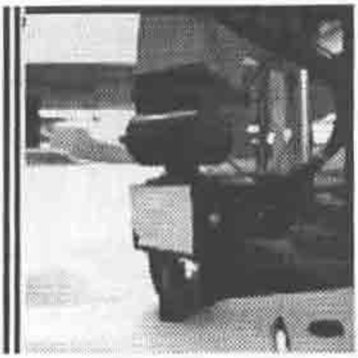

Hand is lifting an object
Probability = 0.972103

Fig. 29. A manipulation sequence (observer view) (1).


Hand alone in scene
Probability = 0.897626


Hand enclosing an object
Probability = 0.987823


Hand rotating an object
Probability = 0.912675

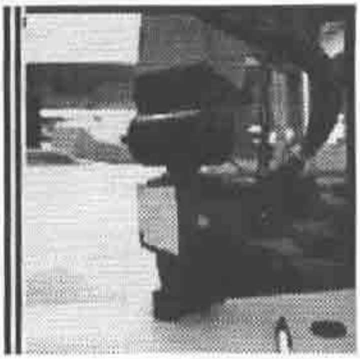

Hand rotating an object
Probability = 0.994534

Fig. 30. A manipulation sequence (observer view) (2).

Tracking is performed for some features on the gripper, using the MaxVideo system. The visual tracking system works in real time and a position control vector is supplied to the observer manipulator. The 2D uncertainty levels were tested. Feature extraction with uncertainty is performed using different noise levels as shown in Section 5, the enclosing 'envelopes' were determined for the mechanical system, the rejection algorithms are completed and utilized. The refined and recovered 3D distribution of uncertainties are used for navigating the automaton and asserting state transitions.

Some snap shots depicting the observer view, within an experiment that involves grasping, lifting and screwing is shown in Figures 29 and 30. The corresponding observer state output is written underneath each image and the corresponding uncertainty in recovered and displayed. The configuration of the

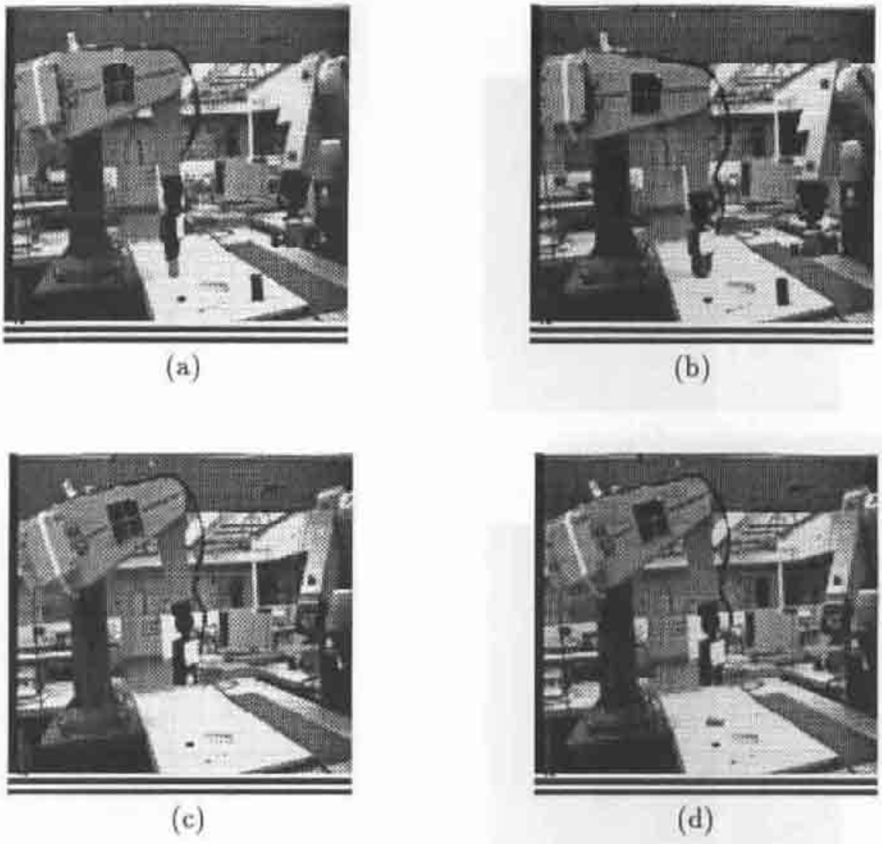


Fig. 31. The observer and manipulation agent configuration.

manipulating agent workspace and the observer is shown in some snap shots in Figure 31.

Figures 32 and 33 illustrate another manipulation sequence. In that sequence the lord gripper manipulates a set of objects laid on a table. 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 34 and 35 illustrate another manipulation experiment. In that sequence the hand tries to insert a peg in a hole. The screen structure is the same as for the previous experiment. 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.

