To be presented at the 4th International Symposium on Experimental Robotics, Stanford, CA, June 1995

An Object-Oriented Framework for Event-Driven Dextrous Manipulation

James M. Hyde, Marc R. Tremblay, and Mark R. Cutkosky
Center for Design Research
Stanford University
Stanford, California 94305-2232

Abstract
Multi-fingered robotic end-effectors have not yet made significant inroads into practical applications, partly due to the complexity of dextrous manipulation tasks. This paper develops an approach for assembling tasks from relatively simple phases which are punctuated by discrete events, signaling the transfer of operation to the next phase in a sequence. We examine the constraints active during phases, and develop methods for conducting smooth transitions between phases. Techniques for robust event detection in the presence of disturbances are also described. Experimental data is shown in support of the phase transition and event detection methods.

1. Introduction

A dextrous manipulation task can be viewed as a sequence of control phases punctuated by events. For example, as the fingers of a hand close upon an object they are driven using position control, but when manipulating an object they are driven to maintain control of internal forces. In this example, the sensation of contact is an event that signals the transition from one control phase to the next.

In manipulation, many events are associated with changing contact conditions and attendant changes in the kinematic and dynamic equations that describe the behavior of the fingers and object. The changing dynamics and kinematic constraints require different control laws and/or controller gains during each phase.

In the physiology literature, it has been observed that when humans grasp and manipulate objects they employ a sequence of responses triggered by events detected using a combination of haptic sensors [Joh91]. One motivation of the present work is to emulate the human ability to shift smoothly between responses.

The basic concept of a phase/event approach to dextrous manipulation and a discussion of the tactile sensors used to detect events have been given elsewhere [Cutkosky and Hyde 1993]. In this paper we focus on a framework developed for defining and executing phases and transitions and for detecting events.

In the present work, a task phase includes a specific control law, along with explicit force/motion constraints and provisions for force or motion trajectory specifications for the duration of the phase. The transition from one phase to the next is triggered by events which can be expected or unexpected. A phase-manager selects an appropriate subsequent phase based on the detected event. Using phase-based control, a complex manipulation task can be decomposed into comparatively simple elements, making the manipulation process tractable.

The concept of decomposing a task into discrete units is hardly new. However, there are some special considerations in the context of dextrous manipulation:

- Events are primarily associated with changes in contact status and are detected with (comparatively noisy) haptic sensors.
- Time constants are short; for example, if incipient slippage is detected, corrective action must be taken in milliseconds. Consequently the control framework must strike a balance between flexibility and computational efficiency.
- It is essential to maintain smooth control across the transitions associated with changes in the kinematic structure of the grasp. Rough transitions will disturb the sensors that are being relied on to detect events.

Our goal is a general purpose programming environment for dextrous manipulation which accounts for these considerations and permits flexible, robust designation and execution of manipulation tasks.

Our control framework is implemented as an object-oriented system with object classes for phases and events. We describe the framework and its motivating design decisions. We provide results of simple manipulation experiments conducted using the framework and discuss what they have taught us, motivating future extensions of the system.

2. Related Work

Although multifingered robotic hands have been available for over a decade, their use has been largely confined to a few research laboratories and their application in practical tasks under autonomous control is virtually nonexistent. One reason for the slow progress in putting such hands to work is the complexity of dextrous manipulation from the standpoints of kinematics, dynamics, sensing, control, and planning.

As Table 1 indicates, the requirements for successful manipulation can be broadly divided into three levels. At the highest level, one is concerned with task planning,

High-level (symbolic events, states)	task planning, grasp choice, discrete event systems, Petri nets, etc.	
Mid-level	phase, transition control	event detection
Low-level (co-op control, trajectory specification)	operational space dynamics, object impedance control, kinematics, forces, etc.	

Table 1: High-, mid- and low-level requirements for dextrous manipulation

grasp choice, etc. At this level, events such as acquiring or releasing a part are treated as symbolic entities that occur instantaneously and shift the hand/tool system from one state to the next. Considerable research has been conducted on discrete event systems for robotic manipulation and many of the techniques including fuzzy Petri Nets [Cao93] and discrete event systems [Sob92, Kat94, McC93] are applicable to dextrous manipulation.

