Brenan J. McCarragher, Geir Hovland, Pavan Sikka, Peter Aigner, and David Austin

Hybrid Dynamic Modeling and Control of Constrained Manipulation Systems

Discrete event systems are presented as a powerful framework for a large number of robot control tasks. This paper presents a general description of the discrete event modeling and control synthesis for robot manipulation. Additionally, methods for the effective monitoring of the process based on the detection and identification of discrete events are given. The effectiveness and versatility of the approach are demonstrated through a wide variety of experiments. Applications are demonstrated in assembly, on-line training of robots, advanced perception capabilities, human-robot shared control and the understanding of human manipulation skills.

Keywords: robotic manipulation, discrete event control, assembly, hidden markov models

A significant issue in robotics and automated systems is the increased level of complexity competing against the need for simple systems for technology transfer to practical settings. The decision making and intelligent control of robotic tasks are becoming less and less tractable, and so less appealing to those unfamiliar with the technology. Additionally, process monitoring using signal processing and sensory perception techniques further mystifies the process of robot control. Strong need exists for a framework which simplifies both the control and monitoring elements of robotics, yet is powerful enough to address complicated tasks such as assembly.

This work demonstrates the use of hybrid dynamic systems to simplify the monitoring, control, interfacing and analysis of constrained manipulation tasks. The overall strategy uses a discrete event system framework providing a method for describing complex processes in a simple, yet formal, manner. Each discrete state encodes a continuous control law or decision maker. In this way, the complexity of control is simplified and tractable. Additionally, the process is monitored through the recognition of the discrete events, when the dynamics of the sensing signal is strongest, reducing the need for detailed discrete state monitoring. Moreover, the formal description enables performance analysis and verification through a variety of techniques.

Intuitively, a hybrid dynamic system consists of a discrete event system (DES) interacting with a continuous-time system. Usually, the discrete event system is a decision maker or controller and operates at an abstract level. The continuous-time system, therefore, is the actual plant executing the decisions of the DES. For our purposes, the continuous-

1070-9932/97/$10.00©1997 IEEE

time system will be the robot manipulator and its continuous controller. By contrast, then, the discrete event system will be a higher-level, abstract decision-maker which issues commands to the continuous-time system.

Hybrid dynamic systems have been used in wide-ranging applications, most notably in manufacturing systems and network protocols. For example, [12] use Petri nets for the hierarchical analysis of manufacturing systems and [9] develops optimal dispatching policies for elevator control systems using hybrid dynamic systems. Most of the research work in the area has been concerned with developing and proving control-theoretic ideas for specific classes of systems. Stiver and Antsaklis [33] and Gollu and Varaiya [13] have given general formulations for the modeling and analysis of hybrid dynamic systems. As well, Brockett [7] develops a general basis for the modeling of a wide range of motion control systems using hybrid dynamics. Several authors [19,28,32] have developed a number of techniques for the synthesis of hybrid dynamic controllers. It is on the strength of this, and other research, that we apply hybrid dynamic systems theory to the problem of robotic manipulation.

We will first present the hybrid dynamic modeling of robotic manipulation tasks. The modeling framework sets the foundation for the process monitoring and control of the tasks. Both the monitoring and the control take advantage of the discrete event structure to simplify the process. Experimental results are then given for the basic operation of the system.

One of the deficiencies in the literature is the application of hybrid dynamic systems to physical systems. As such, several constraints relevant to practical implementation are overlooked. In our research, we have concentrated on implementation to physical systems to address this imbalance. As such, we present results for the on-line training of assembly lines using task-level adaptive control; for the control of advanced sensory perception; for the sharing of control between a human operator and an otherwise autonomous system; and for the acquisition of human skill. Each of these examples takes significant advantage from the hybrid dynamic systems framework.

DISCRETE EVENT MODELING OF CONSTRAINED MANIPULATION

We will consider a constrained motion system which involves the motion of a polyhedral *workpiece* with possible constraints introduced by contact between the workpiece and a fixed polyhedral *environment*. Systems of this type are typical of assembly processes. In a previous work, McCarragher [25] demonstrates that all possible states of contact between two polyhedral parts in Cartesian space can be described as combinations of edge-edge and surface-vertex contacts. Hybrid dynamic modeling is particularly appropriate for assembly processes as there are a small number of possible contact configurations and, hence, we can dramatically reduce the complexity of the continuous time process by abstracting to a higher level.

We will consider a specific case of a hybrid dynamic system, consisting of a discrete event controller interfaced to a constrained motion system involving two polyhedral parts (as discussed above). The structure of the adopted hybrid dynamic model is as shown in Figure 1. The system consists of three parts, which are the continuous time plant, the discrete event controller, and the interface. Each component is detailed below, with the description tailored to the restricted constrained motion systems that we consider. Mathematically, the continuous time plant will be defined by a set of differential equations describing the motion of the workpiece. The discrete event controller is modeled as an automaton describing task-level decision making, whereas the interface serves as the communication between the continuous manipulation process and the decision maker. In many cases the continuous plant and the interface are combined and described using one discrete event model [13].

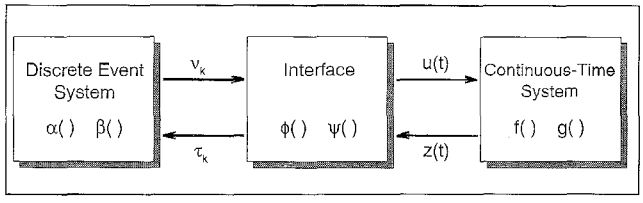


Figure 1. Block Diagram Representation of Hybrid Dynamic System.

The Continuous-Time System

Consider the general motion control of an object or workpiece. The equation of motion of the workpiece in free-space is given generally as

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t),\mathbf{u}(t)) \tag{1}$$

where $\mathbf{x}(t)$ is the continuous time state vector, and $\mathbf{u}(t)$ is the input vector. As the workpiece interacts with an inertially fixed environment, the free-space dynamics of Eq. 1 become constrained. For each edge-edge or surface-vertex contact, this constraint may be written as

$$g_j(\mathbf{x}(t)) = \mathbf{0} \tag{2}$$

where g_j is the constraint equation for the j^{th} edge-edge or surface-vertex contact. Distance functions are ideal candidates for g_j (e.g., the shortest distance between a surface and vertex). Note, Eq(2) is only valid when the j^{th} edge-edge or surface-vertex pair is actually in contact.

The Discrete Event System

The continuous time plant is controlled by a task-level, discrete event controller modeled as an automaton. The automaton is a quintuple, (S, E, C, α, β), where S is the finite set of states, E is the set of plant events, C is the set of controller events, $\alpha{:}S\times E\rightarrow S$ is the state transition function, and $\beta{:}S\rightarrow C$ is the output function. Each *state* in S, denoted γ_i, identifies a distinct state of contact between the workpiece and the environment. For this paper, the states will be taken as the possible combinations of edge-edge and surface-vertex contacts as discussed above. Each *controller event* in C, denoted v_k, is generated by the discrete event controller, whereas each *plant event* in E, denoted τ_k, is generated by the conditions in the continuous plant. For our purposes, plant events are generat-

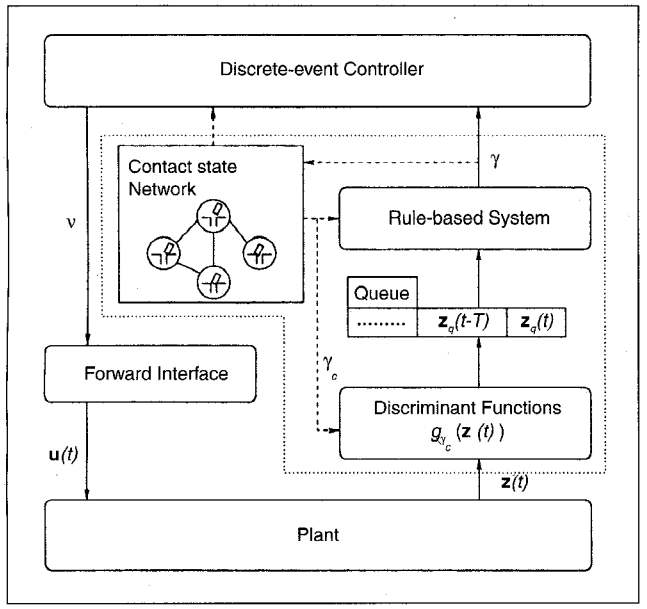


Figure 2. The process monitor within a hybrid system framework. The dotted line encloses the main functional blocks of the process monitor.

ed when the continuous-time system changes contact state. Thus, the definition of each event includes the edge-edge or surface-vertex pair involved and the type of contact state change. The contact state changes are *loss of contact, gain of contact*, and *over-force*. The over-force condition occurs when the system attempts to move the workpiece through the environment.

The dynamics of the discrete event controller are given generally by

$$\gamma_{k+1} = \alpha(\gamma_k, \tau_k) \tag{3}$$

$$\nu_k = \beta(\gamma_k) \tag{4}$$

where $\gamma_k \in S$, $\tau_k \in E$, and $\nu_k \in C$. The input and output signals of the discrete event controller are asynchronous sequences of events, rather than continuous time signals. The function α is functionally dependent on f and g as defined by the continuous-time system. Note that the state of the continuous-time system is $\mathbf{x}$, whereas γ is the state variable of the discrete event system and is dependent on $\mathbf{x}$.

The Interface

The interface is responsible for the communication between the plant and the controller, since these components cannot communicate directly due to the different types of signals being used. The interface consists of two maps ϕ and ψ. The first map ϕ converts each controller event into a plant input as follows

$$\mathbf{u}(t) = \phi(\nu_k) \quad t_k \le t \le t_{k+1} \tag{5}$$

where ν_k is the most recent controller event before time t. We require that $\mathbf{u}(t)$ be a piecewise constant plant input with possible discontinuities only when controller events occur. For synthesis purposes it is often easier to combine the control equations (4) and (5) such that the control input is directly a function of the discrete state

$$\mathbf{u}(t) = \phi(\beta(\gamma_k)) \quad t_k \le t \le t_{k+1} \tag{6}$$

The second map ψ converts the state space of the plant into the set of plant events.

$$\tau_k = \psi(\mathbf{z}(t)) \tag{7}$$

where the vector $\mathbf{z}$ is an observer and consists of the variables being used by the process monitor to observe the state of the plant. For example, in the planar case with force and changes in force being the variables of interest, $\mathbf{z}(t) = [F_x(t), F_y(t), \dot{F}_x(t), \dot{F}_y(t)]^T$, where F_x and F_y denote the planar components of force. Note that Eq.(7) does not imply that τ_k changes continuously as $\mathbf{z}(t)$ changes. The state space of the plant is partitioned into contiguous regions. The function ψ generates a new plant event only when the state first enters one of these regions. The map ψ is called a *process monitor*.

PROCESS MONITORING

Process monitoring is a key element of hybrid-dynamic systems. The process monitor corresponds to the map ψ of the interface and transforms the plant state $\mathbf{x}(t)$ through the observer $\mathbf{z}(t)$ into a discrete state γ, as per Eq.(7). It uses the temporal patterns of the observer $\mathbf{z}(t)$ associated with process state transitions to recognize the transitions in real-time, which is required for effective discrete-event control, error detection and error recovery. For example, in assembly tasks, a contact state transition can be characterized by patterns of variables representing the system state, such as the velocity of the object and the force acting on the object. In the following sections we present two complementary methods for process monitoring.

Rule-Based Process Monitor

Rule-based process monitoring exploits the sudden changes in the values of the observer $\mathbf{z}(t)$ associated with process state transitions. The temporal patterns of the observed $\mathbf{z}(t)$ associated with a process state transition are formulated as a set of rules which are then used to recognize the corresponding transition. Rule-based process monitoring overcomes several problems associated with approaches based on template matching and qualitative reasoning [21]. The rules are more general than templates which are based on the current value of the observer. Also, the rules are obtained by demonstration rather than qualitative reasoning methods which have the problem that consistent qualitative algebras are difficult to obtain [36].

The organization of the process monitor within the hybrid-dynamic framework is shown in Figure 2. The process monitor works in two steps. In the first step, the process monitor samples the variable $\mathbf{z}$ and then *quantises* or *discretises* it into the variable $\mathbf{z}_q$. For example, in an assembly task, $\mathbf{z}_q = [F_{xq}, F_{yq}, \dot{F}_{xq}, \dot{F}_{yq}]^t$. The discretisation is performed using discriminant functions g_{γ_c} based on the current contact state γ_c. The details of this step can be found in [30].