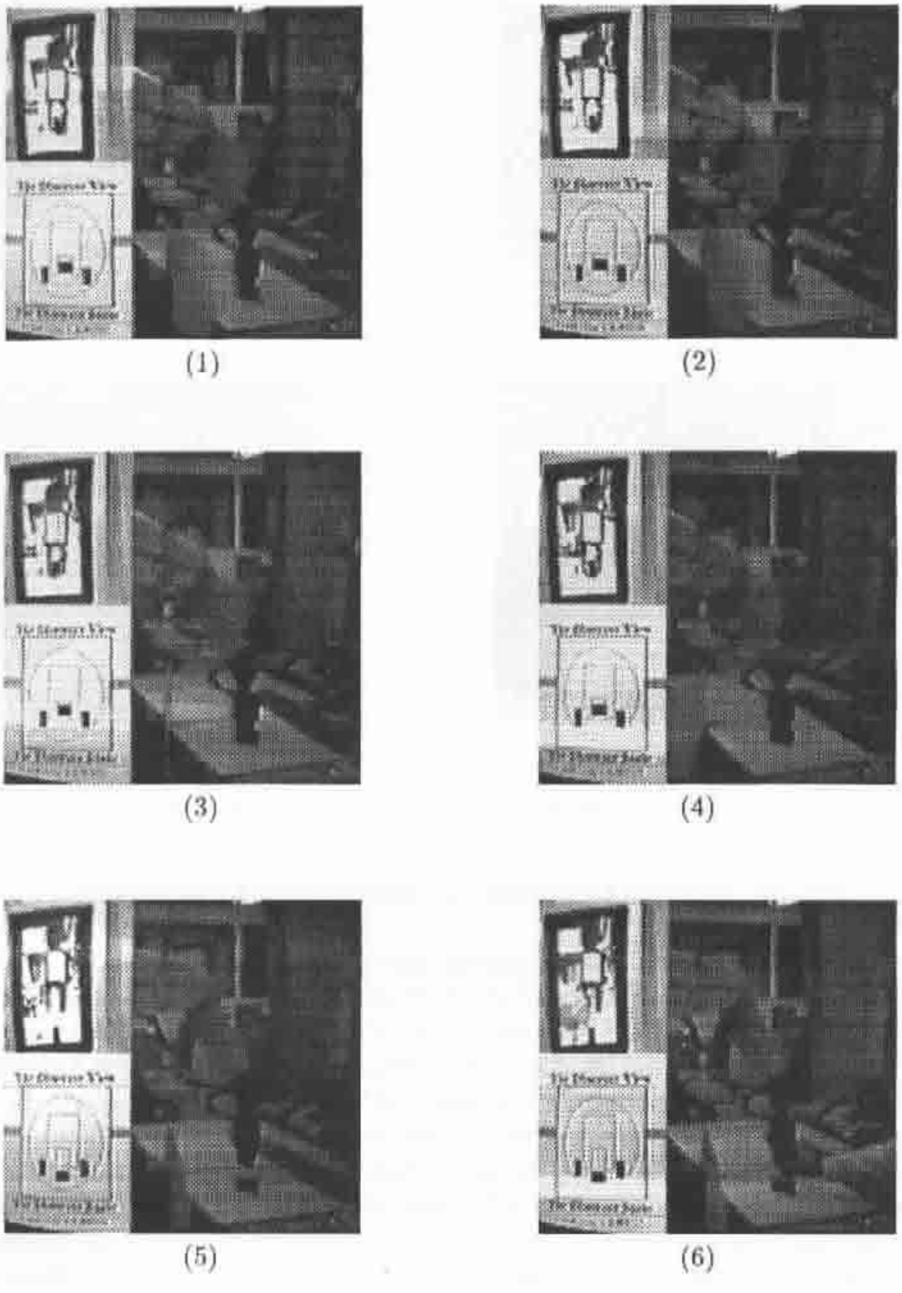


Fig. 32. Observer state and view (1).

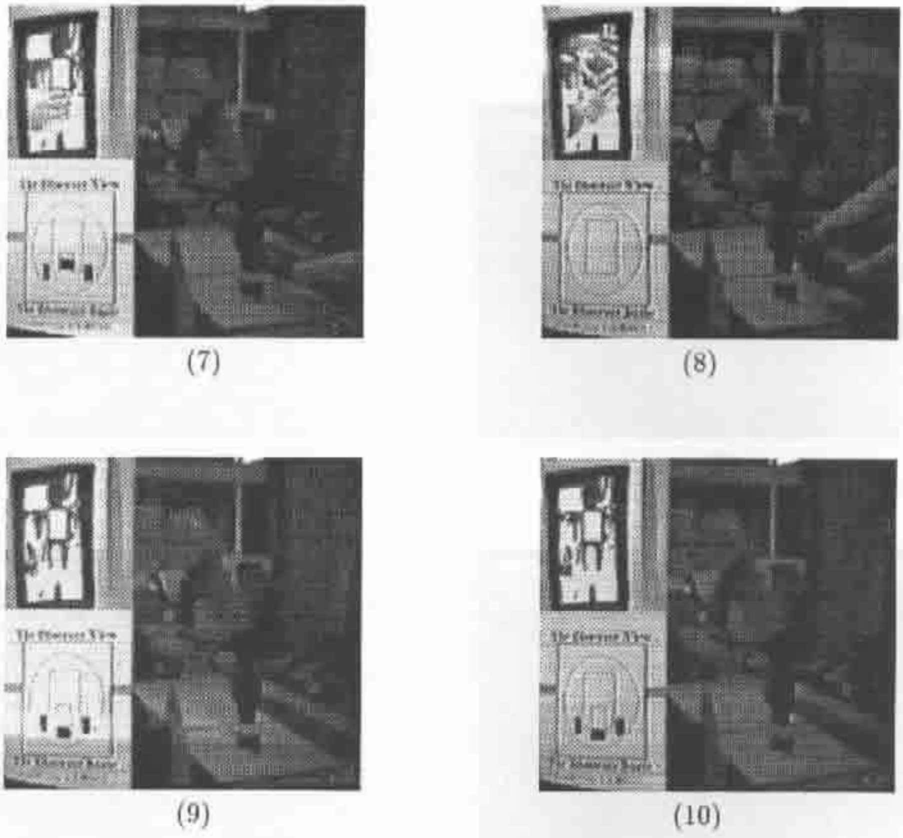


Fig. 33. Observer state and view (2).

8. Conclusions and Future Work

8.1. CONCLUSIONS

We have proposed a new approach for solving the problem of observing a moving agent. Intelligent observation and recognition of events is performed as opposed to simple tracking. In particular, we described a system for observing a manipulation process. Our approach uses the formulation of discrete event dynamic systems (DEDS) as a high-level model for the hybrid evolution of the dynamic scene. The proposed system utilizes the a priori knowledge about the domain of the manipulation actions in order to achieve efficiency and practicality. Task models are built by the coarse quantization of the visually observable manipulation actions and constructing a DEDS automaton description. The high level formulation allows for *recognizing* and *reporting* the visual system state as a symbolic description of the observed tasks.

The proposed formulation takes into consideration the presence of uncertainties in the observed behaviour of the system. The uncertainties are utilized in order to

