Sequential Composition of Dynamically Dexterous Robot Behaviors

R. R. Burridge

Texas Robotics and Automation Center
Metrica, Inc.
Houston, Texas 77058, USA
burridge@mickey.jsc.nasa.gov

A. A. Rizzi

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3891, USA
arizzi+@ri.cmu.edu

D. E. Koditschek

Artificial Intelligence Laboratory and Controls Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan 48109-2110, USA
kod@eecs.umich.edu

Abstract

We report on our efforts to develop a sequential robot controller-composition technique in the context of dexterous "batting" maneuvers. A robot with a flat paddle is required to strike repeatedly at a thrown ball until the ball is brought to rest on the paddle at a specified location. The robot's reachable workspace is blocked by an obstacle that disconnects the free space formed when the ball and paddle remain in contact, forcing the machine to "let go" for a time to bring the ball to the desired state. The controller compositions we create guarantee that a ball introduced in the "safe workspace" remains there and is ultimately brought to the goal. We report on experimental results from an implementation of these formal composition methods, and present descriptive statistics characterizing the experiments.

KEY WORDS—hybrid control, controller composition, dynamical dexterity, switching control, reactive scheduling, backchaining, obstacle avoidance

1. Introduction

We are interested in tasks requiring dynamical dexterity—the ability to perform work on the environment by effecting changes in its kinetic as well as potential energy. Specifically, we want to explore the control issues that arise from dynamical interactions between robot and environment, such as active balancing, hopping, throwing, catching, and juggling. At the same time, we wish to explore how such dexterous behaviors can be marshaled toward goals whose achievement requires at least the rudiments of strategy.

In this paper, we explore empirically a simple but formal approach to the sequential composition of robot behaviors. We cast "behaviors" —robotic implementations of user-specified tasks—in a form amenable to representation as state regulation via feedback to a specified goal set in the presence of obstacles. Our approach results ideally in a partition of state space induced by a palette of pre-existing feedback controllers. Each cell of this partition is associated with a unique controller, chosen in such a fashion that entry into any cell guarantees passage to successively "lower" cells until the "lowest," the goal cell, has been achieved. The domain of attraction to the goal set for the resulting switching controller is shown to be formed from the union of the domains of the constituent controllers.[1]

1.1. Statement of the Problem

We have chosen the task domain of paddle juggling as one that epitomizes dynamically dexterous behavior. The ball will fall to the floor unless repeatedly struck from below or balanced on the paddle surface. Thus the dynamics of the environment

The International Journal of Robotics Research
Vol. 18, No. 6, June 1999, pp. 534-555,

1. Portions of this paper have appeared in an earlier work (Burridge 1996).

necessitate continual action from the robot. Moreover, the intermittent nature of the robot-ball interaction provides a focused test bed for studying the hybrid (mixed continuous and discrete) control problems that arise in many areas of robotics. In this paper, the machine must acquire and contain a thrown ball, maneuver it through the workspace while avoiding obstacles that necessitate regrasping[2] along the way, and finally bring it to rest on the paddle at a desired location. We refer to this task as *dynamical pick and place* (DPP). By explicitly introducing obstacles,[3] and requiring a formal guarantee that the system will not drive the ball into them, we add a strategic element to the task.

From the perspective of control theory, our notion of "behavior" comprises merely the closed-loop dynamics of a plant operating under feedback. Yet for our task, as is often the case when trying to control a dynamical system, no available feedback control algorithm will successfully stabilize as large a range of initial conditions as desired. Instead, there exists a collection of control laws, each capable of stabilizing a different local region of the system's state space. If they were properly coordinated, the region of the workspace across which their combined influence might be exerted would be significantly larger than the domain of attraction for any one of the available local controllers. The process of creating a switching strategy between control modes is an aspect of hybrid control theory that is not presently well established. In this paper, we use conservative approximations of the domains of attraction and goal sets of the local controllers to create a "prepares" graph. Then, backchaining away from the controller that stabilizes the task goal, we use the graph to create a partition of the state space of the robot into regions within each of which a unique controller is active. This leads to the creation of complex switching controllers with formal stability properties, and this specific instance of stable hybrid control represents our present notion of behavioral composition.