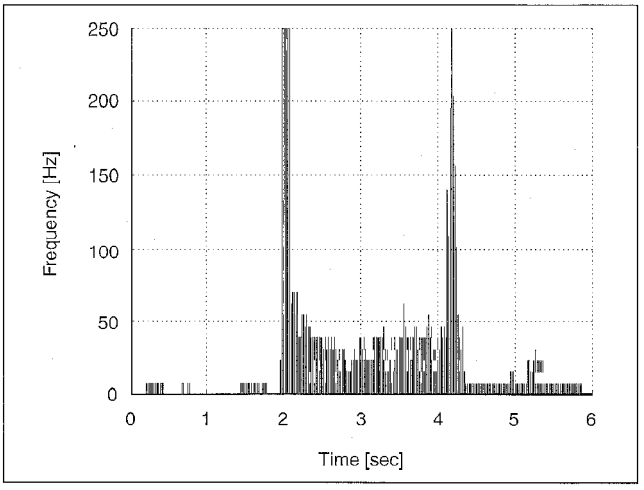


Figure 3. Spectrogram of the F_y force measurements for two events. The first event (dynamic forces) starts at t=2.0 sec followed by another relatively static event at t = 4.2 sec. The measured force was sampled at 500 Hz.

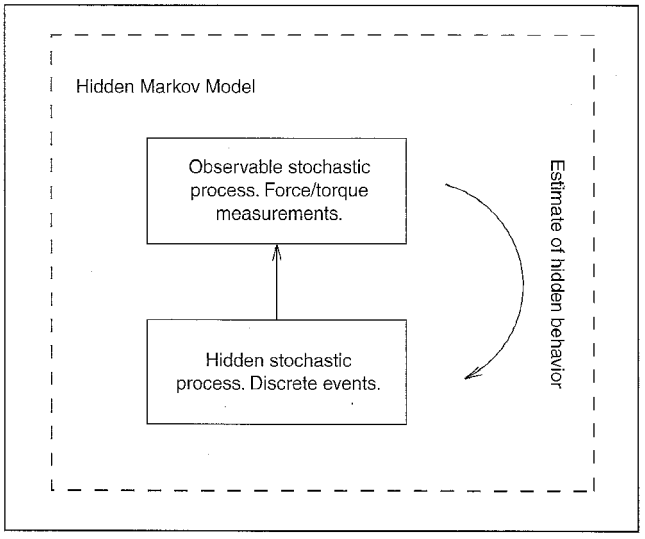


Figure 4. Illustration of an HMM. The underlying stochastic process can only be observed through another stochastic process.

The second step of the process monitor consists of using a rule-based system [3] to identify the process states and transitions based on a sequence of discretised variables. A rule is defined as a *condition-action pair* where the condition part specifies the conditions under which the rule can be applied and the action part specifies the actions that are taken when the rule is applied. For example, in the context of contact monitoring, the following general rule can be used for recognizing a contact state transition:

IF

$$\begin{aligned} \text{state} &= \gamma_1 && \textbf{AND} \\ \dot{F}_{xq}(t) &= d_1 && \textbf{AND} \\ \dot{F}_{xq}(t-T) &= d_1 && \textbf{AND} \\ \dot{F}_{xq}(t-2T) &= d_2 && \textbf{AND} \\ \dot{F}_{xq}(t-3T) &= d_3 && \textbf{AND} \\ \dot{F}_{xq}(t-4T) &= d_1 \end{aligned}$$

THEN

$$\text{state} = \gamma_2$$

Here, $d_1 \cdots d_3$ refer to the discrete values of the quantised variables, γ_1, $\gamma_2 \in S$ denote contact states and T denotes a system-dependent delay.

The rule shown above checks for a sequence of discretised values of the third component $\dot{F}_{xq}$ of the observer $\mathbf{z}(t)$ to recognize a transition from contact state γ_1 to γ_2. In general, for each contact state transition, similar rules will exist for each of the components of the observer $\mathbf{z}(t)$, This leads to a very general system for monitoring contact.

Whenever a new value of the discretised observer $\mathbf{z}_q(t)$ is obtained, the rule-base is matched to the temporal sequence $\{\cdots \mathbf{z}_q(t-T)\mathbf{z}_q(t)\}$ to obtain a set of matching rules. Each rule indicates a transition and so votes for that particular transition. If all the selected rules indicate the same transition, and the number of selected rules is equal to the number of components of the observer $\mathbf{z}(t)$ then that transition is chosen as the recognized transition; otherwise, no transition is selected and the process state remains unchanged. The rules are indexed by the contact state and so the matching process considers only those rules from the rule-base that are applicable for the current contact state γ_c. This greatly reduces the computational requirements of the process monitor and is a significant advantage of using the hybrid dynamic framework model. The details of this approach can be found in [31].

HMM Process Monitor

A complementary process monitor, based on Hidden Markov Models (HMMs) and force measurements, is also well suited to discrete event recognition. An HMM is a powerful tool for describing how stochastic signals evolve in time. The force measurements are a rich source of information for contact tasks. Notice from Figure 3 that the frequency band is broad and occurs within a short time scale of the event.

An HMM is a doubly stochastic process. Each individual discrete event is represented by a hidden stochastic process which can not be observed directly but only through another observable stochastic process (see Figure 4). The HMM is able to estimate the behavior of the hidden process (discrete events) from the observable process. Motivated by Figure 3, the force signals are mapped to the frequency-domain by the FFT algorithm and used as the observable stochastic process.

The HMM approach to discrete event recognition is given by three steps as shown in Figure 5. The training of the HMMs is performed off-line. In real-time operation the probabilities of the measured forces matching the trained models are calculated. Finally, the discrete event with best match is chosen.

1. **Training**—Each individual discrete event τ_k is described by an HMM Λ_k. The model parameters of Λ_k have to be estimated from several training examples of the force signals. The parameters are estimated using the Baum-

Welch re-estimation formula (see [16] for the details). The formula is an iterative solution and at each iteration the probability of the HMM matching the force signals in the training set is increased. The training is performed only once and off-line.

2. **Evaluation**—When a discrete event occurs, every HMM corresponding to an admissible discrete event is evaluated. This is called *scoring* the HMMs. The score P_k is the probability of the model Λ_k matching frequency-domain force signals resulting from the event.
3. **Recognition**—The HMM with the largest score is chosen and the corresponding discrete event τ^* is sent to the discrete event controller, i.e.,

$$\tau^* = \tau_k \quad \text{where } k = \arg\max_k P_k \tag{8}$$

By comparing the individual HMM scores, a confidence level of the recognized event is calculated. The confidence levels are useful when incorporating a sensory perception controller, see our later discussion of Control of Sensory Perception. A confidence level of the final decision can be defined as follows.

$$c = \frac{P_{k^*}}{\sum_k P_k} \in [0, 1] \tag{9}$$

where P_k is the individual model score for HMM Λ_k and P_{k^*} is the largest HMM model score.

The stochastic model for the hidden process (for each individual discrete event) is shown in Figure 6. Several left-to-right models have been proposed for describing dynamic signals, see for example [29]. Note that there are several internal states q_i for each discrete event Λ_k. The states q_i describe the internal behavior of the discrete events. For example, if the force signals for event τ_k are static, then it is likely that the state of Λ_k will be q_1 most of the time, i.e., a_{11} is large. On the other hand, if the force signals are dynamic for event τ_k, it is likely that the hidden state will proceed from left to right in the model Λ_k, i.e., a_{ii} is small, $i=\{1, 2, \cdots, 7\}$. The estimation of the a_{ij} parameters is exactly how the force signals are used to estimate the behavior of the discrete events, as illustrated in Figure 4. The internal behavior of the discrete events is influenced by for example friction between the workpiece and the environment, sensor noise and dynamics of the manipulator.

Evaluation

Both process monitoring techniques presented above require training examples of the force measurements arising from the contact between the workpiece and the environment. Training the process monitors on empirical data ensures that phenomena such as sensor noise and friction between the workpiece and the environment are accounted for. The rule-based monitor recognizes a contact state transition in less than 1 ms of CPU time on a Motorola 68040-based processor. This is an advantage in real-time operation. However, the recognition rates are relatively low at 84%. In contrast, the HMM monitor has relatively better event recognition at about 97% but requires considerably more computational effort. Thus, the two monitors are complementary. By combining

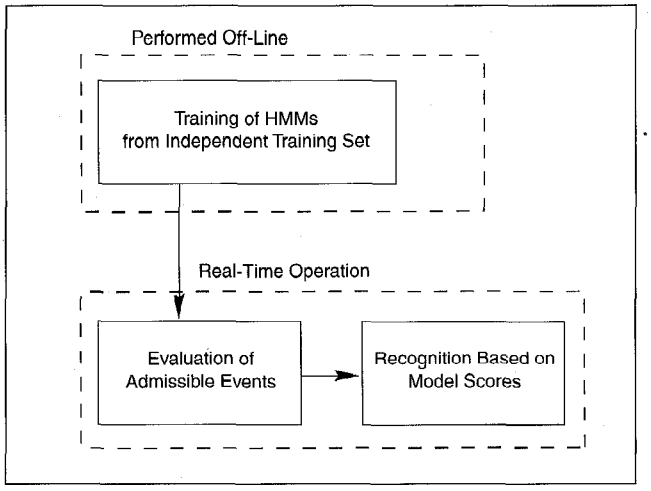


Figure 5. The HMM approach to discrete event recognition.

results from the two monitors, as we will show in the section on Control of Sensory Perception, we obtain better event recognition rates than either monitor while the average processing cost is kept low.

Before we show the performance details of the two process monitors, we will discuss the control syntheses procedures.

DISCRETE EVENT CONTROL SYNTHESIS

The control commands are determined by first establishing a *desired event* for each state. The desired event is selected to move to a state "closer" to the target state, that is, to move towards completion. The desired events may be determined manually or automatically, depending upon the application. For any given state, we use the desired event and geometric considerations of the workpiece and environment to establish conditions on the command to be executed.

Using the discrete event model described previously, we will develop a condition on the admissible velocity commands that will maintain a current contact, called the maintaining condition. Additionally, for nominal part locations, we will derive a necessary condition for a discrete event to occur, called the enabling condition, and we will derive a sufficient

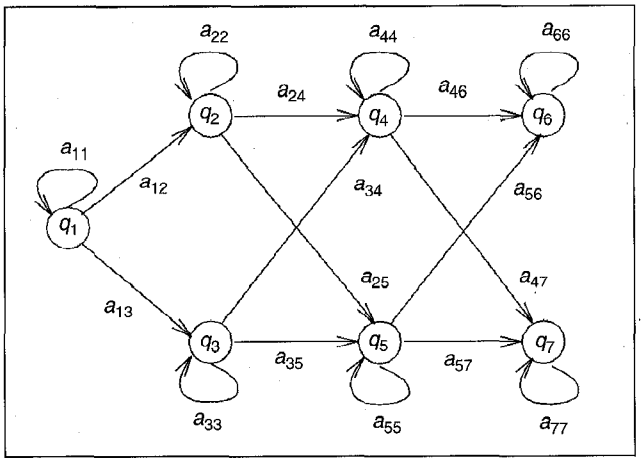


Figure 6. HMM Λ_k for each event. q_i are the states and a_{ij} are the hidden stochastic process state change probabilities.

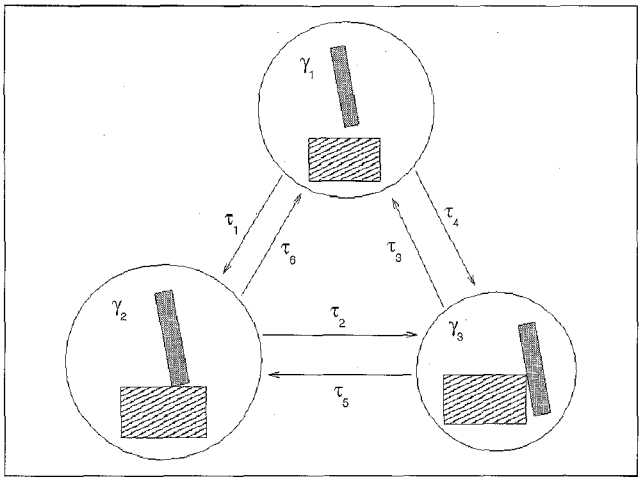


Figure 7. The task used to demonstrate the process monitoring techniques. The task used consisted of the following sequence of states shown in the figure: $\gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3 \rightarrow \gamma_1 \rightarrow \gamma_3 \rightarrow \gamma_2 \rightarrow \gamma_1$.