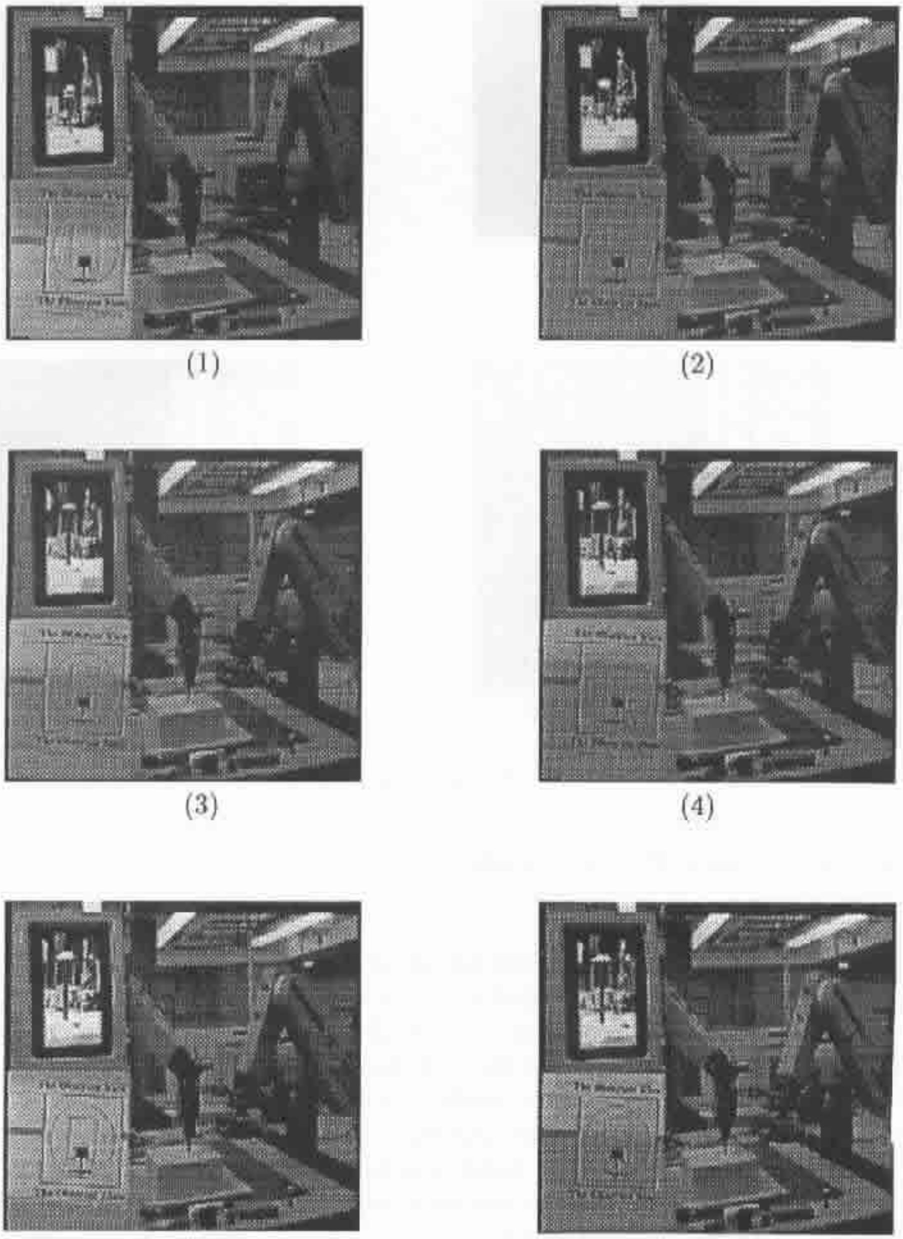
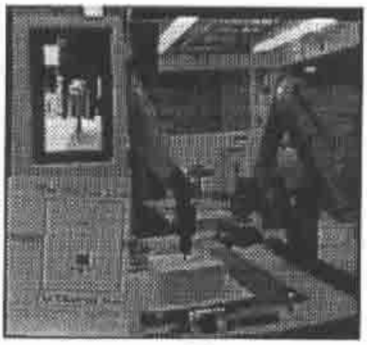
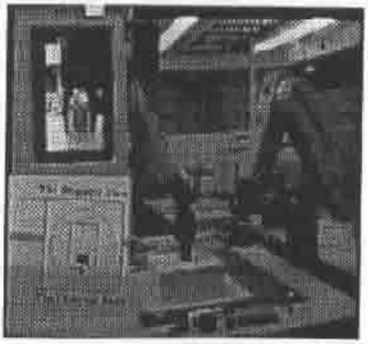
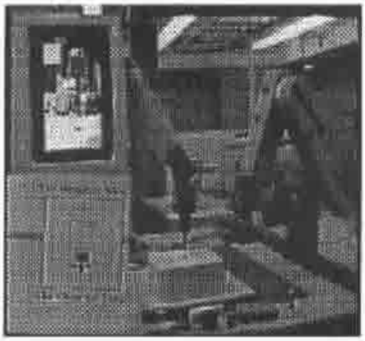
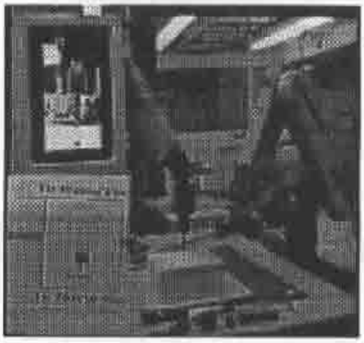


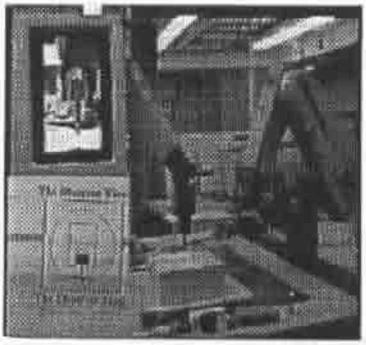
Fig. 34. Observer state and view (3).


(7)


(8)


(9)


(10)


(11)

(12)

Fig. 35. Observer state and view (4).

achieve robustness and to allow for correcting the observer's actions. Asserting transitions within the state description of the visual tasks is based on the recovered values of the observed parameters and the associated world uncertainties. The process develops coarse quantization of the visual manipulation actions in order to attain an active, adaptive and goal-directed sensing and reporting mechanism. Discrete aspects of the observation process are exploited in order to attach a meaningful symbolic interpretation of the observed task at different instances of the visual process. The formulation is flexible, since the quantization thresholds between different states can be tuned as the observed task requires. Continuous aspects of the process are also preserved as the relevant parameters are observed as the agent moves.

We started by describing the automaton model of a discrete event dynamic system as applied to visual observation of manipulation processes, and the observer construction mechanisms. We then developed efficient 2D and 3D low-level event-identification mechanisms for determining different manipulation actions in the system and for moving the observer. Next, we defined and constructed different levels for converting the raw 2D image data models into meaningful 3D descriptions of the world events. The formulation computes and refines uncertainty models resulting from errors in the 2D and 3D recovery mechanisms and the agent movements. The formulation allows the observer to navigate the decision making automata in real time with a stable behaviour through the state space trajectory and thus assert world events and transitions utilizing the developed framework. We also discuss some techniques for the automatic building of observer automatons.