To test our methods, we use a three-degree-of-freedom robot whose paddle-juggling capabilities are already well established (Rizzi 1994). Although our available juggling control laws induce very good regulation about their respective goal sets, there will always be ball states that each cannot contain: the goal sets have limited domains of attraction. However, many of the states that are "uncontainable" by one control law can be successfully handled by another with different parameter settings, such as a new goal point, or perhaps different gain settings. In this paper, we resort solely to such parameter retunings to explore the problem of behavioral composition. For all aspects of the robot's task—containing the initial throw, moving the ball through the workspace, negotiating the obstacle that divides the workspace, and finally bringing the ball to rest on the paddle—we require that it use members of an established palette of juggling control laws with different goal points or gain tunings.

Our group has explored paddle juggling for some time (Rizzi, Whitcomb, and Koditschek 1992; Rizzi 1994), and it should be emphasized here that this is not a paper about juggling per se. Rather, our juggling apparatus provides a convenient test bed for the more general switching control methods presented in Section 3, which necessitate the exploration of the juggling behavior described in Section 2.5.

1.2. Previous Literature

1.2.1. Pick and Place in Cluttered Environments

Our focus on the dynamical pick-and-place problem is inspired in part by the Handey system developed by Lozano-Pérez and colleagues (Lozano-Pérez, et al. 1987), who emphasized the importance of developing task-planning capabilities suitable for situations where regrasping is necessitated by environmental clutter (see footnote 1); however, our setup presents dynamical rather than quasi-static regrasping problems. Indeed, our emphasis on robustness and error recovery as the driving considerations in robot task planning derives from their original insights on fine-motion planning (Lozano-Pérez, Mason, and Taylor 1984), and the subsequent literature in this tradition bears a close relation to our concerns, as is touched upon below.

Comprehensive work by Tung and Kak (1994) yielded a contemporary quasi-static assembly environment in the best traditions of the Handey apparatus. The system built a plan based on sensed initial conditions, spawned an execution process that sensed exceptions to the planned evolution of states, and produced actions to bring the world's state back to the intermediate situation presumed by the plan. We wish to examine the ways in which this higher level "feedback" (local replanning *is* a form of feedback) can be reasoned about and guaranteed to succeed, albeit in the drastically simplified setting of the dynamical pick-and-place problem outlined above.

1.2.2. Hybrid Control and Discrete-Event Systems

The difficult question of how to reason about the interplay between sensing and recovery at the event level in robotics has been considerably stimulated by the advent of the Ramadge-Wonham DES control paradigm (Ramadge and Wonham 1987). In some of the most careful and convincing of such DES-inspired robotics papers, Lyons proposed a formalism for encoding and reasoning about the construction of action plans and their sensor-based execution (Lyons 1993;

2. In general, regrasping is required when the free configuration space attendant upon the current grasp is disconnected, and the goal does not lie in the presently occupied connected component. As will be seen, in the DPP problem the configuration space of the "palming" grasp disconnects the goal from the component in which the ball is typically introduced. The robot can only "connect" the two components by adopting a "juggling" grasp, whose configuration space has a dramatically different topology.

3. The finite length of the paddle has always induced an implicit obstacle, but we have not hitherto modeled or accounted for it in our juggling work (Rizzi, Whitcomb, and Koditschek 1992).

Lyons and Hendriks 1994). In contrast to our interest, however, Lyons explicitly avoided the consideration of problems wherein geometric and force-sensor reports must be used to estimate progress, and thereby stimulate the appropriate event transitions.

1.2.3. Error Detection and Recovery

Despite the many dissimilarities in problem domain, models, and representation, the work discussed here seems to bear the closest correspondence to the "fine-motion planning with uncertainty" literature in robotics (for example, as discussed in Latombe's text (Latombe 1991)) originated by Lozano-Pérez, Mason, and Taylor (1984). Traditionally, this literature focuses on quasi-static problems (generalized damper dynamics (Whitney 1977) with Coulomb reaction forces (Erdmann 1984) on the contact set). Furthermore, the possible control actions typically are restricted to piecewise constant-velocity vectors, whereby each constant piece is corrupted by a constant disturbance vector of bounded magnitude and unknown direction. In contrast, we are interested in Newtonian dynamical models, and our control primitive is not a constant action, but rather the entire range of actions consequent on the closed-loop policy, Φ, to be specified below. Differences in models notwithstanding, we have derived considerable motivation from the "LMT" (Lozano-Pérez, Mason, and Taylor 1984) framework as detailed in Section 1.3.3.