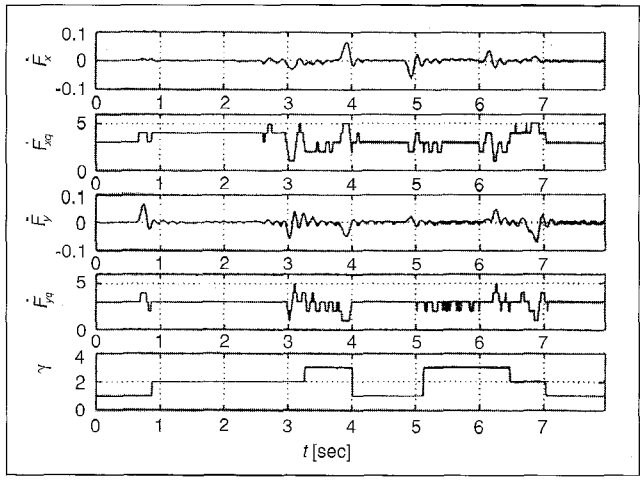


Figure 8. Measured force derivatives and recognized contact states γ. $\dot{F}_x$, $\dot{F}_y$ *are the measured planar force derivatives and* $\dot{F}_{xq}$, $\dot{F}_{yq}$ *are the corresponding discretised values.*

condition for a discrete event not to occur, called the disabling condition.

Maintaining Condition

The motion of the system described by Eq.(1) is constrained by Eq,(2). The first possible task of the controller is to ensure that the control commands satisfy this geometric constraint. To derive admissible velocities that satisfy the geometric constraint, we can differentiate Eq.(2) to give

$$\frac{\partial}{\partial \mathbf{x}}\left[g_j(\mathbf{x})\right]\frac{d}{dt}\mathbf{x}(t) = 0 \quad t_k \le t \le t_{k+1} \tag{10}$$

where g_j is the constraint function for this contact. This can be rewritten as

$$\mathbf{a}_j^T \dot{\mathbf{x}}(t) = 0 \quad t_k \le t \le t_{k+1} \tag{11}$$

where $\mathbf{a}_j = \frac{\partial}{\partial \mathbf{x}} g_j(\mathbf{x})$ is a column vector with length equal to the number of degrees of freedom. Eq.(11) is our maintaining condition in that it must be satisfied to maintain the contact or geometric constraint. When g_j is a distance measure, Eq.(11) becomes a requirement that the distance between the points of contact remains zero (i.e., the points remain in contact).

Enabling Condition

In addition to determining motion that maintains a constraint, it is desired to determine the motion such that the workpiece encounters the next discrete state γ_{k+1}. Since the system is not in γ_{k+1}, the following must be true:

$$g_j(\mathbf{x}(t)) = K \quad t_k \le t \le t_{k+1} \tag{12}$$

where, without loss of generality, K is a positive constant. In order to direct the system such that $K \rightarrow 0$, we require the time derivative to be negative.

$$\mathbf{a}_j^T \dot{\mathbf{x}}(t) < 0 \quad t_k \le t < t_{k+1} \tag{13}$$

Eq. (13) is our enabling condition. It is a necessary condition for discrete event τ_{k+1} to occur. When g_j is a distance measure, Eq.(13) becomes a requirement that the distance decreases, that is, the intended points of contact move closer together.

Disabling Condition

The third condition, the disabling condition, is derived directly from the enabling condition. Since Eq.(13) is a necessary condition for a discrete event to occur, a sufficient condition for a discrete event not to occur is obtained by changing the direction of the inequality.

$$\mathbf{a}_j^T \dot{\mathbf{x}}(t) \ge 0 \quad t_k \le t < t_{k+1} \tag{14}$$

where j indicates the discrete states (constraint equations) that are not desired to occur. Essentially, this disabling condition prevents K from decreasing in magnitude. When g_j is a distance measure, Eq.(14) becomes a requirement that the distance between the possible points of contact does not decrease (i.e., the points stay apart).

Table 1. The Performance of the Process Monitor

Performance (%)						
τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	All
After initial training						
100	48	75	68	93	100	79
After retraining						
100	72	61	83	93	95	84.5

Solving for the Control Command

The desired event determines which of the above conditions should be applied for each possible edge-edge or surface-vertex contact. The maintaining condition [Eq.(11)] is used when it is desired to maintain a contact. Note, when it is desired to immediately violate the current constraint by breaking the contact, the maintaining condition is not used. The enabling condition [Eq.(13)] is used to enable the loss or gain of a contact. The disabling condition [Eq.(14)] is used to prevent unwanted gains of contact. From the desired event, we now find a set of constraints on the control command, one for each possible edge-edge or surface-vertex contact. The control command is now determined by satisfying this set of constraints. Any method for satisfying the set of constraints will yield an acceptable discrete event velocity command. One method [4], which uses a search technique to maximize the minimum distance to each constraint for maximum robustness, is suggested.

EXPERIMENTS

The process monitoring methods and the discrete event control of assembly have been implemented and tested experimentally and the results are presented in the following.

Process Monitoring

Rule-Based Process Monitor

The rule-based process monitor was tested on a task consisting of several contact states resulting from contact between a rectangular peg and a solid block. These contact states are illustrated in Figure 7 and are denoted by γ_1, γ_2 and γ_3. Figure 7 also shows the contact state network for the experimental setup, and the allowed transitions $\tau_1 \cdots \tau_6$ between the contact states. The task used to test the process monitor consists of the sequence of contact states $\gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3 \rightarrow \gamma_1 \rightarrow \gamma_3 \rightarrow \gamma_2 \rightarrow \gamma_1$.

The task was demonstrated several times by a person and the data from three demonstrations was used to set up the rule-base of the process monitor described in the section on Rule-Based Monitors. The process monitor was used to monitor the task described above and Figure 8 shows the output of the two stages of the process monitor. The topmost graph in Figure 8 shows $\dot{F}_x$, the derivative of the force acting tangential to the top surface of the solid block. The second graph shows the discretised variable $\dot{F}_{xq}$ produced by the process monitor. Similarly, the third and fourth graphs relate to the force in the normal direction. The graph at the bottom of Figure 8 shows the discrete events recognized by the rule-based process monitor. For example, the graph indicates that the process monitor recognized the first transition τ_1 from contact state γ_1 to γ_2 at t=0.9s, approximately.

The performance of the contact monitor was measured by using it to monitor 25 demonstrations of the task described above. The percentage of transitions recognized correctly by the rule-based process monitor is summarized in Table 1. Table 1 shows the recognition accuracy for each individual transition for the task and also indicates the overall recognition accuracy. The first set of numbers in Table 1 shows the performance after initially training the rule-base using 3 demonstrations. However, note that the recognition accuracy of the process monitor for transition τ_1 from contact state γ_2 to γ_3 is very poor at 48%.

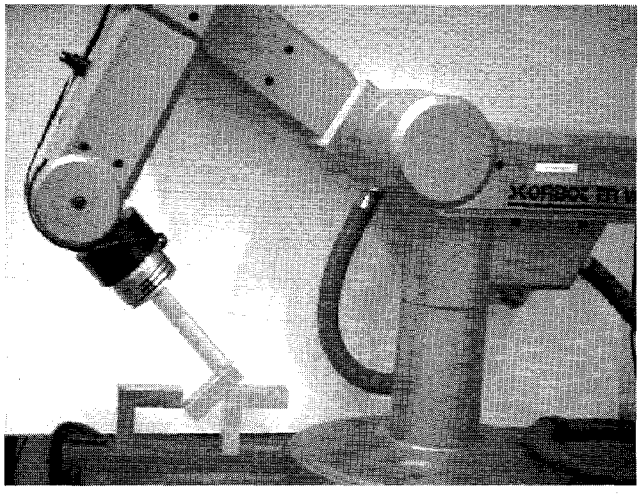


Figure 9. Eshed Scorbot 5-DOF manipulator and JR3 force/torque sensor used in the experiments.

Closer examination of the data revealed that for several demonstrations there were qualitative differences in the data corresponding to this transition. The contact monitor was therefore changed by replacing one of the demonstrations in the training set and then obtaining a new set of rules for the rule-base. The second set of numbers in Table 1 shows the recognition accuracy of the process monitor based on the new rule-base. The Table shows that the recognition accuracy of the process monitor for transition τ_2 improved to 72% and the overall recognition accuracy also improved to 84.5%. This demonstrates that the process monitoring system is very flexible and changes can easily be made to improve performance.

HMM Process Monitor

The HMM process monitor was implemented for discrete event contact recognition of an L-shaped assembly. The experimental setup consists of a 5-DOF Eshed Scorbot, and a 6-axis JR3 force/torque sensor as shown in Figure 9.

For this particular assembly task we define 11 contact states as shown in Figure 10. If all physically possible con-

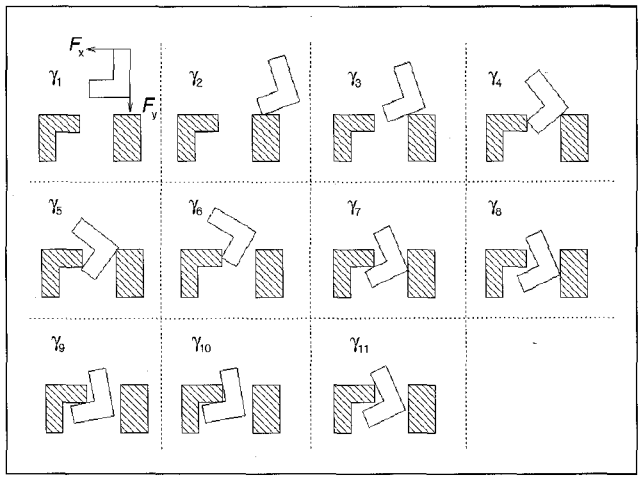


Figure 10. Model of contact states for a planar assembly task.

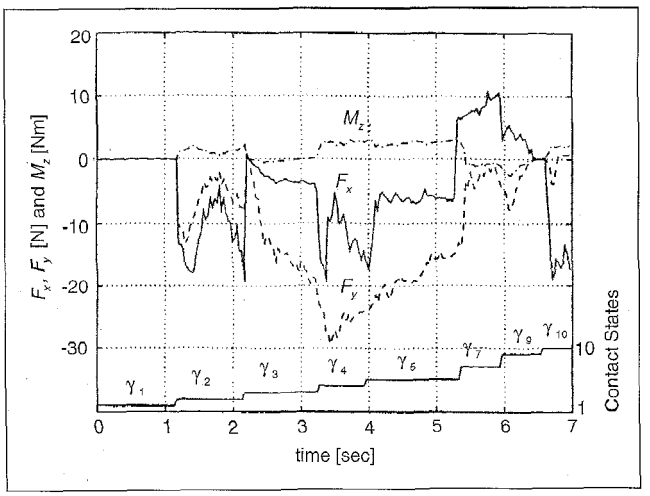


Figure 11. Force measurements for contact state sequence $\gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3 \rightarrow \gamma_4 \rightarrow \gamma_5 \rightarrow \gamma_6 \rightarrow \gamma_7 \rightarrow \gamma_8 \rightarrow \gamma_9 \rightarrow \gamma_{10}$.

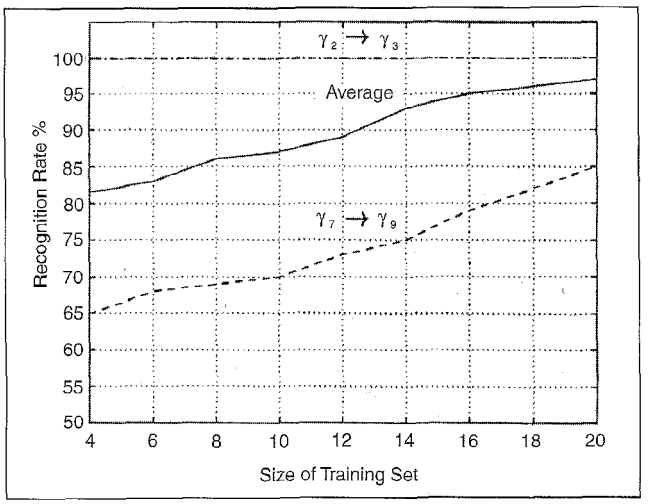


Figure 12. The performance rate vs. the training set size.