8.2. APPLICATIONS

The developed observer model could be used for a variety of visual observation tasks within many domains. The approach lends itself to be a practical and feasible solution that uses existing information in a robust and modular fashion. The work examines closely the possibilities for errors, mistakes and uncertainties in the manipulation system, observer construction process and event identification mechanisms. Ambiguities are allowed to develop and are resolved after finite time, recovery mechanisms are devised too. Theoretical and experimental aspects of the work supports adopting the framework as a new kind of basis for performing goal-directed sensing for many task-oriented recognition, inspection and observation of visual phenomena.

The observer could be used for autonomous observation and reporting of fully or semi-automated manufacturing tasks. The observer can determine error sequences within a task and report on them and possibly provide feedback to the manipulating robots. The framework could be used for intelligence gathering in hostile environments, where there exists some a priori knowledge about some aspects of the terrain and/or targets. The observer can be used for structured inaccessible workspaces where there exists elements that are potentially dangerous for humans, as in the case of piling up heaps of toxic waste. The thresholds for quantizing the framework could be tuned to allow for expanding the state space of the DEDS automaton for tasks that require precision in reporting, for example within a surgical operation that is performed by a robotic end effector.

The general framework of a controllable state machine, that is augmented by mechanisms for identifying and recovering the events under uncertainty, could be used for general observers. Manipulation observers are a subset, obstacle avoidance observers and planning observers are another class that our framework could be used for designing. In the next section we discuss the contributions of the work and some conclusions.

8.3. CONTRIBUTIONS

We see the major contributions of this work in the following areas:

- Defining a framework for intelligent and autonomous observation. The observer *recognizes* visual tasks, *understands* the meaning of the dynamic scene evolution and *reports* symbolically on the current visual state, it also repositions itself intelligently.
- Utilizing the existing knowledge about the environment for developing a *predictive* model of visual observation that delivers a goal-directed sensing mechanism in real time with guarantees for stability and observability.
- Constructing event identification mechanisms for recovery of 2D and 3D actions within the visual manipulation domain. The events are then used for asserting state transitions within the DEDS automaton.
- Modeling and using visual uncertainties for recovering the 3D world event uncertainties and to assert and report on *distinct* and *discrete* visual states, while preserving and using the continuous dynamics of the system.
- Utilizing the computed world uncertainties for asserting state transitions, navigate the observer automaton, backtracking, performing the necessary tracking actions and error recovery.

In the next section we examine extension ideas and future research opportunities.

8.4. EXTENSIONS AND FUTURE RESEARCH

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. The system can be made more 'modular' by constructing a general automaton model of a discrete event

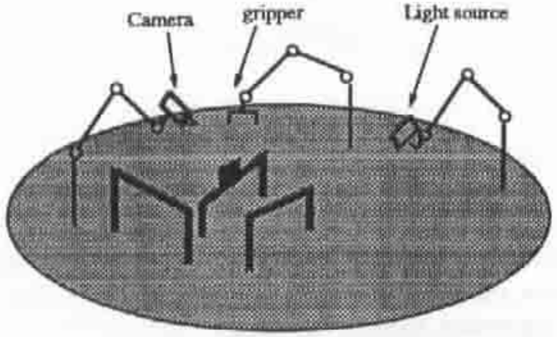


Fig. 36. Light source location as a controllable event.

would be necessary. Light sources and other sensors could be considered as types of resources, that is, controllable events. Figure 36 depicts such a situation where the light source position is contrallable.

The idea of DEDS as skeletons for observation under uncertainty can be explored further to allow for various other visual tasks. We discussed observing manipulation as a subset of observing moving agents and/or environments, however, similar formulation can be described for other tasks, like recognizing stationary objects with optimal observation costs, i.e., minimal motion events. Perturbation analysis [12, 44, 82] can be performed for the average task behaviour of frequent visual events within a specified manipulation domain. Disappearing objects and partially occluded objects can also be recognized optimally using the proposed scheme, using time proximity as another dimension for asserting the identity of different targets, that is, allow recognition and/or tracking to be completed within a pre-specified, task-dependent time frame. Delayed computation decision states could be used to augment the system. At those states decisions would have to be made regarding whether to go on with a specific computation, or to do it with a certain resolution. Observers could be formulated for mobile agents working in an environment. Different types of observers would be needed and would probably have to cooperate with each other and with the agents.

Appendix

A 3D RECOVERY ALGORITHM

One can model an arbitrary 3D motion in terms of stationary-scene/moving-viewer as shown previously in Figure 12. The optical flow at the image plane can be related to the 3D world as indicated by the following pair of equations (in case of a planar surface), for each point (x, y) in the image plane

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

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

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, p and q are the planar surface orientations. The situation becomes, for each point, two equations in eight unknowns, namely, the scaled translational velocities V_X/Z_o, V_Y/Z_o and V_Z/Z_o, the rotational velocities Ω_X, Ω_Y and Ω_Z and the orientations p and q. Differential methods could be used to solve those equations by differentiating the flow field and by using approximate methods to find the flow field derivatives. The existing methods for computing the derivatives of the flow field usually do not produce accurate results. Our algorithm uses a discrete method instead, i.e., the vectors at a number of points in the plane is determined and the problem reduces to solving a system of nonlinear equations.

It should be noticed that the resulting system of equations is nonlinear, however, it has some linear properties. The rotational part, for example, is totally linear, also, for any combination of two spaces among the rotational, translational and slope spaces, the system becomes linear. For the system of equations to be consistent, we need the flow estimates for at least four points, in which case there will be eight equations in weight unknowns.

TWO-FRAME ALGORITHM

The algorithm takes as input the estimate of the flow vectors at a number of points $\geqslant 4$ obtained from motion between two images. It iterates updating the solution of each subspace by using the solution of the other two subspaces. Each update involves solving a linear system, thereby it requires to solve three linear systems to complete a single iteration. This process continues until the solution converges, or until no significant improvement is made. The algorithm proceeds as follows:

```
1. Set $p, q = 0$;
   Input the initial estimate for rotation;
   Solve the linear system for translation;
2. Use the translation and rotation from step 1;
   Solve the linear system for the slope;
3. Set $i = 1$;
```

```
While ($i \leqslant$ Max. Iterations) and (no convergence) Do
    Solve for the rotations using latest estimates of translations, $p$ and $q$;
    Solve for the translations using latest estimates of rotations, $p$ and $q$;
    Solve for $p, q$ using latest estimates of translations and rotations;
end While;
```