At a lower level, dextrous hands pose formidable challenges in terms of coordinating the motions and forces applied by fingers grasping and manipulating an object. The literature in this area is extensive and includes important developments in kinematics, dynamics and control. Our own framework particularly builds on the operational space dynamics formulation of Khatib [Kha87], the representations of internal forces developed by Nagai and Yoshikawa [Nag93], the object impedance control formulation of Schneider [Sch89], and the MDL force/motion trajectory language of Brockett [Bro88].

While there is an extensive literature concerning the high-level and low-level issues in table I, the middle level, at which the framework presented in this paper resides, has received comparatively little attention. Some important exceptions include the work of Schneider [Sch89], Brock [Brk93], and Brockett [Bro94].

Schneider combined his object-impedance controller for cooperating arms with state tables of events and associated control actions. The finite state programming method employed branching and looping to improve task execution robustness.Brock introduced task segments described by a *context*, in which a particular *action* was executed in the pursuit of a *goal*. Brockett extended the theoretical framework of [Bro88] to address hybrid systems composed of continuous dynamics and discrete events.

Other relevant work includes techniques for smoothly and stably switching between motion and force control. Several techniques from the literature are reviewed by Hyde and Cutkosky [Hyd93] and compared with a method based on input command shaping.

The success of decomposing a task into phases also depends critically on the ability to detect events that will trigger the transition from one phase to the next. Eberman and Salisbury [Ebe94] used fingertip force/torque sensor information to label some simple events through a combination of signal processing and sequential hypothesis testing. In the present work we use a variety of dynamic tactile sensors which have been demonstrated in previous work [How90, Son94, Tre93] to be especially suited for registering contact events in dextrous manipulation.

3. Phase/Event Control Framework

Our control framework spans the gap between high and low level control noted in Table 1 by establishing a set of building blocks (phases and events) that are used to assemble manipulation tasks. Phases control the operation during a task segment in which a particular set of constraints is active, and events signal the shifts from one phase to the next. These building blocks communicate with a low level data manager to obtain sensor information and output commands to actuators.

Chains of Phases and Events

All phases store a list of events that might terminate that phase, along with the corresponding next phase. This structure of event/next-phase pairs allows rapid shifts between phases as events fire, and establishes a chain of phases to describe the task. The chains may contain branches and loops to promote robustness.

There are two types of phases: manager and action phases. Manager phases are responsible for activating the proper phase in a chain, and may govern multiple parallel sub-chains – useful when the fingers of a manipulator may be operating either independently or cooperatively. For example, three fingers might be cooperatively manipulating an object while a fourth finger executes an independent free motion to obtain a better grasping site. Chains may contain other manager phases, defining a hierarchical chain structure.

The action phases are responsible for actually controlling the fingers and/or grasped object. We have defined a class hierarchy of action phases corresponding to the different kinds of control laws needed:

- finger(s) independent versus cooperatively manipulating an object
- finger(s) or object in free motion versus in contact with, and constrained by, the environment
- finger(s) or object in stationary versus sliding contact

Phases are activated in response to events, which are implemented here as objects that store a confidence variable and methods for updating that variable. When an event confidence reaches a certain level, the event “fires” and the chain manager activates the next phase in the sequence. Note that with this interpretation events are instantaneous; the occurrence of an event means that the framework has *committed* to a particular event. This com-