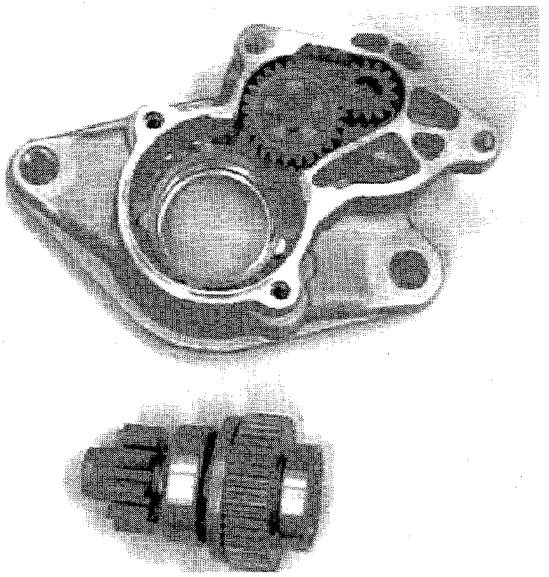


Figure 13. Gear Mechanism for a Starter Motor Assembly.

tact state transitions are considered, we get a total of 25 discrete events.

Figure 11 shows the measured force/torque signals for a complete assembly. The training of the process monitor involves collecting several force/torque measurements, F_x, F_y and M_z for each of the 25 discrete events. Figure 11 shows the force/torque measurements for a few of these discrete events. When a discrete event occurs in real-time operation, characterized by a sudden change in at least one of the forces, all the HMMs corresponding to admissible events are evaluated. The recognized discrete event is the one with the highest HMM score, see Eq.(8).

Figure 12 shows how the successful recognition rate is a function of the training set size. Some of the contact state transitions are easy to recognize, for example $\gamma_2 \rightarrow \gamma_3$. Even with a training set size of 4 examples, the HMM process monitor successfully recognizes this discrete event every time. One of the most difficult transitions to recognize is $\gamma_7 \rightarrow \gamma_9$. With a training set size as high as 20 examples, the recognition rate is only 85%. The average recognition rate, however, is as high as 97% with a training set size of 20 examples.

Control Experiments

In order to demonstrate the standard control operation, an industrial assembly was implemented. The industrial task was the assembly of a gear mechanism for a starter motor as shown in Figure 13. The minimum tolerance for the assembly is 0.075mm. One of the goals of the work is to reduce the costs of the grippers and fixturings through a simple control methodology that can accommodate and react to relatively large uncertainties. Thus, simple grippers were used with an uncertainty of 2.5mm. Due to the large uncertainty-to-tolerance ratio of 33, impedance control was unable to reliably execute the insertion task. However, a discrete event controller with force sensing was able to reliably execute the task.

A series of industrial experiments was run using a six degree-of-freedom robot built by Nippondenso. The robot was belt driven and was position controlled with force sensing. In addition to the uncertainty of the grippers, there was additional uncertainty due to the belts. The belt drives also generated significant noise on the force signal.

For comparison, an impedance controller was also implemented. The impedance controller proved to be successful approximately 50% of the time. The discrete event controller proved to be highly successful, completing insertion approximately 90% of the time. An example force signal and the corresponding event trajectory are shown in Figure 14 . The nominated trajectory is $\gamma_0 - \gamma_5$, where γ_0 is no contact and γ_5 is the full assembly. However, the system travels through many different event trajectories, such as γ_1–γ_2–γ_3–γ_4–γ_5, depending on the velocity commands and the uncertainties in the system. Nonetheless, due to the discrete event model and control structure, successful assembly is accomplished even for undesired event trajectories.

ADVANCED APPLICATIONS

In addition to standard operation, the hybrid dynamic framework favors advanced applications within constrained robotic manipulation. The advanced applications covered in this

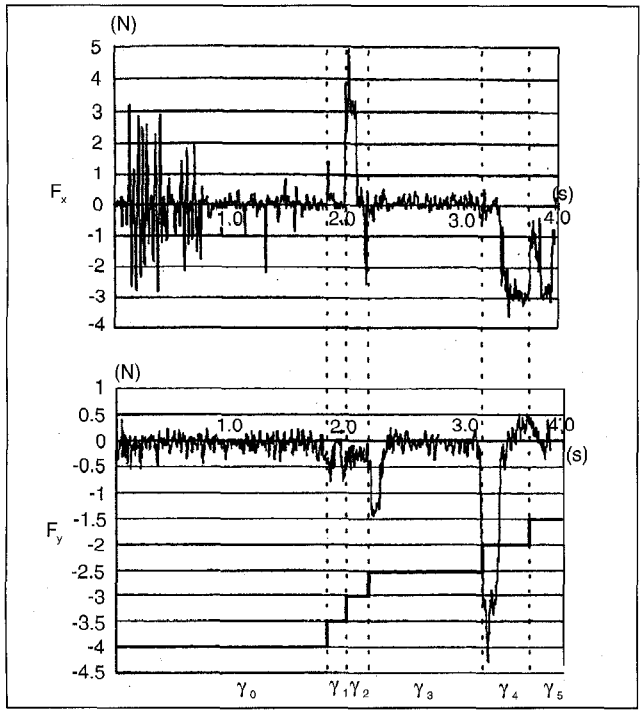


Figure 14. Force Profile and Event Trajectory for Gear Assembly.

paper include: (1) the on-line training of robotic assembly lines using task level adaptive control; (2) advanced sensory perception using stochastic dynamic programming; (3) the human sharing of control; and (4) the understanding of human motion and skill acquisition.

On-line Training of Assembly

Discrete event control offers considerable advantages for assembly tasks including excellent error-recovery characteristics. However, despite determining a velocity command which satisfies the constraint equations of the previous section on Discrete Event Control Synthesis, errors can still occur due to model inaccuracies, tracking control errors, or other unknowns. Unfortunately, the errors will result in a sub-optimal trajectory. In these situations, it is desired to have the system adjust to the new information and adapt the desired velocity commands. The ability to adapt is particularly important in an industrial setting where new products are frequently introduced and the assembly line needs to be "tuned" to the new tasks. The extra time, work and uncertainty involved a sub-optimal operation motivates the need for task-level adaptation of the discrete event controller.

An effective means of task-level adaptation is to adjust the model so that the event conditions more accurately reflect the actual system. This method requires the altering of the $\mathbf{a}_i$ vectors of Eqs. (11), (13) and (14) and then finding a velocity command which satisfies the necessary inequality constraints. Clearly the advantage of such a system is the accurate knowledge of the system which can be gained. The disadvantage is the additional computation required to repeatedly solve a set of inequalities to find a velocity command.

Before proceeding with the development of the adaptation laws, we will introduce some common terminology. The actual constraint will be denoted $\mathbf{a}$, whereas the current best estimate of the constraint at the b^{th} trial will be denoted $\hat{\mathbf{a}}_b$. We will call this the *trial constraint*. The estimate error is $\tilde{\mathbf{a}}_b = \hat{\mathbf{a}}_b - \mathbf{a}$. We also define

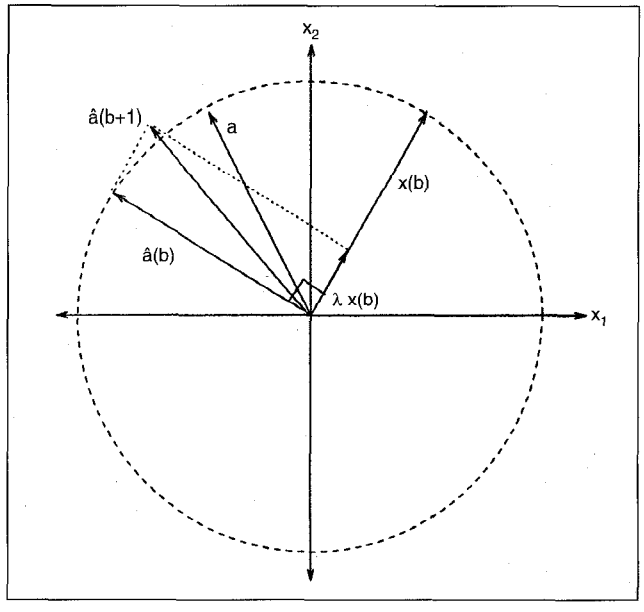


Figure 15. Justification of Direct Feedback Adaptation Law.

$$\Delta = -\mathrm{sgn}\left(\mathbf{a}^T \dot{\mathbf{x}}\right) \tag{15}$$

Intuitively, Δ is a switching function determined by the error condition to be adapted. If we erroneously lost contact then $\Delta = -1$, whereas $\Delta = 1$ if we recorded an over-force condition (a large increase in force). Also, if we gained an unwanted contact then $\Delta = 1$ and if we failed to gain a desired contact then $\Delta = -1$. Two different adaptation laws are now presented. The first offers simplicity and low computational complexity with conditional convergence whilst the second has guaranteed convergence but higher computational costs.

Direct Feedback Adaptation Law

An adaptation law needs to be selected such that the modeling error is reduced. We, propose an adaptation law based on intuitive reasoning about the vector spaces used for discrete event control as shown in Figure 15. Consider the adaptation of a maintaining condition. Here, the estimate of the constraint vector $\hat{\mathbf{a}}$ and the velocity vector $\dot{\mathbf{x}}$ are orthogonal. Yet, the velocity vector is not orthogonal to the actual constraint vector $\mathbf{a}$, indicating the need for adaptation. By adding a portion of the velocity vector to the estimated constraint vector, the difference between the estimated and the actual constraint vectors decreases. Hence, the following adaptation law is proposed:

$$\hat{\mathbf{a}}_{b+1} = \hat{\mathbf{a}}_b - \Delta\lambda\dot{\mathbf{x}}_b \tag{16}$$

where $\lambda > 0$ is the adaptation rate, and $\dot{\mathbf{x}}_b$ is the velocity com-

mand vector for the b^{th} trial. Given this adaptation law, two issues arise. The first is demonstrating the convergence of the estimated model constraint to the actual parameters. The second is the selection of λ such that the adaptation remains stable. Both of these issues can be answered using Lyapunov theory. For the complete proof of Lyapunov stability, the reader is referred to [24]. The result of that proof is that stability, and hence convergence to zero modeling error, is guaranteed if the following condition on the adaptation rate is met:

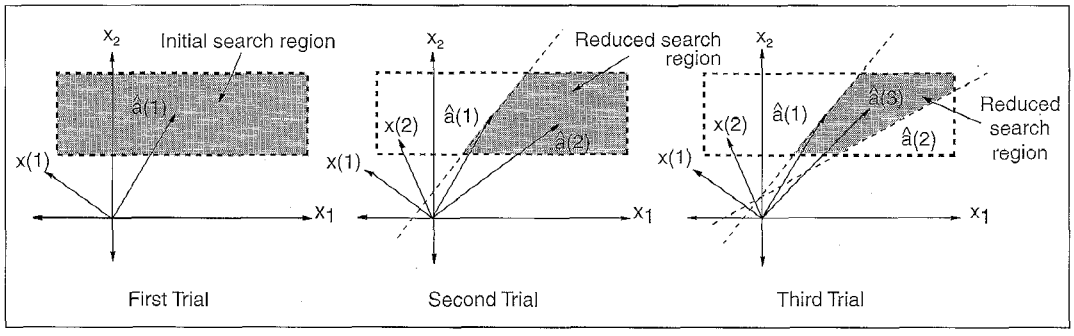


Figure 16. Justification of Bounding Adaptation Law.

$$\lambda < \frac{2\Delta\tilde{\mathbf{a}}_b^T\dot{\mathbf{x}}_b}{\|\dot{\mathbf{x}}_b\|^2} \tag{17}$$

Examination of how to satisfy Eq.(17) for each of the discrete event conditions yields the following requirements. For simplicity and to highlight the adaptation equations, we will assume that the velocity vector has been normalised to $\|\dot{\mathbf{x}}_b\|^2 = 1$. This assumption has little effect as only the direction of the velocity vector is important.

For the maintaining condition, Eq.(17) is satisfied provided

$$\lambda < -2\Delta\mathbf{a}^T\dot{\mathbf{x}}_b \tag{18}$$

and, for the enabling and disabling conditions, Eq.(17) is satisfied if

$$\lambda < 2\Delta\hat{\mathbf{a}}^T\dot{\mathbf{x}}_b. \tag{19}$$

Satisfying Eq.(18), however, is difficult because the quantity $\mathbf{a}^T\dot{\mathbf{x}}_b$ is not exactly known. Only the sign is known from the detection of discrete events. Furthermore, as the modeling error decreases, $\tilde{\mathbf{a}}_b$ decreases and so the adaptation rate must also decrease in order to satisfy Eq.(17). Specifics for the actual adaptation rate will depend on implementation, such as the rate of force increase or how quickly contact is lost, indicating the size of modeling error that exists. Nonetheless, Eq.(18) and the guideline of a small and decreasing adaptation rate will suffice for now.