1.2.4. Juggling

In an earlier work (Burridge, Rizzi, and Koditschek 1997), we introduced the notions of dynamical safety and obstacle avoidance, and in 1995 we discussed the controller-composition algorithm mentioned in Section 3 (Burridge, Rizzi, and Koditschek 1995). The major contribution of this paper is in furthering our work by showing how to compose controllers in such a manner as to avoid an obstacle that disconnects the workspace.

In previous work in our lab (Buehler, Koditschek, and Kindlmann 1990b; Rizzi and Koditschek 1994), a great deal of attention has been paid to the lower level palette of controllers. Our architecture implements event-driven robot policies whose resulting closed-loop dynamics drive the coupled robot-environment state toward a goal set. We strive to develop control algorithms that are sufficiently tractable as to allow correctness guarantees with estimates of the domain of attraction as well. Thus, we have focused theoretical attention on "practicable stability mechanisms"—dynamical systems for which effectively computable local tests provide global (or, at least, over a "large" volume) conclusions. At the same time, we have focused experimental attention on building a distributed computational environment that supports flexibly reconfigurable combinations of sensor and actuator hardware controllers, motion estimation and control algorithms, and event-driven reference-trajectory generators. The result has been a series of laboratory robots that exhibit indefatigable goal-seeking behavior, albeit in a very narrow range of tasks. In this paper, we explore the "higher level" problem of generalizing the range of tasks by composing the existing controllers, achieving new tasks that no single existing controller is capable of in isolation.

1.3. Overview of the Approach

1.3.1. Feedback Confers Robustness

Physical disturbances to a system can be classified, broadly speaking, into two categories. The first includes the small, persistent disturbances arising from model inaccuracies and sensor and actuator noise. Ball spin, air drag on a ball modeled as a point mass, and spurious changes in camera measurements introduced by lighting variations in a scene represent typical examples of this first type of disturbance. The second category includes large, significant changes to the world as expected. For example, if a parts tray is severely jostled or arrives unexpectedly empty, an assembly robot's world model is likely to have been grossly violated. Such large disturbances occur very rarely (otherwise, they would have been accounted for systematically by the robot's designers!), but are potentially catastrophic.

A common tradition in robotics and control has been to model these disturbances in the hope of treating them rationally. One typically appeals to stochastic representations of small perturbations, and introduces an optimal design (for example, a Kalman filter) to handle them. Analogously, recovery from catastrophe is similarly model-based, typically being addressed by special "exception-handling" routines specifically written for each particular failure anticipated. But disturbances are generally left out of nominal models precisely because they are so difficult to model. The systematic origin of the typical small disturbance often defies a stochastic representation, and the resulting optimal designs may be overly aggressive. If the human system designers haven't imagined a severe failure mode, then the system will be unable to react to it when it occurs, with possibly disastrous results.

Feedback-controlled systems can be designed to confer a level of robustness against both types of disturbance without recourse to explicit disturbance models. On one hand, asymptotically stable systems are generically locally structurally stable. This means that the first class of disturbances can degrade performance (e.g., biasing the goal set, or coarsening the neighborhood of the goal set to which all initial conditions in its domain of attraction are eventually brought), but if small enough, they cannot destroy the stability of the (possibly biased) goal. On the other hand, if the closed-loop system induced by a feedback controller exhibits a global (or at least "large") domain of attraction, then large disturbances can never "take it by surprise," as long as the original state

space and dynamical model used to represent the world (not the disturbance) remain valid following the perturbation. The system will merely continue to pursue the same shepherding function, now applied at the new state, with the same tireless progression toward the goal as before. Of course, such disturbances need to be rare relative to the time required to recover from them, or we should not expect the system to ever approach the goal.