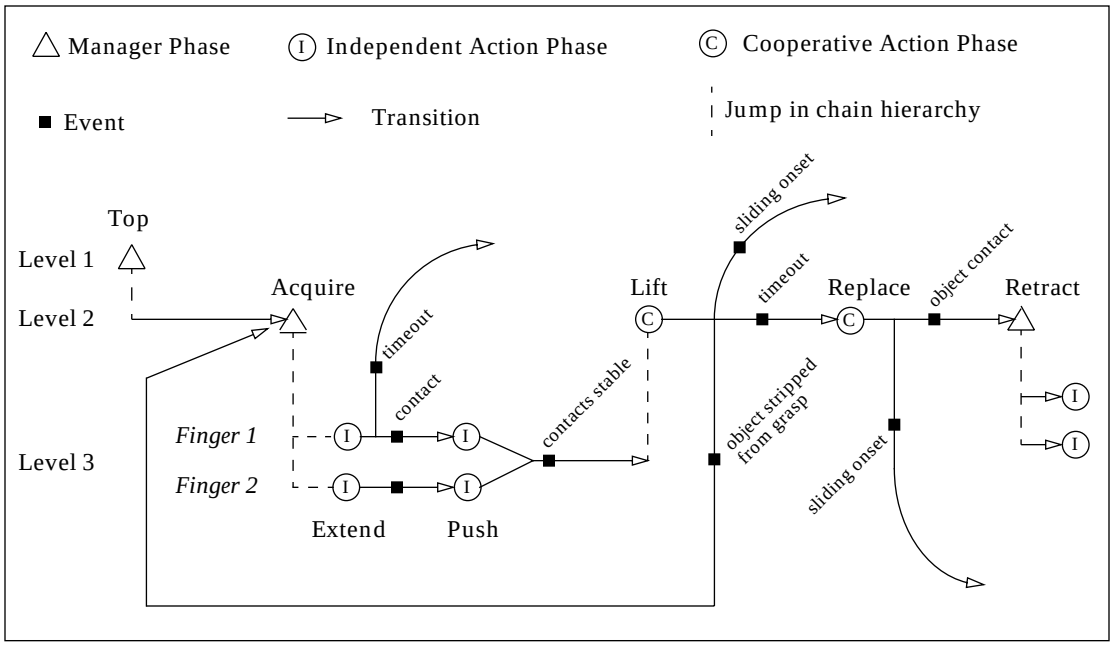


Figure 1: The phase/event structure for an Acquire, Lift, Replace task in which two fingers alternate between independent and cooperative phases in response to detected events. Arcs denote transitions into phase chains not shown here.

mitment logically takes the system from one state to the next.

Figure 1 shows events, manager and action phases for a simple grasp-lift-replace task involving a two-fingered hand. The entire task is governed by a single manager phase *Top*, which controls the task sequence: *Acquire*, *Lift*, *Replace*, *Retract*. We note that the *Acquire* phase is also a manager phase, controlling the execution of two parallel chains in which the left and right fingers independently approach the object, make contact and exert a prescribed contact force. If the fingers fail to make contact with the object within a certain time, a timeout event occurs, signalling the transition to a different chain designed to handle this problem.

Other branches are established by the multiple events that might terminate the *Lift* and *Replace* phases. If the object is stripped from the grasp during *Lift*, for instance, the detection of this event triggers a loop back to the start of the *Acquire* phase. Ultimately, a sequence of phases and sub-chains with branches and loops may become complicated. The literature on task planning and analyses of reachability, convergence, etc. will be applicable in these situations.

The notation in Figure 1 approximately matches the standard notation of discrete event systems. Action phases, denoted by circles, correspond to states (or "places" in Petri net notation); events are box labels that prompt the system to shift from one state to the next. Note that the term "transition" in the Petri net literature corresponds to what we would call an "event."

Transitions and Event Confidence

We define transitions as the beginning and ending sections of phases that are used to either prepare for an impending event or ramp up operation in the phase following an event. Figure 2 presents an example of a transition that would occur between the *Extend* and *Push* phases of Figure 1. The top plot in Figure 2 shows the evolving confidences of the events tracked in the pre-contact phase. When one or more of those confidences reaches a "preparatory threshold," an "alert action" is triggered, slowing the fingertip to a constant velocity as it approaches the object. More generally, alert actions are automatic adjustments in trajectory and/or control gains taken to facilitate transitions to subsequent phases. Because we do not yet know which event will actually occur, we can only take preparations within the context of the current phase and its control law and constraints. Eventually, the finger contacts the object, causing the corresponding event confidence to breach a "commitment threshold," triggering the switch to the contact phase. Once we have committed to a particular event and subsequent phase it is only logical to initiate control actions and event detection computations within the context of the new phase.

Returning to Figure 2, after the phase shift occurs, the "startup action" for the contact phase halts the controller