To summarize, Eq.(16) gives an adaptation law for which the estimated model parameters converge to the actual model parameters provided Eq.(17) is satisfied. It has been shown that the condition for stability (Eq.(17)) is satisfied by the selection of the adaptation rate according to Eq.(18) or Eq.(19) for each case within discrete event control.

Bounding Adaptation Law

Motivated by the problems in determining a bound on λ for the Direct Feedback Adaptation Law, a second adaptation law is proposed. The Bounding Adaptation Law assumes that the actual constraint vector is known to be within some connected search region. This is a weak assumption as the search region can be made very large (even the entire set of possible constraint vectors). Successive experiments or *trials* are used to reduce the size of the search region, as shown in Figure 16.

More formally, the Bounding Adaptation Law assumes that the actual constraint vector $\mathbf{a}$ is bounded by a set of constraints, $\Psi(b)$. We define $\mathbf{R}(\Psi(b))$ as the set of constraint vectors that satisfy the constraints $\Psi(b)$. By the assumption above, $\mathbf{a} \in \mathbf{R}(\Psi(0))$. Once the bounding region has been established, we select a trial constraint, denoted $\hat{\mathbf{a}}_b$, somewhere within the region.

$$\hat{\mathbf{a}}_b \in \mathbf{R}\big(\Psi(b)\big). \tag{20}$$

Using the trial constraint, a velocity command is determined and this command is tried experimentally in the next execution of the controller. If an error occurs during the execution of the velocity command and the state monitor recognizes the unexpected event, we know which edge-edge or surface-vertex pair gained or lost contact or recorded an over-force condition. This information is contained in the definition of the event. Thus, we can determine which of the maintaining, enabling or disabling conditions failed. For each type of failure a new bound upon the actual constraint may be determined. The bounded region is reduced in size using this new bound, as shown in Figure 16. This process is repeated until the bounded region converges to a correct constraint value.

This adaptation procedure may be formally derived, for each type of error, resulting in

$$\mathbf{a}^T\dot{\mathbf{x}}_b > \hat{\mathbf{a}}_b^T\dot{\mathbf{x}}_b \tag{21}$$

or

$$\mathbf{a}^T\dot{\mathbf{x}}_b < \hat{\mathbf{a}}_b^T\dot{\mathbf{x}}_b \tag{22}$$

Eq.(21) or (22), depending upon the type of failure, establishes a new bound upon $\mathbf{a}$ which reduces the size of the set $\mathbf{R}(\Psi(b))$. Intuitively, the new bound states that $\mathbf{a}$ lies more (or

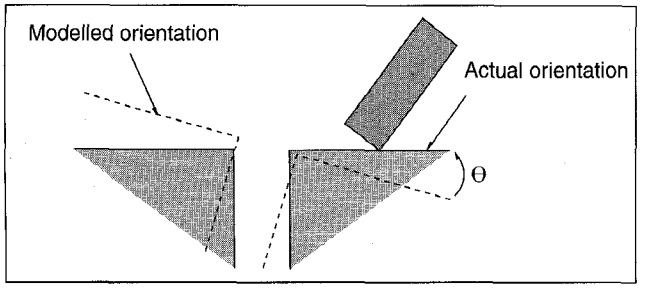


Figure 17. Two-Dimensional Assembly Task with Orientation Error.

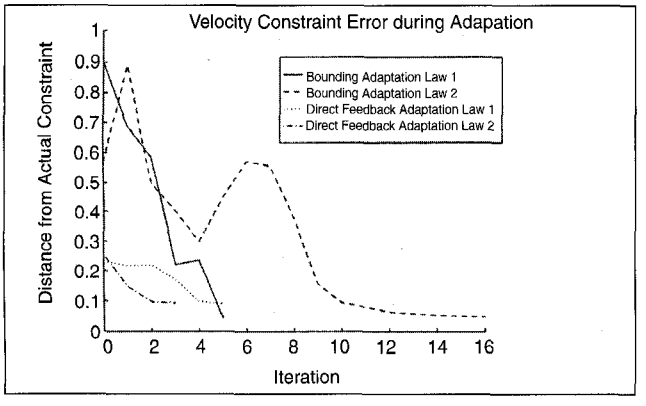


Figure 18. Experimental and Simulated Results for Adaptation Laws: Bounding Adaptation Law 1 demonstrates simulated results for a large initial search region; Bounding Adaptation Law 2 demonstrates simulated results for a small initial search region; Direct Feedback Adaptation Law 1 demonstrates experimental results for a low exponential decay rate; and Direct Feedback Adaptation Law 2 demonstrates experimental results for a high decay rate.

less) in the direction of $\dot{\mathbf{x}}_b$ than $\hat{\mathbf{a}}_b$.

Provided we select the trial constraint within an open subset of the search region the Bounding Adaptation Law converges to the correct constraint value. [27] presents the details of a formal proof of convergence, using Lyapunov stability theory. Note that the trial constraints may actually diverge from the actual constraints for some periods.

An important feature of the bounding law is that it uses only the knowledge that one of the conditions has failed (and the direction of failure when a maintaining condition fails) to reduce the size of a search region in which the actual constraint vector is known to lie. The major drawback of the Bounding Adaptation Law is that it requires more computation than the Direct Feedback Adaptation Law. Under the Bounding Adaptation Law the trial constraint $\hat{\mathbf{a}}_b$ must be found, such that $\hat{\mathbf{a}}_b \in \mathbf{R}(\Psi(b))$. This requires the solution of the set of constraints $\Psi(b)$. As in our discussion of Discrete Event Control Synthesis, we suggest a method [4], which uses a search technique to maximize the minimum distance to each constraint for maximum robustness.

Experiments

To demonstrate the effectiveness and convergence characteristics of the adaptation process, we will consider the motion control of an automated planar assembly task as depicted in Figure 17. The goal is to maintain contact between the corner of the workpiece and the horizontal surface, as shown in the figure. The estimated model parameters suggest that the surface is 10° off the horizontal as in Figure 17. This condition easily arises during actual operation due to alignment errors on the incoming fixtures (environments). The difficulty is due to the transport and support mechanisms for the fixtures, which are easily misaligned and often change during the course of a production run. As such, it is a good test case for adaptation.

For the Direct Feedback Adaptation Law, we need to determine values of λ to satisfy the convergence conditions. It has been shown that, in some cases, the bounds on λ are unknown. The experimental task demonstrated here is such a case and an empirical expression for λ has been developed using the basic assumption that the error decreases exponentially and so, λ should decay exponentially.

Convergence of the adaptation laws to the actual constraint vector is demonstrated for a number of cases in Figure 18. The Direct Feedback Adaptation Law has been implemented for a Puma 560 robot with considerable tracking errors (it was deliberately not calibrated). Figure 18 shows the performance of the Direct Feedback Adaptation Law with a high exponential decay rate, experimentally determined for fast convergence. Results are also presented with a smaller decay rate which leads to slower convergence. Also, *Direct Feedback Adaptation Law 2* contains a state detection error (occurring at iteration 2), a feature of any practical work. This practical example demonstrates that the Direct Feedback Adaptation Law is able to recover from errors elsewhere in the system.

For the Bounding Adaptation Law these results show that convergence to the correct constraint is achieved despite extremely poor initial information. Note that, in the periods when the trial constraint is diverging from the actual constraint the adaptation process is "searching" areas of the search region that are remote from the actual constraint. These regions are quickly eliminated and the trial constraints again converge to the correct constraint. A property of the Bounding Adaptation Law seems to be the periodic excursions away from the actual constraint (as occurred at iterations 1, 5 and 11). Simulated results for a smaller search region are also presented, highlighting the speed with which convergence may be achieved.

Evaluation

Two adaptation laws have been presented which successfully converge to the correct constraint vectors. These adaptation laws have been tested by simulation and experiments and from these experiments, it can be seen that each adaptation law has its good and bad points. The Direct Feedback Adaptation Law allows the adjustment of the rate of convergence and error recovery properties but the proper selection of λ is not guaranteed. The great advantage of the Bounding Adaptation Law is its guaranteed convergence, assuming no state detection errors. However, state detection errors can lead to failure to converge. In most practical situations, the Direct Feedback Adaptation Law is probably best suited as empirical expressions for λ lead to good results. The Bounding Adaptation Law may be more appropriate in critical applications (e.g., medical applications), when coupled with a reliable state detection scheme.

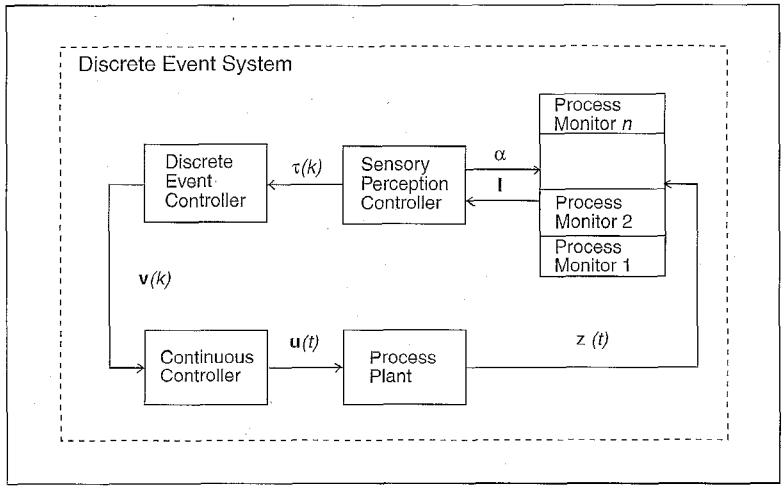


Figure 19. Discrete event system with dynamic sensory perception capabilities. v(k) is a vector of discrete controller commands occurring at time k, u*(t) and z(t) are continuous plant inputs and outputs,* I *is a vector of symbolic process plant information, a is a command (or action) from the sensory perception controller and τ(k) is the final output from the SPC.*

Control of Sensory Perception

Reliable sensing is essential for successful control of plants in uncertain environments, as was just demonstrated in the previous section. Traditionally, control systems receive measurements from a fixed sensing architecture where all the sensors are used all the time. Hence, the bandwidth of the overall control structure is limited by the slowest sensor. We present a new technique for the real-time control of sensory perception. Typically, only a few sensors are needed to verify nominal operation. When an anomaly develops, additional sensors are utilized. The benefits of the proposed method are an increased reliability compared to individual sensors while the bandwidth is kept high.

The control of sensory perception is well suited to the hybrid dynamic framework. The process monitors provide feedback to the discrete event controller only when discrete events occur. Hence, processing time is available between events for use by the sensory perception controller. A block diagram of a discrete event control system with sensory perception capabilities is shown in Figure 19.

The control structure in Figure 19 has been implemented for the discrete event control of a robotic assembly task, see Figure 9. Three process monitoring techniques are available to the sensory perception controller.

Process Monitor 1 uses measurements of the manipulator's joint angles and the forward kinematics to find the 3-DOF Cartesian positions P_x, P_y and α. The recognition rate of the monitor is relatively low while the main advantage of the method is a low computational effort.

Process Monitor 2 is based on force measurements and qualitative template matching, and is similar to the rule-based process monitor presented earlier. The monitor has a reasonably high recognition rate and the computational effort is low.

Process Monitor 3 is another force/torque based method. The planar forces F_x, F_y and M_z are logged for approximately 0.5 seconds. With this substantial amount of force information, process monitor 3 is very reliable. However, the monitor requires a significant amount of computational effort to analyze the information and hence is relatively slow. The method is described in more detail in the previous section on the HMM process monitor.

The method used for the dynamic sensory perception is based on stochastic dynamic programming and is described in detail in [15]. The method starts with an initial confidence level of zero and all monitors enabled. Then the sensory perception consists of two parts. First, an iterative dynamic programming (DP) algorithm evaluates all possible orderings of enabled process monitors by calculating the DP value function V. The dynamic programming model is formulated as an optimal stopping problem. At each iteration two actions are evaluated; a_1—terminate the sensory perception, or a_2—consult another process monitor. Second, a sensory perception controller (SPC) selects the ordering of enabled process monitors with the highest V. If the optimal action for this ordering is a_2, then the first monitor in the ordering is consulted. The confidence level output from the monitor is recorded. Next the monitor is disabled and the sensory perception problem is repeated with the new initial confidence level. The SPC terminates when the optimal action is a_1 or all monitors are disabled. The final recognized discrete event is sent to the discrete event controller.