1.3.2. Feedback Strategies as Funnels

A decade ago, Mason introduced the notion of a funnel as a metaphor for the robust, self-organizing emergent behavior displayed by otherwise unactuated masses in the presence of passive mechanical guideways (Mason 1985).[4] He understood that the shape of the guideways constituted an embodied computational mechanism of control that ought to be rendered programmable. In this paper, we embrace Mason's metaphor and seek to extend it to active electromechanical systems (i.e., possessing sensors and actuators) which are more obviously programmable.

Figure 1 depicts the idealized graph of a positive-definite scalar-valued function centered at a goal point—its only zero and unique minimum. If a feedback control strategy results in closed-loop dynamics on the domain (depicted as the $x - y$ plane in these figures) such that all motions, when projected up onto the funnel, lead strictly downward ($\dot{z} < 0$), then this funnel is said to be a Lyapunov function. In such a case, Mason's metaphor of sand falling down a funnel is particularly relevant, since all initial conditions on the plane lying beneath the over-spreading shoulders of the funnel (now interpreted as the graph of the Lyapunov function) will be drawn down the funnel toward its center—the goal.

Although the funnels strictly represent Lyapunov functions, which may be difficult to find, we note that such functions must exist within the domain of attraction for all asymptotically stable dynamical systems. Thus we loosen our notion of "funnels," and let them represent attracting dynamical systems, with the extent of the funnel representing a known invariant region (for a Lyapunov function, this would be any of the level curves).

In Figure 2, we depict a complicated boundary such as might arise when the user specifies an "obstacle"—here, all states outside the bounded area of "good" states.[5] Ideally, we would like a control strategy that funnels all "good" states to the goal, without allowing any to enter the obstacle. We suggest this in the figure by sketching a complex funnel whose shape is exactly adapted to the domain.[6] Unfortunately, it is

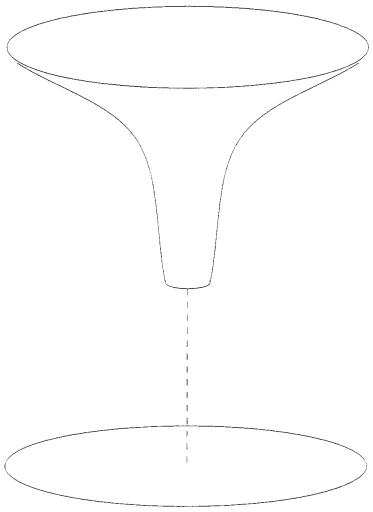

Fig. 1. The Lyapunov function as a funnel: an idealized graph of a positive-definite function over its state space.

usually very difficult or impossible to find globally attractive control laws, so Figure 2 often cannot be realized, for practical reasons. Moreover, there is a large and important class of mechanical systems for which no single funnel (continuous-stabilizing feedback law) exists (Brockett 1983; Koditschek 1992). Nevertheless, one need not yet abandon the recourse to feedback.

1.3.3. Sequential Composition as Preimage Backchaining

We turn next to a tradition from the fine-motion planning literature, introduced by Lozano-Pérez, Mason, and Taylor (1984)—the notion of preimage backchaining. Suppose we have a palette of controllers whose induced funnels have goal sets that can be placed at will throughout the obstacle-free state space. Then we may deploy a collection of controllers whose combined (local) domains of attraction cover a large part of the relevant region of the state space (e.g., states within some bound of the origin). If such a set is rich enough that the domains overlap and each controller's goal set lies within the domains of other controllers, then it is possible to *backchain* away from the overall task goal, partitioning the state space into cells in which the different controllers will be active, and arrive at a scheme for switching between the controllers that will provably drive any state in the union of all the various domains to the single task-goal state.

This method of combining controllers in sequence is depicted in Figure 3. Note that the controller represented by each funnel is only active when the system state is in the

4. Other researchers in robotics and fine-motion planning have introduced comparable metaphors (Ish-Shalom 1984; Lozano-Pérez, Mason, and Taylor 1984).

5. Of course, this picture dramatically understates the potential complexity of such problems, since the free space need not be simply connected as depicted.

6. Rimon and Koditschek (1992) addressed the design of such global funnels for purely geometric problems, wherein the complicated portion of the domain was limited to the configuration space. In contrast, in this paper we are concerned with complex boundaries that run through the velocities as well—e.g., see Fig. 11.