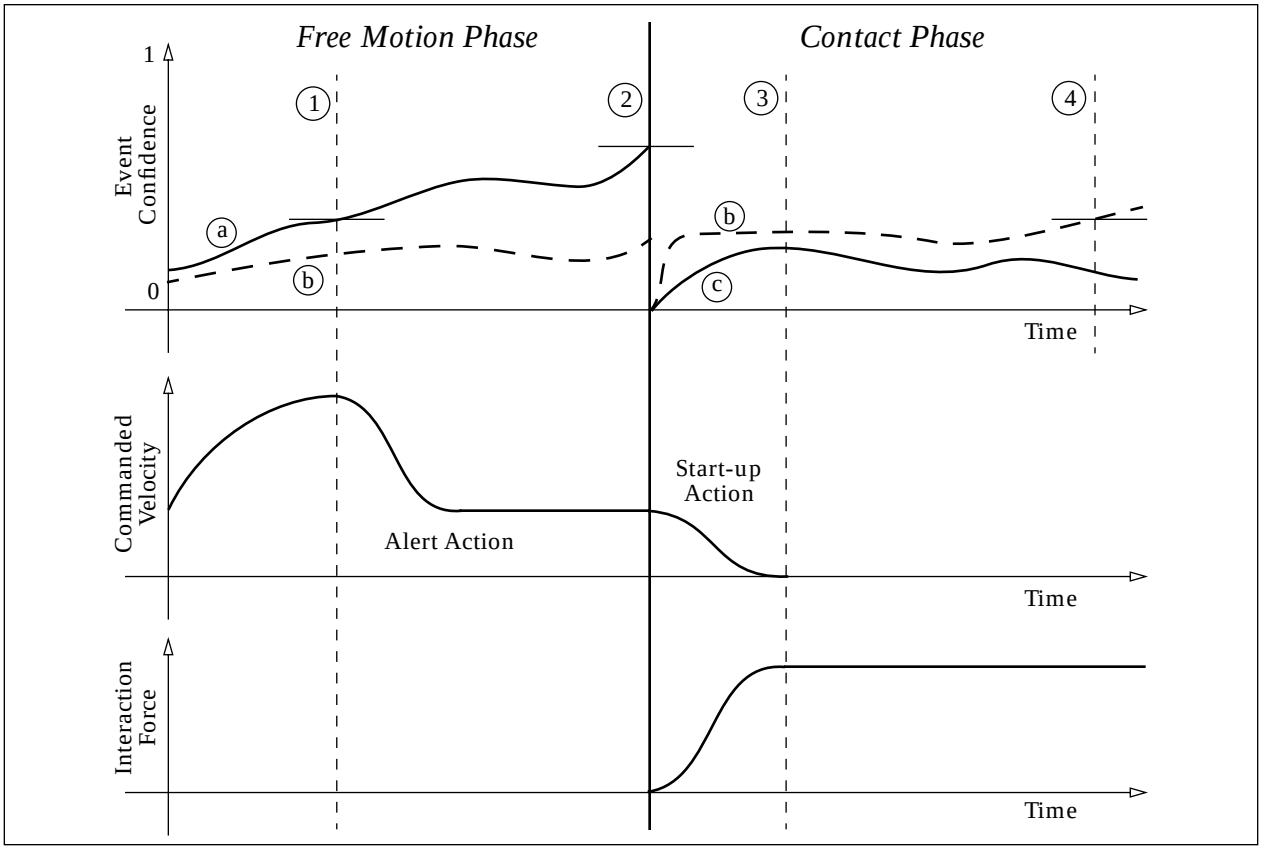


Figure 2: Event Confidences, Alert Action, and Startup Action for a transition prompted by a contact event. The Free Motion phase monitors the confidence levels of events a and b. At time 1, event a's confidence exceeds a preparatory threshold, prompting an alert action which slows the approach to a constant velocity. At time 2, event a crosses a commitment threshold, shifting execution to a Contact phase and resetting the event confidences. The new phase starts by halting the setpoint inside the object, causing the interaction force to increase. The startup action ceases at time 3. In the Contact phase, events b and c are monitored. Event b eventually reaches a preparatory threshold at time 4, causing another alert action (not shown).

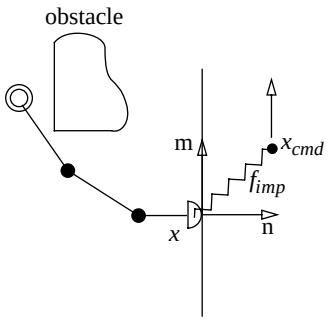
setpoint inside the object, causing an increase in the interaction force exerted between the fingertip and object. The event confidences are reset, and the set of scanned events in the new phase may differ from those in the previous phase. The new events are monitored and their confidences will evolve to trigger additional alert actions and phase shifts.

Action Phases & Constraints

The Free Motion and Contact phases noted in Figure 2 are examples of Action phases. Action phases, as mentioned above, are responsible for specifying the laws that control the behavior of a fingertip or grasped object. This responsibility includes specifying trajectories of setpoints for the phase control laws; following the approach of Brockett [Bro94] it is understood that trajectories may include both forces and positions with variable control gains.