Complexity Analysis

As we mentioned earlier, one should notice in the equations relating the flow velocities with the slope, rotational and translational velocities that they are 'quasi-linear', if one can say so. The equations exhibit some linear properties. This suggests that a purely iterative technique for solving nonlinear equations might not be an excellent choice, since, the variables are linearly related in some way. To think of a way of 'inverting' the relations might be a good start, although to do that without a framework based on iterating and gravitating towards a solution is not a good idea. This makes one think of applying a method which converges faster than a purely iterative scheme like Newton's method. However, the complexity of Newton's method is determined by the complexity of computing the inverse Jacobian, which is of an order of N^3, or $N^{2.81}$ multiplications as the lower bound using Strassen's technique. In our case, since we have at least 8 equations in 8 unknowns, the complexity is of order $8^3 = 512$ multiplications at every iteration, and the method does not make any use of the fact that the set of equations at hand exhibits some linear properties. The algorithm proposed, on the other hand, makes very good use of the fact that there are some linearity in the equations, by inverting the set of relations for each subspace at every iteration. The complexity at every iteration is of the order of the complexity of computing the pseudo-inverse which is of the order of $(3^3 + 3^3 + 2^3)$ multiplications at each iteration, where the first 3 comes from solving the system for the rotational variables, the second 3 is for the translations, the last 2 is for p and q. This is equal to 62 multiplications at every iteration, which is significantly less than the 512 multiplications in a method like Newton's for example. It was noticed that the algorithm converged to solution in a very small number of iterations for most experiments we have conducted so far. The maximum number of iterations was 6.

Using the latest solution obtained from the two-frame analysis as the initial condition for the next two-frame problem in the image sequence would further decrease the complexity, as the next set of parameters would, most probably, be close in values to the current parameters, thus the number of iterations needed to converge to the new solution would decrease significantly.

Observations

- The algorithm is not sensitive to the initial condition of the orientation parameters. The plane is simply assumed to be a frontal one at the beginning. The slope parameters evolves with iterations.
- The algorithm is sensitive to input noise just like other existing algorithms, some experiments shows the sensitivity with respect to the change of viewing angle. Similarly, the algorithm performs better for a large number of points that are evently distributed throughout the planar surface, than it does for clustered, smaller number of image points.
- It is proven that there exists dual solutions for such systems. However, if our method gravitates towards a 'fixed point' in the solution space we can find the other explicitly in terms of the first one from the relations given by Waxman and Ullman [92].

MULTI-FRAME ALGORITHM

The ordinary differential equations that describe the evolution of motion and structure parameters are used to fing the expression for the expected parameter change in terms of the previous parameter estimates. The expected change and the old estimates are then used to predict the current motion and structure parameters.

At time instant t, the planar surface equation is described by

$$Z = pX + qY + Z_o.$$

To compute the change in the structure parameters during the time interval $\mathrm{d}t$, we differentiate the above equation to get

$$\frac{\mathrm{d}Z}{\mathrm{d}t} = p\,\frac{\mathrm{d}X}{\mathrm{d}t} + X\,\frac{\mathrm{d}p}{\mathrm{d}t} + q\,\frac{\mathrm{d}Y}{\mathrm{d}t} + Y\,\frac{\mathrm{d}q}{\mathrm{d}t} + \frac{\mathrm{d}Z_o}{\mathrm{d}t}.$$

The time derivatives of (X, Y, Z) in the above expression are given by the three components of the vector $-(\mathbf{V} + \mathbf{\Omega} \times \mathbf{R})$ that represent the relative motion of the object with respect to the camera. Substituting these components for the derivatives and the expression $pX + qY + Z_o$ for Z we can get the exact differentials for the slopes and Z_o as

$$\begin{aligned}
\mathrm{d}Z_o &= Z_o\big[(\Omega_Y + V_X)p - (\Omega_X - V_Y)q - V_Z\big]\,\mathrm{d}t,\\
\mathrm{d}p &= \big[p(\Omega_Y p - \Omega_X q) + (\Omega_Y + \Omega_Z q)\big]\,\mathrm{d}t,\\
\mathrm{d}q &= \big[q(\Omega_Y p - \Omega_X q) - (\Omega_X + \Omega_Z p)\big]\,\mathrm{d}t.
\end{aligned}$$

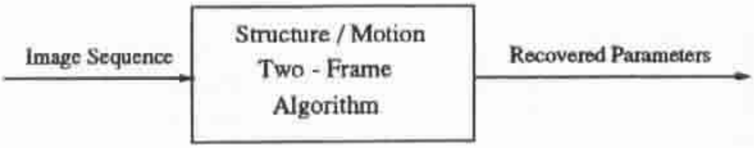


Fig. 37. Two-frame algorithm.

Using the above relations, we can compute the new structure parameters at time $t + \mathrm{d}t$ as

$$p' = p + \mathrm{d}p, \qquad q' = q + \mathrm{d}q \quad \text{and} \quad Z'_o = Z_o + \mathrm{d}Z_o.$$

Thus the slope parameters evolve at time $t + \mathrm{d}t$ as follows

$$\begin{bmatrix} p' \\ q' \end{bmatrix} = \begin{bmatrix} p \\ q \end{bmatrix} + \begin{bmatrix} \Omega_Y p - \Omega_X q & \Omega_Z & \Omega_Y \\ -\Omega_Z & \Omega_Y p - \Omega_X q & -\Omega_X \end{bmatrix} \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} \mathrm{d}t.$$

The new translational velocity $\mathbf{V}'$ at time $t + \mathrm{d}t$ can be found in the absence of accelerations from

$$\mathbf{V}' = \mathbf{V} + \mathbf{V} \times \mathbf{\Omega}\,\mathrm{d}t.$$

Dividing V' by Z'_o we get the new expected scaled translational velocity components at time $t + \mathrm{d}t$ as follows

$$\begin{bmatrix} V'_X \\ V'_Y \\ V'_Z \end{bmatrix} = \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} + \begin{bmatrix} -s & \Omega_Z & \Omega_Y \\ -\Omega_Z & -s & \Omega_X \\ \Omega_Y & -\Omega_X & -s \end{bmatrix} \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \mathrm{d}t$$

where s is expressed as follows

$$s = (\Omega_Y + V_X)p - (\Omega_X - V_Y)q - V_Z.$$

The expected rotational parameters at time $t + \mathrm{d}t$ remain equal to their values at time t since

$$\mathbf{\Omega}' = \mathbf{\Omega} + \mathbf{\Omega} \times \mathbf{\Omega}\,\mathrm{d}t = \mathbf{\Omega}$$

and thus

$$\left(\Omega'_X, \Omega'_Y, \Omega'_Z\right) = (\Omega_X, \Omega_Y, \Omega_Z).$$

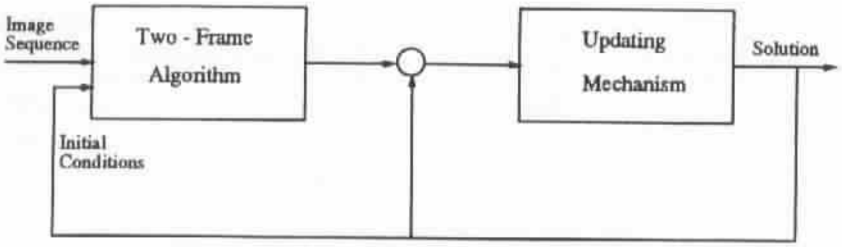


Fig. 38. Multi-frame algorithm.