The performance of the SPC was evaluated from a sample set of 100 discrete events and compared with the individual process monitoring techniques. The results are given in Table 2. The table shows the average successful recognition rates and the costs of the individual process monitors. With individual average recognition rates of 79%, 85% and 87% the average recognition rate of the sensor selection controller is as high as 97% which is better than any individual process mon-

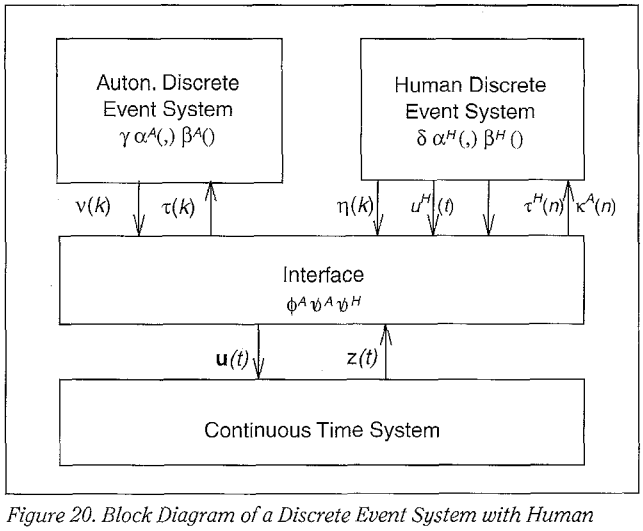


Figure 20. Block Diagram of a Discrete Event System with Human Integration.

Table 2. Evaluation of Different Process Monitoring Techniques and the Sensory Perception Controller

Monitor	Rate	CPU Time
Position	79%	0.10
Template Matching	85%	0.08
Hidden Markov Models	87%	0.87
SPC	97%	0.38

itor. The CPU time spent by the SPC depends on the number of monitors used for each event. The average CPU time for the sample set of 100 discrete events was found to be 0.38 seconds. These results clearly show the benefits of fusing several sensing techniques for the process monitoring of robotic assembly.

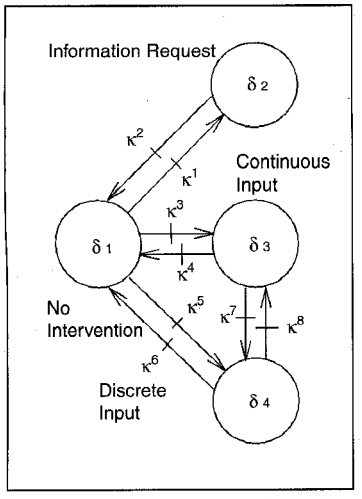


Figure 21. Human States and Events.

The main advantage of the sensory perception control structure is an improved event recognition rate compared to individual event monitors while the total cost is kept low. For cases where it is too expensive to use all the event monitors simultaneously, the maximum total cost can be limited and hence the proposed solution is well suited for use in real-time control systems with bandwidth requirements.

One advantage of the dynamic programming approach to the control of sensory perception is the ease with which new process monitoring techniques can be added to the system. The dynamic programming model remains unchanged, while more iterations are required by the DP algorithm to evaluate the new monitoring techniques. Interesting additional monitoring techniques include vision, energy and other force based methods, for example [11,34].

The real-time sensory perception controller uses lookup tables generated by the dynamic programming algorithm. The lookup tables are generated off-line. The SPC does not require any heavy computations and uses only the lookup tables generated by the DP algorithm. Hence, the proposed method is well suited to fast real-time operation.

Human Sharing of Control

The integration of a human supervisor into control systems is of benefit as the abilities of the human greatly enhance the control system. These benefits include decision making and fault recovery capabilities. Here a framework is presented which allows interactions of a human supervisor, modeled as a discrete event system, to be integrated into an autonomous process also modeled as a discrete event system. The discrete event formalism provides the means of modeling complex processes consisting of continuous and discrete components in an efficient and systematic manner. This is useful when modeling complex human interactions.

Framework

The integration of a discrete event model of human interactions with a discrete event model of an autonomous robotic system is considered. The combined structure consists of four separate subsystems, namely, the Human Discrete Event System (HDES), the Autonomous Discrete Event System (ADES), an interface and the continuous plant [1] as shown in Figure 20.

The continuous time system and the ADES have been described in our earlier sections on continuous time and discrete systems, respectively. The interface described earlier is modified to allow for the integration of the HDES.

Human Discrete Event Model

The discrete event formalism adopted to model human interactions is ideal as we can directly model the continuous and discrete levels of the human's interaction with the autonomous control system. Aspects of the human which are required to interact with a discrete event control system are various types of input actions. These input actions are modeled as states of an automaton and include *No Interaction*, in which the autonomous system performs the desired task without human interaction; *Continuous Interaction*, where the human can input continuous commands such as velocity and acceleration; *Discrete Interaction*, in which the human can force an event to occur or change the current state; and finally, *Information Request*, where the human can request more information about the system such as position data, force data, current state etc. The HDES automaton modeling the four human input actions is shown in Figure 21.

Each discrete state, δ, is defined as one of the four possible classes of human interventions. Each event $\kappa(n)$ is generated by a human input, $\mathcal{H}$, or a plant event such that

$$\kappa(n) = \chi^H(\mathcal{H}) + \kappa^A(n) \tag{23}$$

where χ^H is a function which maps human input to HDES events. Index n specifies the order of the discrete events in the HDES only. $\kappa^A(n)$ is a HDES event caused by plant events and is defined by

$$\kappa^A(n) = \psi^H(\mathbf{z}(t)) \tag{24}$$

where ψ^H is a mapping from the continuous state to discrete events of the HDES. Controller event $\eta(n)$ is generated by the HDES. The dynamics of the human discrete event controller are given by

$$\delta(n+1) = \alpha^H(\delta(n), \kappa(n)) \tag{25}$$

$$\eta(n) = \beta^H(\delta(n)) \tag{26}$$

where α^H is the state transition function and β^H is the output function. The outputs of the HDES are the continuous command $\mathbf{u}^H(t)$ (in this case velocity) due to human input and the discrete command issued by the human $\tau^H(n)$.

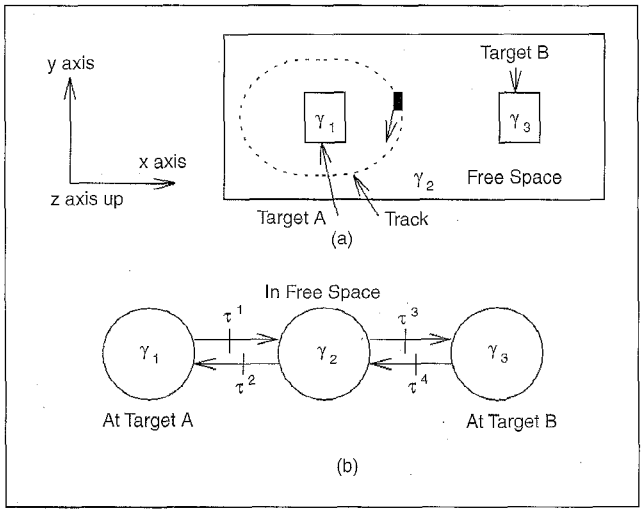


Figure 22. a. Physical Layout and ADES States; b. ADES Automaton.

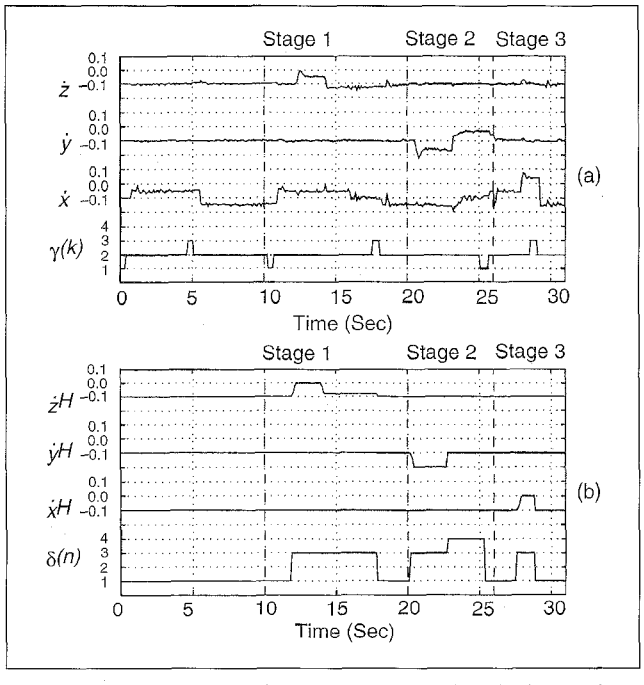


Figure 23. Experimental results: a. ADES states and total velocities; b. HDES states and human velocities.

The Interface of the Combined System

The interface must allow for communication between the HDES, ADES and the continuous system. Therefore the functionality of the interface is twofold. First, the interface must combine the continuous command from the human with the continuous command of the autonomous controller to generate a combined continuous command for the plant. Second, the interface must extract from plant events and human input, events for the ADES and HDES.

A combined continuous velocity command can be generated by adding the velocities from the ADES and the velocity input by the human via the HDES.

$$\mathbf{u}(t) = \mathbf{u}^A(t) + \mathbf{u}^H(t) \quad t(k) \le t \le t(k+1) \tag{27}$$

where $\mathbf{u}^A(t)$ is the velocity command from the autonomous system, $\mathbf{u}^H(t)$ is the velocity input by the human and $\mathbf{u}(t)$ is the combined continuous input vector to the plant.

Events, $\tau(k)$, for the ADES are derived via a map, ψ^A, from the plant state, $\mathbf{z}(t)$ [similar to Eq.(7)] and from discrete human input, $\tau^H(n)$, according to

$$\tau(k) = \psi^A(\mathbf{z}(t), \tau^H(n)). \tag{28}$$

Experiments Demonstrating Framework Operation

To demonstrate the operation of the above framework, a maneuvering task with interference from a moving obstacle was implemented. The autonomous task of the system is to move the robot between Target A and Target B. The obstacle, a train, moves around the track asynchronously compared to the robot. The trajectory of the robot takes it across the train track and therefore an eventual collision is inevitable. The responsibility of the human is twofold. The human first needs to recognize a potential collision and then take action in order to avoid the collision. The human was able to input continuous velocity commands and hence steer the robot to avoid the train.

The autonomous task, controlled by the ADES, is modeled by three states as shown in Figure 22(b). Figure 22(a) shows how the states of the ADES relate to the physical layout of the system. States γ_1, γ_2 and γ_3 correspond to areas in the workspace, Target A, Free Space and Target B, respectively.

The experimental data of a sample run is presented in Figure 23. Figure 23(a) shows the states of the ADES, $\gamma(k)$, and the measured velocities of the robot end-effector, $\dot{x}$, $\dot{y}$ and $\dot{z}$. Figure 23(b) shows the states of the HDES, $\delta(n)$, and the continuous input velocity of the human, $\dot{x}^H$, $\dot{y}^H$ and $\dot{z}^H$. It can be seen from the figure that in the time interval from 0 to 10 seconds the human did not interact, HDES state δ_1. In this interval the measured x-velocity shows that the robot was moving with a constant velocity until Target B was reached which caused a change of state to γ_3. Subsequently the $\dot{x}$-direction changed, and the robot again moved until Target A was reached. For further investigation, the figure is separated into several stages of the experiment. Stages 1, 2 and 3 will be examined further.