Action phases also explicitly store the constraints acting on the system. Constraints may be natural, reflecting the holonomic or non-holonomic constraints imposed by environment surfaces, or user-specified, denoting artificial constraints created to ensure the proper execution of a phase. For instance, to avoid actuator saturation leading to non-linear behavior, constraints can be imposed on commanded setpoint accelerations or exerted forces. Natural and user-specified constraints are stored explicitly within each phase, rather than embedded implicitly into phase control laws. In this way a modest repertoire of basic phase types can be customized to meet task-specific constraints regarding contact kinematics, friction, etc.

Figure 3 provides a breakdown of the constraints active during a fingertip sliding phase. In the figure, a fingertip slides along a constraining surface under impedance control. The gap between the setpoint and actual fingertip position establishes an impedance force, f_{imp},which is sub-



Natural Constraints		User-Specified Constraints	
Environment	Robot	Hard	Soft
$x_n = const$	$\lvert\tau_i\rvert \le \tau_{i,max}$	$x_{cmd} \le x_{max}$	$\lvert f_{imp}\rvert \le f_{imp,soft}$
$f_m = \mu f_n$	$x \le x_{max}$	$\dot{x}_{cmd} \le \dot{x}_{max}$	
		$\ddot{x}_{cmd} \le \ddot{x}_{max}$	
		$\lvert f_n\rvert \le f_{n,max}$	
		$\lvert f_{imp}\rvert \le f_{imp,hard}$	

Violations cause an event

Figure 3: Constraint categorization for a Sliding phase using impedance control. The controller setpoint is moving in the "m" direction. If the links contact the obstacle, further setpoint motion may cause a violation of the impedance force soft or hard constraints. A soft constraint violation prompts the phase to take action to back away from the constraint to avoid actuator saturation. Violation of the hard constraints or natural constraints triggers a shift to a different phase.

ject to user-specified soft and hard constraints. In addition, the contact prevents further motion into the surface and defines a constraint on the normal and tangential interaction force components. These are natural constraints.

If the fingertip slides far enough, the manipulator links will encounter a workspace obstacle. Further setpoint motion will cause the impedance force to violate its soft constraint, prompting reactive behavior from the phase to avoid actuator saturation. If the impedance force exceeds its hard constraint, or if any of the other constraints in the first three columns of Figure 3 are violated, events fire, triggering a shift to a subsequent phase.

4. Event Detection

Events have been described thus far as objects used to signal the shifts between phases. The reliable detection of events is a rich problem in itself. Events have built-in functions for updating their confidences on the basis of a combination of sensor information and context.

In our framework, the individual fingertips are equipped with multiple sensors, each of which can provide several types of information. For example, tip position, velocity and acceleration can be computed from joint sensor data and short-time energy can be obtained from skin acceleration sensor data. We refer to each of these types of information as *sensor-based* features. When contextual information about the robot's behavior is included as a feature (desired velocity or force, for example) we refer to these features as *context-based* features.

For each phase in a manipulation task, if one identifies the possible events and the features required to detect them, one can construct a feature space for the phase. Let us define a feature F as a set of discrete real numbers f corresponding to all possible values for that feature and let Φ be the current phase of a manipulation task. Now let us define an n-dimensional Euclidean feature space corresponding to the cartesian product of the family of sets, or observed features, denoted by:

$$\boldsymbol{F}_{\Phi}^{n} = F_1 \times F_2 \times \ldots \times F_n$$

At any given moment during the phase, there will be an n-tuple $(f_1,f_2,...,f_n)$ which corresponds to the current feature values.

Within this feature space, each event $e_{i\Phi}^{q_i}$ will occupy a set of regions ξ corresponding to a q_i - dimensional subspace:

$$e_{i(\Phi)}^{q_i} = \{ \bigcup_{j=1}^{k} \xi_{ij} \,\Big|\, \xi_{ij} \subseteq e_{i(\Phi)}^{q_i} \text{ for all } i \in \aleph_p \}$$

where k corresponds to the number of regions ξ associated with each event e. If we define an n-tuple in the n-dimensional feature space of a given phase as

$$\boldsymbol{f}_{\Phi}^{n} = (f_1, f_2, \ldots, f_n)$$

It then follows that the condition

$$\boldsymbol{f}_{\Phi}^{n} \in E_{\Phi}^{p}$$

must be satisfied for an event to have *possibly* occurred.