Our first multi-frame algorithm uses a weighted average of the expected parameters at time $t + \mathrm{d}t$ from the above equations and the calculated parameters using the two-frame iterative algorithm as the solution at time $t + \mathrm{d}t$, and continues in the same way until the end of the frame sequence. Thus it keeps track of the past history of parametric evolution. We further develop the first multi-frame algorithm to exploit the temporal coherence of 3D motion. We develop the ordinary differential equations which describe the evolution of motion and structure in terms of the current motion/structure and the two-dimensional flow vectors in the image plane. We assume that the 3D motion is piecewise uniform in time, i.e., $\dot{\mathbf{\Omega}} = \dot{\mathbf{V}} = \mathbf{0}$. We then use the equations expressing the time derivative of the slope derived above and the fact that the derivative of the rotational velocities is zero and develop the following expressions for the scaled translational velocities and the depth Z_o

$$\frac{\mathrm{d}V'_X}{\mathrm{d}t} = -V'_X \frac{1}{Z_o} \frac{\mathrm{d}Z_o}{\mathrm{d}t}, \qquad \frac{\mathrm{d}V'_Y}{\mathrm{d}t} = -V'_Y \frac{1}{Z_o} \frac{\mathrm{d}Z_o}{\mathrm{d}t}$$

and

$$\frac{\mathrm{d}V'_Z}{\mathrm{d}t} = -V'_Z \frac{1}{Z_o} \frac{\mathrm{d}Z_o}{\mathrm{d}t}, \qquad \frac{1}{Z_o} \frac{\mathrm{d}Z_o}{\mathrm{d}t} = -V'_Z - pv_x - qv_y.$$

The extended Kalman filter can then be used to update the solution of the differential equations. The behaviour of the two-frame algorithm and the multi-frame algorithm can be conceptualized as a control system a shown in Figures 37 and 38. Parallel implementations could be designed for the system, thus solving for the structure – motion parameters for each surface separately. In fact, solving the linear system at each iteration could also be parallelized. Extra processing is needed to segment the polyhedra-like hand into separate planar surfaces.

References

1. Abdou, I. E. and Pratt, W. K.: Quantitative design and evaluation of enhancement/thresholding edge detectors, *Proc. IEEE* **67**(5) (1979).

2. Alameldin, T.: Visualization of 3D workspaces, PhD Thesis, Computer and Information Science Department, University of Pennsylvania, 1991.
3. Aloimonos, J. and Bandyopadhyay, A.: Active vision, in *Proc. 1st Int. Conf. on Computer Vision*, 1987.
4. Anandan, P.: A unified perspective on computational techniques for the measurement of visual motion, in *Proc. 1st Int. Conf. on Computer Vision*, 1987.
5. Anderson, H. L.: GRASP lab. camera systems and their effects on algorithms, Technical Report MS-CIS-88-85 and GRASP Lab. TR 161, University of Pennsylvania, 1988.
6. Andersson, M. and Lundquist, A.: Tracking lines in a stereo image sequence, Technical Report TRITA-NA-9116 and CVAP-87, Computational Vision and Active Perception Laboratory, Royal Institute of Technology, Stockholm, Sweden, 1991.
7. Bajcsy, R., Krotkov, E., and Mintz, M.: Models of errors and mistakes in machine perception, Technical Report MS-CIS-86-26 and GRASP Lab. TR 64, University of Pennsylvania, 1986.
8. Bajcsy, R.: Active perception, *Proc. IEEE* **76**(8) (1988).
9. Bajcsy, R. and Sobh, T. M.: A framework for observing a manipulation process, Technical Report MS-CIS-90-34 and GRASP Lab. TR 216, University of Pennsylvania, June 1990.
10. Bajcsy, R. and Sobh, T. M.: 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.
11. Ballard, D. H. and Ozcandarli, A.: Eye fixation and early vision: Kinetic depth, Technical Report, Computer Science Dept., University of Rochester, 1988.
12. Bak, P. and Chen, K.: Self-organized criticality, *Scientific American*, January 1991.
13. Barron, J. L., Jepson, A. D. and Tsotsos, J. K.: The feasibility of motion and structure from noisy time-varying image velocity information, *Int. Computer Vision*, December 1990.
14. Benahmed, N. M.: Camera calibration for dynamic environment, M.S. Thesis, Department of Electrical Engineering, University of Pennsylvania, 1989.
15. Benveniste, A. and Guernic, P. L.: Hybrid dynamical systems theory and the SIGNAL language, *IEEE Trans. Automatic Control* **35**(5) (1990).
16. Bergholm, F.: A theory on optical velocity fields and ambiguous motion of curves, in *Proc. 2nd Int. Conf. Computer Vision*, Florida, 1988.
17. Bergholm, F.: Motion from flow along contours: A note on robustness and ambiguous cases, in *Int. J. Computer Vision* **3** (1988), 395–415.
18. Binford, T. O.: Generic surface interpretation: Observability model, Technical Report, Robotics Laboratory, Stanford University, 1991.
19. Brave, Y. and Heymann, M.: Control of discrete event systems modeled as hierarchical state machines, Technical Report CIS-9012, Computer Science Department, TECHNION – Israel Institute of Technology, March 1991.
20. Burt, P. J., Yen, C., and Xu, X.: Multiresolution flow-through motion analysis, in *Proc. 1983 IEEE Conf. Computer Vision and Pattern Recognition.*
21. Burt, P. J., et al.: Object tracking with a moving camera, *IEEE Workshop on Visual Motion*, March 1989.
22. Cameron, A. and Wu, H.: A framework for sensory planning, in *Proc. Int. Conf. Automation, Robotics and Computer Vision (ICARCV'90)*, Singapore, September 1990.
23. Cao, X.: The predictability of discrete event systems, in *Proc. 27th Conf. on Decision and Control*, December 1988.
24. Carlsson, S. and Eklundh, J.: Object detection using model based prediction and motion parallax, in *Proc. 1st European Conf. on Computer Vision*, Antibes, France, April 1990.