In **Stage 1**, the time interval from t=10 to t=20, a continuous input from the human was recorded. Figure 24 (Stage 1) shows the sequence of the experiment taking place. At t≈12 the human recognized that a collision would occur and moved the joystick upwards in order to allow the train to pass under the robot. The state change in the HDES from δ_1 to δ_3 generated event κ_3. The measured velocity $\dot{z}$ clearly shows the human input affecting the velocity of the robot. At t≈14 the human then pushed down on the joystick to reduce the speed of the robot in the upward z direction. This was necessary as $\mathbf{u}^A$ was not large enough to overcome the human input. This state continued until t≈18, at which time the robot reached Target B. This event also caused the HDES to return to the state of no intervention, event κ_4 (a plant forced HDES event). Note that at t≈16, the $\dot{x}$-velocity returned to zero. This is

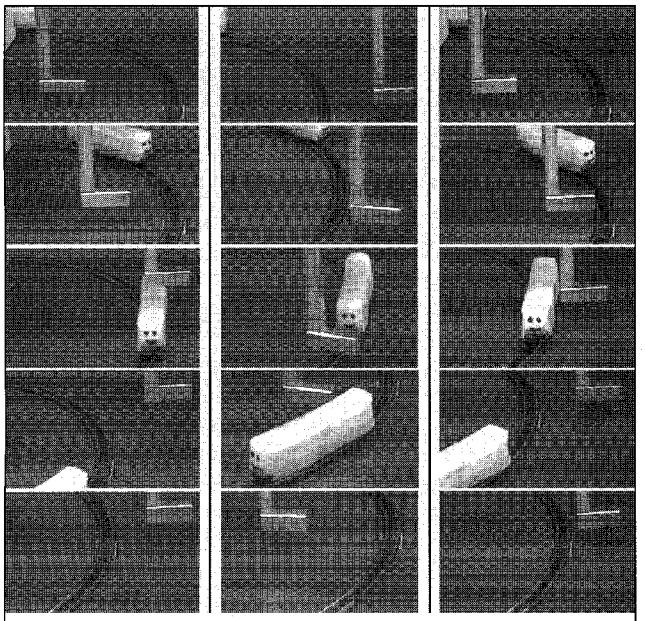


Figure 24. Experiment Stages 1, 2 and 3.

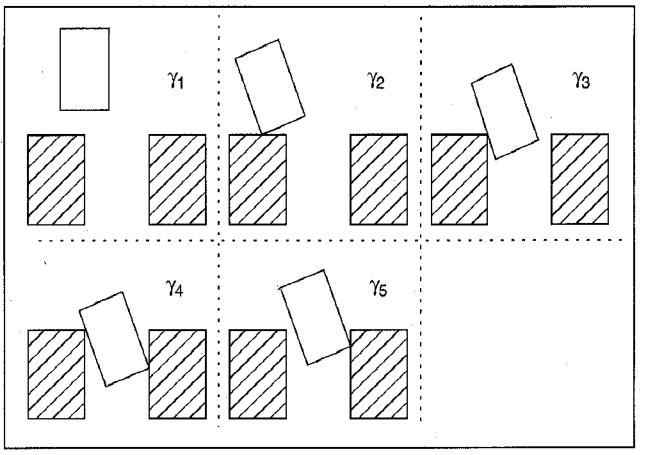


Figure 25. Planar peg-in-the-hole assembly with 5 contact states and 12 possible discrete events.

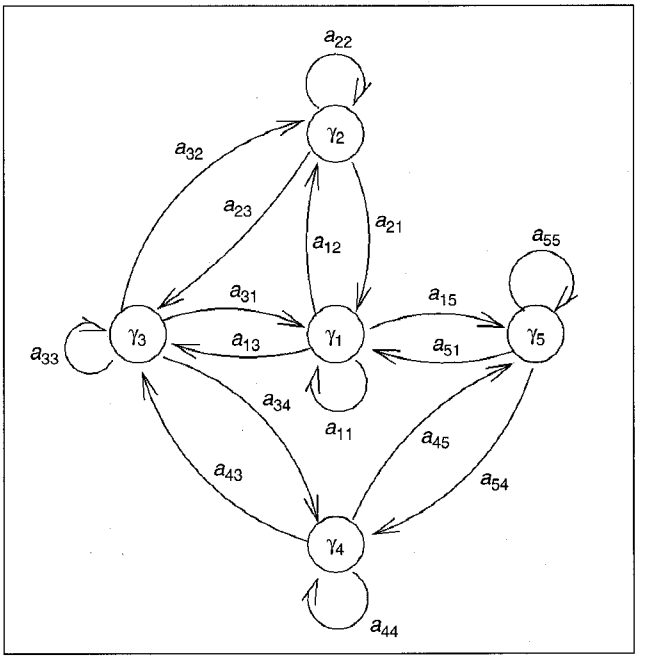


Figure 26. A stochastic automaton representing the discrete event controller for the peg-insertion task.

because the additional $\dot{x}$-velocity caused the robot to move above the target, hence no extra movement in the y-direction was necessary. A small downward z motion was recorded, bringing the robot into the target area. After spending a small amount of time at Target B, the robot then moves back into free space.

In **Stage 2**, the time interval from t=20 to t=26, a continuous input followed by a discrete command occurred. Figure 24 (Stage 2) again shows the sequence of the experiment. Shortly after t≈20, the human issued a command to move the robot in the negative y-direction. This interaction caused the robot to move away from the train and across the track in front of the train. The velocity change can be seen in Figure 23(b), $\dot{y}^H$. This event is recorded as κ_3, no intervention to continuous intervention, in the HDES. The velocity change is reflected by the change in $\dot{y}$. The human issued a discrete input at t≈23 which caused event κ_7. This event caused the continuous input to be returned to zero as can be seen in $\dot{y}^H$. After the human input returns to zero, a negative velocity was generated by the ADES to bring the robot back to the Target A, shown by $\dot{y}$. At t≈25, Target A was reached. This returned the HDES to the state of no intervention, δ_1 (event κ_6).

In **Stage 3** of the experiment, t=26 to t=31 seconds, a positive x-velocity was commanded by the human causing the robot to accelerate past the front of the train quickly. The sequence is shown in Figure 24 (Stage 3). In this stage the velocity command was canceled by an ADES event when the robot reached the target. The human velocity was returned to zero at this time. Note that because of the acceleration in the x-direction, it took less time to move from Target A to Target B.

The experimental results demonstrate the interactions between the four subsystems, the ADES, HDES, the interface and the continuous system. The results show that integrating a human into an otherwise autonomous discrete event control systems is successful using our framework. The framework allowed the human supervisor to interact and share control with the autonomous system. Using this type of human integration it is also possible for the human to assist the autonomous robot in case of errors such as environment modeling errors. Additionally, the framework allowed each subsystem to be designed and analyzed individually.

Human Skill Acquisition

Human operators are able to specify and perform constrained manipulation tasks easily. Therefore, an understanding of how people are able to learn and perform these tasks can be an important factor in the design of robot systems for such manipulation tasks. We present a new approach to task-level

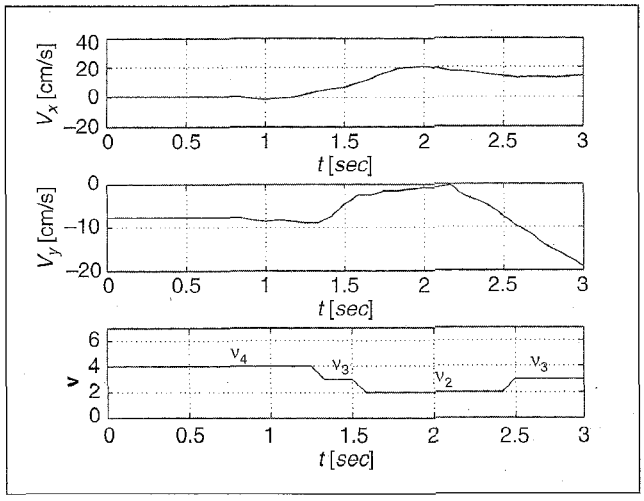


Figure 27. Typical velocity commands obtained from human demonstration data.

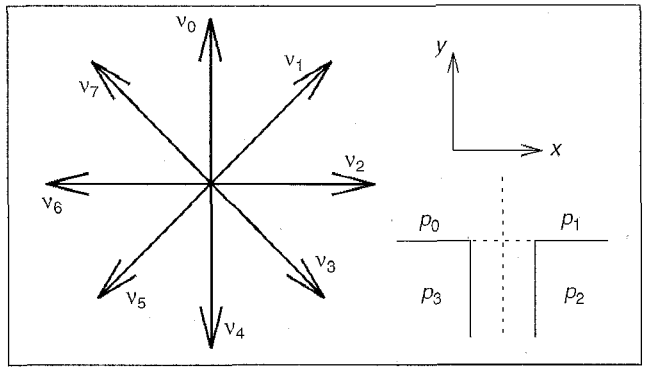


Figure 28. The discretised positions and velocities used to generate the discrete observation symbols of the HMM.

modeling of human manipulation skills using HMMs [18,29]. The states of the hidden Markov model correspond to the contact states of the task. The skill is acquired from human demonstration of the task, and the observation symbols of the HMM are obtained from the position and velocity of the manipulated object recorded during the demonstration. The HMM, once learned, is used to control a robot performing the task. The observation symbols of the HMM now correspond to the reference commands for the robot controller.

The approach is illustrated using a planar peg-in-hole task characterized by the 5 contact states $\gamma_1 \cdots \gamma_5$ shown in Figure 25. Figure 26 shows the stochastic automaton of the HMM based on the contact state network for the task. The states $\gamma_1 \cdots \gamma_5$ of this stochastic automaton correspond to the contact states $\gamma_1 \cdots \gamma_5$ in Figure 25. a_{ij} represents the probability of the transition between contact states γ_i and γ_j. For a given task, each contact state will tend to have a preferred transition that will lead to successful completion, and this preference for a given transition is encoded in the probability a_{ij} corresponding to that transition. The contact state network indicates that not all possible transitions are allowed and these transitions are assigned a probability of 0. Such transitions are not shown in Figure 26, for example, the transition between contact states γ_2 and γ_5.

The complete HMM model requires the observation symbols to be defined for each state of the stochastic automaton. In our model of human skill, these symbols are provided by the position and velocity of the manipulated object. For a given task, each preferred transition from a given contact state can be characterized by a preferred velocity in that state. This preference is encoded in the probability distribution of the observation symbols associated with each state of the automaton. The state transition probabilities of the stochastic automaton and the observation symbol probability distributions associated with each automaton state are obtained from data generated by human demonstration of the task.

Figure 27 shows the velocity of the peg recorded during a human demonstration of the peg-insertion task. The topmost graph shows the velocity in the horizontal direction, while the second graph shows the velocity of the peg tangential to the axis of the hole. We use discrete HMMs to model human skill and hence the observed velocity is discretised to obtain a sequence of observation symbols corresponding to each demonstration. The observed velocity is discretised into one of eight different observation symbols based on the direction of the velocity vector, as shown in Figure 28 . The third graph in Figure 27 shows the sequence of observation symbols v corresponding to the object velocity recorded during a demonstration.

The Baum-Welch re-estimation algorithm was used to obtain the HMM parameters from several demonstrations of the peg-in-hole task by a human. The following matrix **A** of transition probabilities a_{ij} was obtained:

$$\mathbf{A} = \begin{bmatrix} 0.90 & 0.04 & 0.04 & 0 & 0.02 \\ 0.04 & 0.94 & 0.02 & 0 & 0 \\ 0.09 & 0.03 & 0.85 & 0.03 & 0 \\ 0 & 0 & 0.03 & 0.95 & 0.02 \\ 0.09 & 0 & 0 & 0.05 & 0.86 \end{bmatrix}$$

As expected, the probability of transitions not allowed by the contact state network is 0, for example, a_{25}. The diagonal terms have the largest probabilities, indicating that most of the time is spent in a particular contact state and that the contact state transitions are fairly short-lived.

Table 3 shows the probability distributions for the 8 observation symbols for each state of the stochastic automaton. As expected, there is a preferred velocity for each automaton state. For example, in state γ_2, the most probable velocity is v_2 with a probability of 0.9, while v_4, with a probability of 0.4, is the most probable velocity in state γ_5.

The peg-in-hole task can be specified as a sequence of desired contact states starting with the initial state γ_1 and ending in the final state γ_4. For any such sequence, we can use the HMM parameters identified above to obtain the probability of the state sequence as well as the sequence of velocity commands associated with the desired contact state sequence. Table 4 shows the velocities and the probabilities of 3 such sequences.