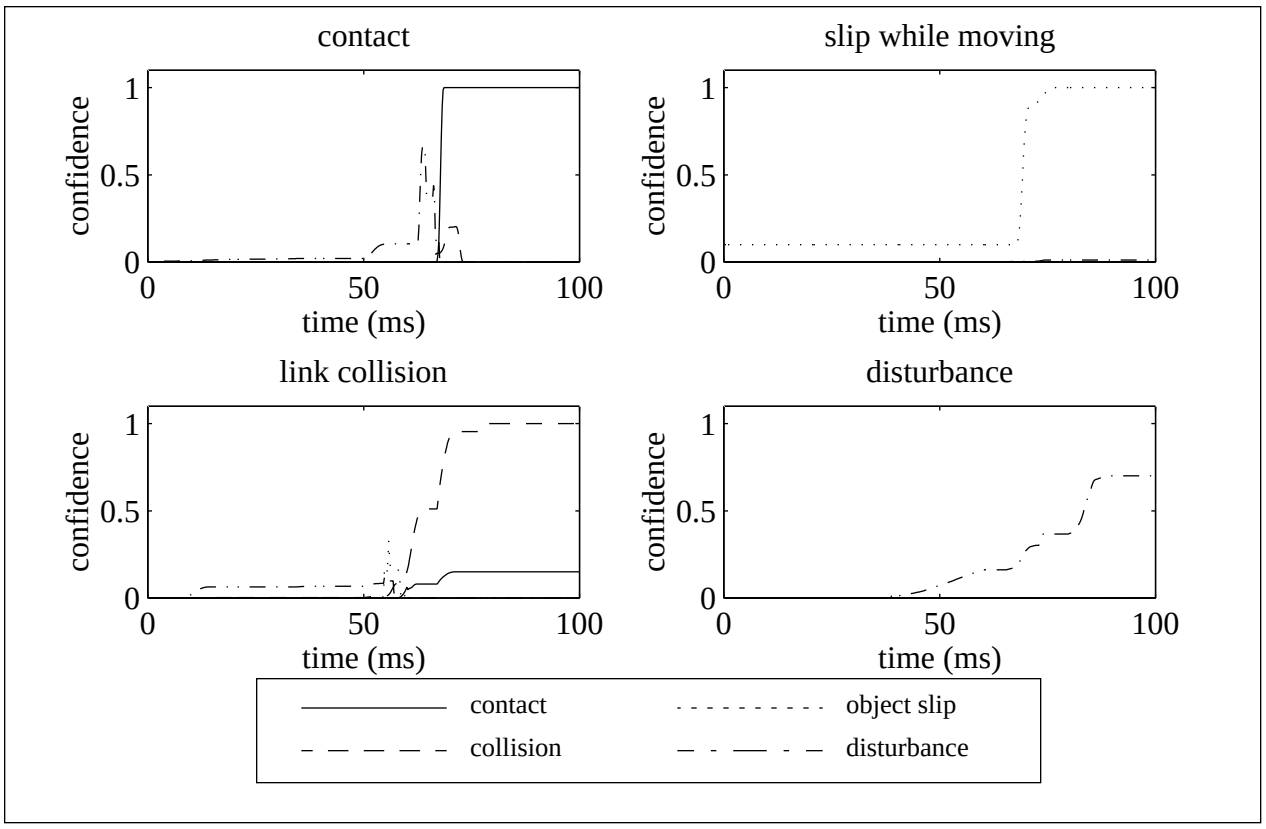


Figure 4: Data from four event detection experiments.

The decision as to whether an event has occurred is made using *confidence distribution functions* (c.d.f.'s), or $\psi(f)$ defined as:

$$\psi(f) : F \rightarrow [0, 1]$$

where 0 means that from that feature's point of view, the event could not possibly have occurred and a value of 1 means that from that feature's point of view, all requirements have been met and the event certainly *could* have occurred. The functions are similar to the membership functions in fuzzy set theory.

Once confidence values have been obtained for each feature, they need to be combined in order to get overall confidence values for each event. By observing these continuously varying confidences, the framework can make decisions relating to event occurrence. Whereas the sensor-based features are typically uncertain and noisy, the context-based features represent current knowledge and serve mainly to rule out certain events. Therefore, the overall confidence for an event consists of the weighted sum of the confidences of the sensor-based features multiplied by confidence values for each of the context based features. Thus a single context-based feature can significantly influence the overall confidence value.