25. Chase, C., Serrano, J., and Ramadge, P.: Periodicity and chaos from switched flow systems: Contrasting examples of discretely controlled continuous system, Technical Report, Department of Electrical Engineering, Princeton University, January 1991.
26. Chaochen, Z., Hoare, C. A. R., and Ravn, A. P.: A calculus of durations, Technical Report, Programming Research Group, Computing Laboratory, Oxford University, June 1991.
27. Chaumette, F. and Rives, P.: Vision-based-control for robotic tasks, in *Proc. IEEE Int. Workshop on Intelligent Motion Control*, Vol. 2, August 1990, pp. 395–400.
28. Chung, J., Liu, J. W. S., and Lin, K.: Scheduling periodic jobs that allow imprecise results, *IEEE Trans. on Computers* **39**(9) (1990).
29. Clark, M. S.: Robot-based real-time motion tracker, in *Proc. 2nd SPIE Conf. on Sensor Fusion*, November 1989.
30. Cooper, J. and Kitchen, L.: Multi-agent motion segmentation for real-time task directed vision, in *Proc. 4th Australian Joint Conference on Artificial Intelligence*, Perth, Australia, November 1990.
31. Deutsch, E. S. and Fram, J. R.: A quantitative study of the orientation bias of some edge detector schemes, in *IEEE Trans. Comput.*, C-27, **3**, March 1978.
32. Fischler, M. and Bolles, R. C.: *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, Readings in Computer Vision, Morgan Kaufmann, 1987.
33. Fram, J. R. and Deutsch, E. S.: On the quantitative evaluation of edge detection schemes and their comparison with human performance, in *IEEE Trans. Comput.*, C-24, **6**, June 1975.
34. Grzywacz, N. M. and Hildreth, E. C.: The Incremental rigidity scheme for recovering structure from motion: Position vs. velocity based formulations, MIT A.I. Memo No. 845, October 1985.
35. Guernic, P. L. et al.: Programming real time applications with SIGNAL, Technical Report, IRISA, Rennes, France, 1990.
36. Guernic, P. L. and Gautier, T.: Data-flow to von Neumann: the SIGNAL Approach, Technical Report 1229, INRIA, Rennes, France, May 1990.
37. Hager, G. D.: Active reduction of uncertainty in multi-sensor systems, PhD Thesis, Computer and Information Science Department, University of Pennsylvania, July 1988.
38. Haralick, R. M. and Lee, J. S. J.: Context dependent edge detection and evaluation, *Pattern Recognition* **23**(1/2) (1990), 1–19.
39. Heeger, D. J.: Models for motion perception, PhD Thesis, Computer and Ination. Science Department, University of Pennsylvania, September 1987.
40. Heel, J.: Dynamic Motion Vision, in *Proc. SPIE Conf. on Computer Vision*, November 1989.
41. Hervé, J., Cucka, P., and Sharma, R.: Quantitative visual control of a robot manipulator, in *Proc. DARPA Image Understanding Workshop*, September 1990.
42. Hervé, J., Sharma, R. and Cucka, P.: Qualitative coordination of a robot hand/eye system, CAR-TR-516, Center for Automation Research, University of Maryland, 1990.
43. Heymann, M.: Concurrency and discrete event control, in *Proc. IEEE Conf. on Decision and Control*, December 1989.
44. Ho, Y.: Performance evaluation and perturbation analysis of discrete event dynamic systems, in *IEEE Trans. on Automatic Control*, July 1987.
45. Hopcroft, J. E. and Ullman, J. D.: *Introducion to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.
46. Horn, B. K. P. and Schunck, B. G.: Determining optical flow, *Artificial Intelligence* **17** (1981), 185–203.
47. Horn, B. K. P.: *Robot Vision*, McGraw-Hill, 1987.

48. Hsu, A. et al.: The design of Platoon maneuver protocols for IVHS, PATH Research Report UCB-ITS-PRR-91-6, Institute of Transportation Studies, University of California at Berkeley, April 1991.
49. Inan, K. and Varaiya, P.: Finitely recursive process models for discrete event systems, *IEEE Trans. on Automatic Control* **33**(7) (1988).
50. Izaguirce, A., Pu, P., and Summers, J.: A new development in camera calibration: Calibrating a pair of mobile cameras, in *Proc. Int. Conf. on Robotics and Automation*, 1985, pp. 74–79.
51. Karaaslan, U., Varaiya, P., and Walrand, J.: Two proposals to improve freeway traffic flow, PATH Research Report UCB-ITS-PRR-90-6, Institute of Transportation Studies, University of California at Berkeley, December 1990.
52. Keren, D., Peleg, S., and Shmuel, A.: Accurate hierarchical estimation of optic flow, TR-89-9, Department of Computer Science, The Hebrew University of Jerusalem, June 1989.
53. Kitchen, L. and Rosenfeld, A.: Edge evaluation using local edge coherence, in *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-11, **9**, September 1981.
54. Kohavi, Z.: *Switching and Finite Automata Theory*, McGraw-Hill, 1979.
55. Kohn, W. and Nerode, A.: An autonomous control theory, Personal correspondence.
56. Krotkov, E.: Results in finding edges and corners in images using the first directional derivative, Technical Report MS-CIS-85-14 and GRASP Lab. TR 37, University of Pennsylvania, 1985.
57. Kuniyoshi, Y., Inaba, M., and Inoue, H.: Teaching by Showing: Generating Robot Programs by Visual Observation of Human Performance, Department of Mechanical Engineering, The University of Tokyo, Technical Report, 1990.
58. Lavignon, J. and Shoham, Y.: Temporal Automata, Technical Report STAN-CS-90-1325, Department of Computer Science, Stanford University, August 1990.
59. Lee, S. W. and Wohn, K.: Tracking moving objects by a robot-held camera using a pyramid-based image processor, Technical Report MS-CIS-88-97 and GRASP Lab. TR 168, University of Pennsylvania, 1988.
60. Lewis, H. R. and Papadimitriou, C. H.: *Elements of the Theory of Computation*, Prentice-Hall, 1981.
61. Li, Y. and Wonham, W. M.: Controllability and observability in the state-feedback control of discrete-event systems, *Proc. 27th Conf. on Decision and Control*, 1988.
62. Lobo, N. and Tsotsos, J. K.: Shape information from image motion: What can we compute from three points?, Technical Report RBCV-TR-89-32, Department of Computer Science, University of Toronto, December 1989.
63. Longuet-Higgins, H. C. and Prazdny, K.: The interpretation of a moving retinal image, in *Proc. Royal Society of London B*, 208, 385–397.
64. Nerode, A. and Remmel, J. B.: A model for hybrid systems, Presented at the Hybrid Systems Workshop, Mathematical Sciences Institute, Cornell University, May 1991.
65. Nielson, L. and Sparr, G.: Projective area-invariants as an extension of the cross-ratio, Technical Report, Lund Institute of Technology, Sweden.
66. Osherson, D. N., Stob, M., and Weinstein, S.: *Systems that Learn*, MIT Press, 1986.
67. Özveren, C. M. and Willsky, A. S.: Aggregation and multi-level control in discrete event dynamic systems, Technical Report CICS-P-199, Center for Intelligent Control Systems, Massachusetts Institute of Technology, March 1990.
68. Özveren, C. M.: Analysis and control of discrete event dynamic systems: A state space approach, PhD Thesis, Massachusetts Institute of Technology, August 1989.
69. Peli, T. and Malah, D.: A study of edge detection algorithms, *Computer Graphics and Image Processing* **20** (1982), 1–21.