From Table 4 we see that the most probable sequence from the initial state γ_1 to the final state γ_4 is $\gamma_1 \rightarrow \gamma_3 \rightarrow \gamma_4$. The corresponding sequence of discretised velocities is $v_4 \rightarrow v_4$. The velocities for the most likely task sequence are used by the

Table 3. Velocity Commands for Each Contact State Found from the Human Demonstration Data when Using the Baum-Welch Training Algorithm for the HMM

	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
γ_1	0	0	0	0.3	0.6	0.1	0	0
γ_2	0	0	0.9	0.1	0	0	0	0
γ_3	0	0	0.1	0.9	0	0	0	0
γ_4	0.1	0	0	0	0	0	0.3	0.6
γ_5	0	0	0	0.3	0.4	0.3	0	0

Table 4. Most Likely Contact State Sequence Found by Multiplying Transition Probabilities a_{ij}

Sequence	Velocities	HMM Prob.
$\gamma_1 \rightarrow \gamma_3 \rightarrow \gamma_4$	$v_4 \rightarrow v_3$	$1.2 \cdot 10^{-3}$
$\gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3 \rightarrow \gamma_4$	$v_4 \rightarrow v_2 \rightarrow v_3$	$2.4 \cdot 10^{-5}$
$\gamma_1 \rightarrow \gamma_5 \rightarrow \gamma_4$	$v_4 \rightarrow v_4$	$1.0 \cdot 10^{-3}$

discrete event controller as input commands to the continuous plant (see Figure 1). However, since the position of the hole is not precisely known, the desired event $\gamma_1 \rightarrow \gamma_3$ is very likely to cause the event $\gamma_1 \rightarrow \gamma_2$. When this situation occurs, the desired event trajectory is changed on-line by finding the new most likely trajectory from $\gamma_1 \rightarrow \gamma_4$. In this particular case, the new desired event trajectory will be $\gamma_1 \rightarrow \gamma_3 \rightarrow \gamma_4$. Thus, the HMM model effectively captures the human skill required to accomplish the peg-in-hole task.

A significant advantage of our approach to skill acquisition is that the acquired skill is represented in terms of physically meaningful concepts. The HMM parameters can be directly identified with the most likely sequence of contact states and the corresponding velocity commands required to accomplish a task. This compares favorably with other parametric models, such as neural networks, where it may be difficult to interpret the parameters in physically meaningful terms.

CONCLUSION

The research work and experiments in this paper have demonstrated the power of the hybrid dynamic framework for the monitoring and control of robotic manipulation. Additionally, we have demonstrated several advanced techniques based on this framework such as shared control, the control of sensory perception and human skill acquisition. Although not discussed here, this work has been successful in an industrial setting as well as the laboratory setting. In applying the hybrid framework we were able to accomplish complex tasks with relatively simple ideas based on discrete events. Our research work continues with emphasis on advanced control and sensing, and new applications.

REFERENCES

[1] Aigner, P. and McCarragher, B., Human Integration into Control Systems: Discrete Event Theory and Experiments, *Proceedings of the 2nd World Automation Congress*, May, 1996.

[2] Astuti, P. and McCarragher, B., Controller Synthesis of Manipulation Hybrid Dynamic Systems, *International Journal of Control*, to appear 1997.

[3] A. Barr and E.A. Feigenbaum, editors, *The Handbook of Artificial Intelligence: Vol I*, William Kaufman, Inc., 1981.

[4] Best, M.J. and Ritter, K., *Linear Programming: Active Set Analysis and Computer Programs*, Prentice-Hall, Inc., 1985.

[5] Biermann, A.W. and Feldman, J.A., On the Synthesis of Finite-State Machines from Samples of Their Behavior, *IEEE Transactions on Computers*, vol C-21, pp. 592-597, June 1972.

[6] Bobrow, D.G., editor, *Qualitative reasoning about physical systems*, Elsevier Science Publishers B. V. (North-Holland), 1984.

[7] Brockett, R., Hybrid Models for Motion Control Systems, in *Essays on Control: Perspectives in the Theory and its Applications*, H.L. Trentelman and J.C. Willems, eds., Birkhauser Publishers, Boston 1993.

[8] Bruyninckx, H., Dutre, S., and De Schutter, J., Peg-on-Hole: A Model Based Solution to Peg and Hole Alignment, *IEEE International Conference on Robotics and Automation*, June 1995.

[9] Pepyne, D.L. and Cassandras, C., Optimal Dispatching Control for Elevator Systems During Peak Traffic, *35th IEEE Conference on Decision and Control*, December 1996.

[10] Delson, N. and West, H., Robot Programming by Human Demonstration: The Use of Human Inconsistency in Improving 3D Robot Trajectories, *IEEE Conference on Intelligent Robots and Systems*, pp. 1248-1255, 1994.

[11] Dutre, S., Bruyninckx, H. and De Schutter, J., Contact Identification and Monitoring Based on Energy, *Proceedings of the 1996 International Conference on Robotics and Automation*, Minneapolis, Minnesota, pp. 1333-1338.

[12] Finotto, P., Crestani, D. and Prunet, F., Hierarchical Analysis for Manufacturing Systems Modeled with Petri Nets, *1996 International Workshop on Discrete Event Systems*, pp. 88-93, August 1996.

[13] Gollu, A. and Varaiya, P., Hybrid Dynamical Systems, *28th Conference on Decision and Control*, December 1989.

[14] Hirai, S. and Asada, H., A Model-Based Approach to the Recognition of Assembly Process States Using the Theory of Polyhedral Convex Cones, *Proc. of 3rd Japan-USA Symposium on Flexible Automation*, 1990.

[15] Hovland, G.E. and McCarragher, B.J., Control of Sensory Perception using Stochastic Dynamic Programming, *Proceedings of the First Australian Data Fusion Symposium*, Adelaide, 21-22 November 1996.

[16] Hovland, G.E. and McCarragher, B.J., Hidden Markov Models as a Process Monitor in Robotic Assembly, *The International Journal of Robotics Research*, to appear January 1998.

[17] Hovland, G.E., Sikka, P. and McCarragher, B.J., Skill Acquisition from Human Demonstration Using a Hidden Markov Model, *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, 22-28 April 1996, pp. 2706-2711.

[18] Huang, X.D., Ariki, Y. and Jack, M.A., *Hidden Markov Models for Speech Recognition*, Edinburgh University Press, 1990.

[19] Lemmon, M. and Bett, C., Hybrid Control System Design Using Robust Linear Control Agents, *Proc. 34th Conference on Decision and Control*, pp. 2688-2693, December 1995.

[20] McCarragher, B.J., Task-level adaptation using a discrete event controller for robotic assembly, *IEEE/RSJ Conference on Intelligent Robots and Systems*, pages 2281-2285, July 1993.

[21] McCarragher, B.J. and Asada, H., Qualitative template matching using dynamic process models for state transition recognition of robotic assembly, *The ASME Journal of Dynamic systems, Measurement and Control*, 115(2A):261-275, June 1993.

[22] McCarragher, B., Force sensing from human demonstration: stiffness, impedance and kinesthetic sensibility, *IEEE Conference on Intelligent Robots and Systems*, 1994.

[23] McCarragher, B.J., Force Sensing from Human Demonstration: Using a Hybrid Dynamic Model and Qualitative Reasoning, *IEEE International Conference on Robotics and Automation*, pp. 557-563, 1994.

[24] McCarragher, B.J., Model adaptive discrete event control, *IFAC Symposium on Robot Control*, pages 661-668, 1994.

[25] McCarragher, B.J., Task Primitives for the Discrete Event Modeling and Control of 6 DOF Assembly Tasks, *IEEE Transactions on Robotics and Automation*, vol. 12, no. 2, April 1996, pp. 280-289.

[26] McCarragher, B.J. and Asada, H., A discrete event controller using Petri nets applied to robotic assembly, *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, June 1992.

[27] McCarragher, B.J. and Austin, D.J., Model-adaptive hybrid dynamic control for constrained motion systems, submitted to *IEEE Transactions on Automatic Control: Special Issue on Hybrid Dynamic Systems*, to appear 1998.

[28] Niinomi, T., Krogh, B.H. and Cury, J.E.R., Synthesis of supervisory controllers for hybrid systems based on approximating automata, *Proc. 34th Conference on Decision and Control*, pp. 1461-1466, December 1995.

[29] Rabiner, L.R. and Juang, B.H., An Introduction to Hidden Markov Models, *IEEE ASSP Magazine*, January 1986, pp. 4-16.

[30] Sikka, P. and McCarragher, B.J., Monitoring contact using clustering and discriminant functions, *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, 22-28 April 1996, pp. 1351-1356.

[31] Sikka, P. and McCarragher, B.J., Rule-based contact monitoring and identification based on demonstration: An inductive approach, Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, Japan, August 1997.

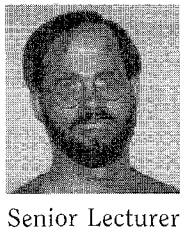
[32] Stiver, J., Antsaklis, P., and Lemmon, M.D., Hybrid control system design based on natural invariants, *Proc. 34th Conference on Decision and Control*, pp. 1455-1460, December 1995.

[33] Stiver, J. and Antsaklis, P., Modeling and Analysis of Hybrid Control Systems, *Proc. 31st Conference on Decision and Control*, December 1992.

[34] Trinkle, J.C. and Zeng, D.C., Prediction of the Quasistatic Planar Motion of a Contacted Rigid Body, *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, 1995, pp. 229-246.

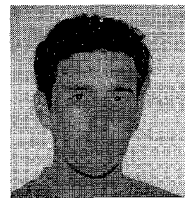
[35] Whitney, D.E., Quasi-Static Assembly of Compliantly Supported Rigid Parts, *ASME Journal of Dynamic Systems, Measurement and Control*, vol 104, pp. 65-77, 1982.

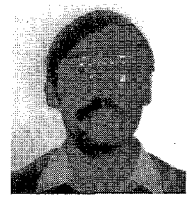
[36] Williams, B.C. and de Kleer, J., A theory of interactions: Unifying qualitative and quantitative algebraic reasoning, *Artificial Intelligence*, 51:1-79, 1991.



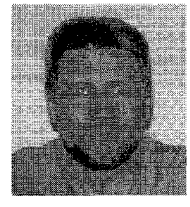
Brenan J. McCarragher received the B.S. and M.S. degrees in aeronautics and astronautics from the Massachusetts Institute of Technology, Cambridge, in 1988 and 1989, respectively. He received the Ph.D. degree, also from M.I.T., in mechanical engineering in 1992. He is currently a Senior Lecturer in the Department of Engineering at the Australian National University. He is Co-director of the Automated Systems Lab and Co-director of the Advanced Manufacturing and Production Systems Group. His research interests are in discrete event and hybrid dynamic systems (particularly applied to mining and manufacturing), robotics, and sensory perception. Dr. McCarragher received the Best Paper Award at the IEEE International Conference on Robotics and Automation in 1993.

Geir E. Hovland was born in Stavanger, Norway in 1970. He received the Siv.Ing (M.Sc. in Engineering) degree from the Department of Engineering Cybernetics at the Norwegian University of Science and Technology, Trondheim, in 1993. He is currently pursuing a Ph.D. degree sponsored by the Research Council of Norway at the Australian National University. His current research interests are Discrete Event Systems, Control of Sensory Perception and Process Monitoring of Robotic Manipulation.

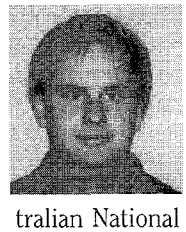




Pavan Sikka was born in India. He obtained his Ph.D. from the University of Alberta and is currently a Post Doctoral Fellow at the Australian National University. His research interests are in Robotics and Artificial Intelligence. In particular, he is interested in tactile sensing, touch-based control of robots, the control of human movement, and robot programming by human demonstration.



Peter H. Aigner was born in Munich, Germany in 1973. He received the B.E. degree in electronic engineering from the Royal Melbourne Institute of Technology, Australia, in 1994. In 1995 he commenced study toward his Ph.D. in engineering, which he is still pursuing, at the Australian National University. His major research interests include human integration into robotic control systems, discrete event systems and shared control.



David Austin was born in Canberra, Australia in 1973. He received a combined Bachelor of Engineering/Bachelor of Computer Science (Hons, University Medal) at the Australian National University in 1995. He is currently a Ph.D. student with the Department of Engineering at the Australian National University. His main area of interest is discrete event control and its applications. He is currently studying discrete event control of uncertain systems with emphasis on the development of control techniques including robust, adaptive and learning discrete event controllers.

Correction

We regret that an error occurred in the feature by R.G. Bonitz and T.C. Hsia, "Calibrating a Multi-manipulator Robotic System," *Robotics and Automation*, vol. 4, no. 1, March 1997, p. 20. Equation 6 should read:

$$e = \left\| \begin{bmatrix} {}^{o1}T_{o1}^{1} & \cdots & 0_4 \\ \vdots & \ddots & \vdots \\ 0_4 & \cdots & {}^{o1}T_{o1}^{n} \end{bmatrix} - I_{4n} \right\|$$