Let the overall confidence Ψ that an event ε is occurring at sampling period *k* be defined as

$${}^{\varepsilon}\Psi_k = \left(\prod_{i=1}^{m} {}^{\varepsilon}_{i}\psi(\tilde{f}_i) \right) \times \left(\sum_{j=1}^{n} \omega_j \; {}^{\varepsilon}_{j}\psi(\hat{f}_j) \right)$$

where *m* corresponds to the number of context-based features, *n* corresponds to the number of sensor-based features, ω_j is the weight assigned to a sensor-based confidence values and ψ is the c.d.f. of a given feature. Note that the same feature will have different c.d.f.'s depending on which event is being considered. As shown in Figure 2, the relationship of the event confidences to established preparatory and commitment thresholds defines when the framework commits to an event and shifts operation from one phase to the next.

5. Experiments and Results

We conducted a series of tests of the event detection and phase transition schemes described in the preceding sections. For our experiments, we chose a "cooperative

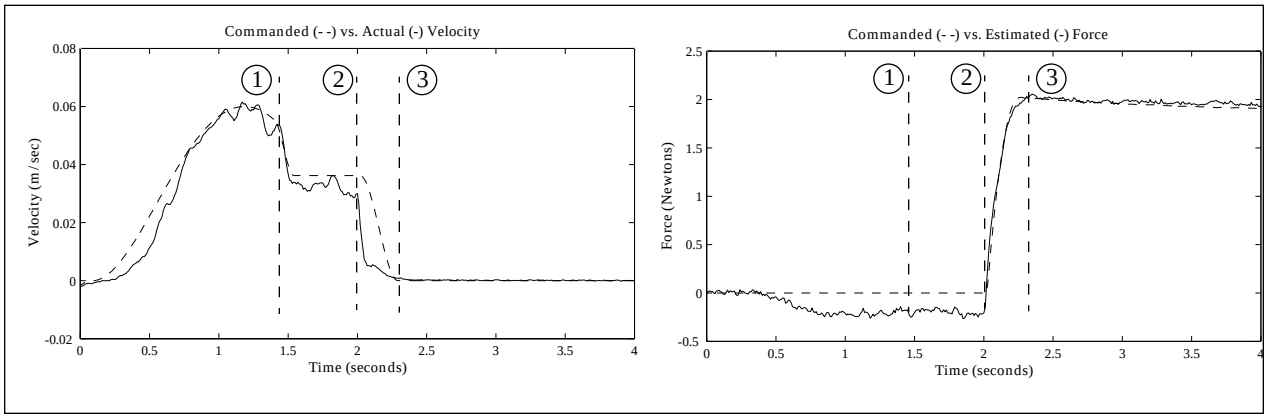


Figure 5: Data from a free motion -> contact transition experiment where a grasped object is brought into contact with an environment surface. Compare with Figure 2. At time 1, an alert action is triggered, causing the object to adopt a constant velocity approach. The object/world contact event is detected at time 2, and the startup action for the contact phase ceases at time 3.

motion" phase in which two fingers manipulate a grasped object. In this phase, we consider four types of events that cause sensor excitation thereby triggering reactions. These events are: object/world contact, link collision, mechanical disturbance and object slip. Four sensors were used for the experiments: position, force, skin acceleration and stress-rate sensors. The sensor-based features included maximum short-time energies of force and skin acceleration sensors as well as position and force errors. The context-based features included commanded object acceleration. Details on the sensors and features are provided in [Tre95].

The overall confidence value for each monitored event was computed for a variety of runs conducted over a range of speeds from 1 to 10 cm/s and with hard and soft objects. In all test cases, the scheme quickly identified the correct event. Figure 4shows the results for 4 typical runs where the overall confidence for each event is displayed as a function of time and the title of the plot indicates the actual event that occurred. For all plots, the event occurs at t=50ms and one can see that in every case the correct event is detected quickly.

Figure 5 shows the results for a typical phase transition. In this case the event is an object/world contact, as in the upper left plot of Figure 4. At approximately 1.4 seconds, the contact event passed the preparatory threshold, prompting a constant velocity approach. At 2 seconds the object/world contact event occurred, shifting execution to a cooperative manipulation Contact phase. The object/world interaction force increases during the start-up action to its desired level of 2 N. The transition is executed quickly without imparting significant vibration into the system. Details on this and other transitions are given in [Hyd95].