70. Ramadge, P. J. and Wonham, W. M.: Supervisory control of a class of discrete event processes, *SIAM J. Control and Optimization*, January 1987.
71. Ramadge, P. J. and Wonham, W. M.: Modular feedback logic for discrete event systems, in *SIAM J. Control and Optimization*, September 1987.
72. Ravn, A. P. and Rischel, H.: Requirements capture for embedded real-time systems, in *IMACS Symposium MCTS, Vol. 2*, France, March 1991, pp. 147–152.
73. Révész, G. E.: *Introduction to Formal Languages*, McGraw-Hill, 1985.
74. Schapire, R. E.: The design and analysis of efficient learning algorithms, PhD Thesis, Massachusetts Institute of Technology, February 1991.
75. Sobh, T. M. and Bajcsy, R.: A model for observing a moving agent, in *Proc. 4th Int. Workshop on Intelligent Robots and Systems (IROS'91)*, Osaka, Japan, November 1991.
76. Sobh, T. M. and Bajcsy, R.: Visual observation of a moving agent, in *Proc. European Robotics and Intelligent Systems Conference (EURISCON'91)*, Corfu, Greece, June 1991 and presented at the *12th Int. Joint Conference on Artificial Intelligence (IJ-CAI)*, Workshop on Dynamic Scene Understanding Sydney, Australia, August 1991.
77. Sobh, T. M.: *A framework for visual observation*, Technical Report MS-CIS-91-36 and GRASP Lab. TR 261, Computer Science Dept., School of Engineering and Applied Science, University of Pennsylvania, May 1991.
78. Sobh, T. M. and Wohn, K.: Recovery of 3D motion and structure by temporal fusion, in *Proc. 2nd SPIE Conference on Sensor Fusion*, November 1989.
79. Sparr, G. and Nielsen, L.: Shape and mutual cross-ratios with applications to orientation problems, Technical Report, Lund Institute of Technology, Sweden, 1990.
80. Subbarao, M. and Waxman, A. M.: On the uniqueness of image flow solutions for planar surfaces in motion, CAR-TR-113, Center for Automation Research, University of Maryland, April 1985.
81. Sull, S. and Ahuja, N.: Segmentation, matching, and estimation of structure and motion of textured piecewise planar surfaces, in *Proc. IEEE Workshop on Visual Motion*, October 1991.
82. Suri, R.: Perturbation analysis: The state of the art and research issues explained via the GI/G/1 queue, *Proc. IEEE* (January 1989).
83. Tomasi, C. and Kanade, T.: Factoring image sequences into shape and motion, in *Proc. IEEE Workshop on Visual Motion*, October 1991.
84. Tomasi, C. and Kanade, T.: Shape and motion without depth, in CMU-CS-90-128, School of Computer Science, Carnegie Mellon University, May 1990.
85. Tsai, R. Y. and Huang, T. S.: Estimating three-dimensional motion parameters of a rigid planar patch, in *IEEE Trans. on Acoustics, Speech and Signal Processing*, ASSP **29**(6) (1981).
86. Tsai, R. Y.: An efficient and accurate camera calibration technique for 3D machine vision, IBM Report.
87. Ullman, S.: Analysis of visual motion by biology and computer systems, *IEEE Computer* (August 1981).
88. Ullman, S.: Maximizing Rigidity: The incremental recovery of 3D structure from rigid and rubbery motion, AI Memo 721, MIT AI Lab., 1983.
89. Varaiya, P. and Shladover, S. E.: Sketch of an IVHS Systems Architecture, PATH Research Report UCB-ITS-PRR-91-3, Institute of Transportation Studies, University of California at Berkeley, February 1991.
90. Vaz, A. F. and Wonham, W. M.: On supervisor reduction in discrete-event systems, Technical Report, Systems Control Group, Department of Electrical Engineering, University of Toronto, 1985.

91. Waxman, A. M. and Wohn, K.: Contour evolution, neighborhood deformation and global image flow: Planar surfaces in motion, *Int. J. Robotics Research* **4**(3) (1985), 95–108.

92. Waxman, A. M. and Ullman, S.: Surface structure and 3D motion from image flow: a kinematic analysis, CAR-TR-24, Center for Automation Research, University of Maryland, October 1983.

93. Weng, J., Huang, T. S. and Ahuja, N.: 3D motion estimation, understanding and prediction from noisy image sequences, *IEEE Trans.*, PAMI **9**(3) (1987).

94. Willner, Y. and Heymann M.: On supervisory control of concurrent discrete-event systems, Technical Report CIS-9009, Computer Science Department, TECHNION – Israel Institute of Technology, October 1990.

95. Wilson, R. and Granlund, G. H.: The uncertainty principle in image processing, *IEEE Trans.* PAMI **6**(6) (1984).

96. Wohn, K. and Maeng, S. R.: Real-time of estimation 2D motion for object tracking, in *Proc. SPIE Conf. on Intelligent Robotics*, November 1989.

97. Wu, H. and Cameron, A.: A bayesian decision theoretic approach for adaptive goal-directed sensing, Technical Report, Philips Laboratories, New York, May 1990.