6. Conclusion

Our preliminary experiments conducted on a system of two fingers have demonstrated the efficacy of this architecture. The underlying object orientation of the phases and chains permits simple extensibility and rearrangement of the chains. The provision of explicit constraints, preparatory and startup actions, and possible terminating events for each phase allows a modest set of basic phase types to be applied to many specific tasks. The data manager, phase, transition, and event objects combine into a general programming environment for dextrous manipulation tasks. Using this environment, we can design and execute manipulation tasks with flexibility and robustness.

7. Acknowledgments

This paper describes work performed at the Stanford University Dextrous Manipulation Laboratory. Funding for this research was provided in part by the Office of Naval Research under the University Research Initiative contract #N-00014-90-J-4014-P01.

8. References

[Brk93] Brock, D. L., "A sensor based strategy for automatic robotic grasping," Massachusetts Institute of Technology Ph.D. Thesis, Department of Mechanical Engineering, 1993.

[Bro88] Brockett, R. W., "On the computer control of movement," IEEE Int'l. Conf. on Robotics and Automation, 1988.

[Bro94] Brockett, R. W., "Dynamical systems and their associated automata, in Systems and Networks, Mathematical Theory and Applications, Academie Verlag, Berlin, 1994.

[Cao93] Cao, T., and Sanderson, A. C., “A fuzzy petri net approach to reasoning about uncertainty in robotic systems,” IEEE Int'l. Conf. on Robotics and Automation, 1993.

[Cut93] Cutkosky, M. R., and Hyde, J. M., “Manipulation control with dynamic tactile sensing”, 6th International Symposium on Robotics Research, Hidden Valley, Pennsylvania, 1993.

[Ebe94] Eberman, B., and Salisbury, J. K., “Application of change detection to dynamic contact sensing,” Int’l Journal of Robotics Research, v. 13, n. 5, pp. 369-394, 1994.

[Hog85] Hogan, N., “Impedance control: an approach to manipulation: parts I, II, and III,” ASME Journal of Dynamic Systems, Measurement, and Control, v. 107, pp. 1-24, 1985.

[How90] Howe, R. D., “Dynamic tactile sensing,” Ph.D. Thesis, Stanford University, October 1990.

[Hyd93] Hyde, J. M., and Cutkosky, M. R., “Contact transition control: an experimental study,” IEEE Int'l. Conf. on Robotics and Automation, 1993.

[Hyd95] Hyde, J. M., “A phase management framework for event-driven dextrous manipulation,” Stanford University Ph.D. thesis, 1995.

[Joh91] Johansson, R. S., and Westling, G., “Afferent signals during manipulative tasks in man,” In Franzen, O., Westman, J. (eds.): Information processing in the somatosensory system: Proceedings of an International Seminar at the Wenner-Gran Center, Macmillan, New York, 1991.

[Kat94] Katayama, Y., Nanjo, Y., and Shimokura, K., “Event-driven motion-module switching mechanism for robot motion control: concept and experiment,” ASME Journal on Dynamic Systems and Control, v. 55, n. 1, 1994.

[Kha87] “Unified approach for motion and force control of robot manipulators: the operational space formulation,” IEEE Journal of Robotics and Automation, v. 3, n. 1, pp. 43-53, 1987.

[McC93] McCarragher, B. J., and Asada, H., “A discrete event approach to the control of robotic assembly tasks,” IEEE Int'l. Conf. on Robotics and Automation, 1993.

[Nag93] Nagai, K., and Yoshikawa, T., “Dynamic manipulation/grasping control of multi-fingered robot hands,” IEEE Int'l. Conf. on Robotics and Automation, 1993.

[Sch89] Schneider, S. A., “Experiments in the dynamic and strategic control of cooperating manipulators,” Stanford University Ph.D. thesis, 1989.

[Sob92] Sobh, T. M., and Bajcsy, R., “Autonomous observation under uncertainty,” IEEE Int'l. Conf. on Robotics and Automation, 1992.

[Son94] Son, J. S., Monteverde, E. A., and Howe, R. D., “A tactile sensor for localizing transient events in manipulation,” IEEE Int'l. Conf. on Robotics and Automation, 1994.

[Tre93] Tremblay, M. R., and Cutkosky, M. R., “Estimating friction using incipient slip sensing during a manipulation task,” IEEE Int'l. Conf. on Robotics and Automation, 1993.

[Tre95] Tremblay, M. R., “Using multiple sensors and contextual information to detect events during a manipulation task,” Stanford University Ph.D. thesis, 1995.