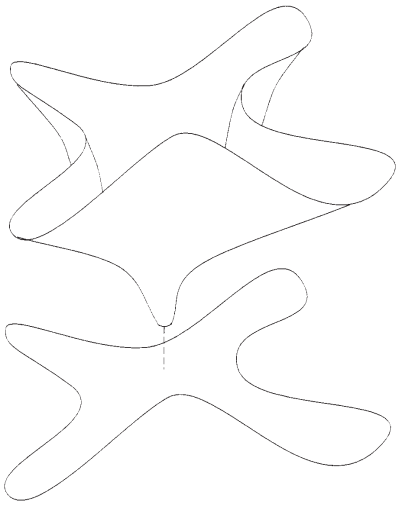

Fig. 2. An ideal funnel capturing the entire obstacle-free state space of the system.

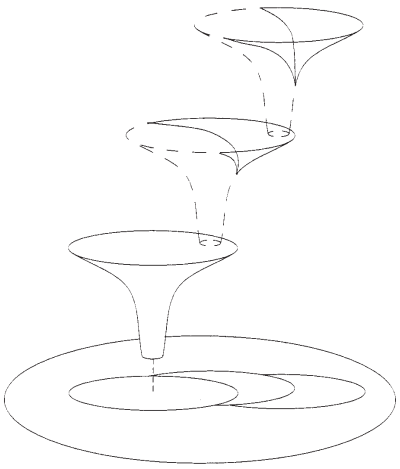

Fig. 3. The sequential composition of funnels. The goal point of each controller lies within the domain of attraction induced by the next-lower controller. Each controller is only active outside the domains of lower controllers. The lowest controller stabilizes the system at the final destination.

appropriate region of the state space (beyond the reach of lower funnels), as sketched at the bottom of the figure, and indicated by the dashed portion of the funnel. As each controller drives the system toward its local goal, the state crosses a boundary into another region of state space where another controller is active. This process is repeated until the state reaches the final cell, which is the only one to contain the goal set of its own controller.

We are now in a position to approximate Figure 2 with Figure 4, and this simple idea captures the essence of the contribution of this paper.

This scheme affords the same robustness against disturbances discussed above. Local stability against small disturbances is obtained, because each funnel implies asymptotic stability for that particular controller and goal point. It also provides robustness against large, unanticipated disturbances, because the control system has no higher-level state, or preconceived plan. If the state of the system shifts to a distant location, then the controller appropriate for *that particular region of state space* automatically activates, and the process begins again. Similarly, if the state takes a favorable leap toward the goal state, or starts in the goal partition, then intermediate controllers will never be activated. Thus, as long as the disturbance doesn't send the state outside the union of all the domains in the partition, it will be effectively handled.

As we try to suggest in Figure 4, this version of preimage backchaining is particularly convenient when the obstacle-free state space has a complicated geometry. Because the domains used for the partition of state space must be invariant, there is a natural notion of safety that arises from this approach. If there are obstacles in the state space, but they

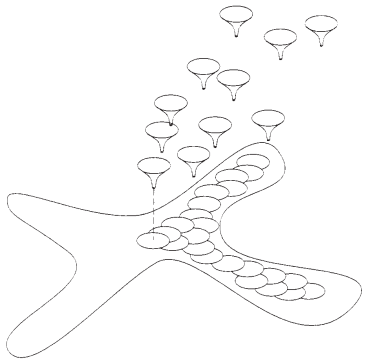

Fig. 4. A suitable composition of local funnels partitions the obstacle-free state space (or a close approximation to it) into cells. Within each cell, a unique controller will be active. See Figure 16 for a composition of controllers that avoids the obstacles in the real robot's state space.

do not intersect any of the cells of the partition, then it follows that the state will never reach an obstacle while under the influence of the composite controller.

1.4. Contributions of the Paper

When written a little more carefully, as is done below, it is fairly easy to verify the theoretical validity of the backchaining approach outlined above. In the absence of noise, each local controller is guaranteed by definition to drive its entire domain to its goal. In doing so, the controller forces the system into

the domain of a higher-priority controller. This process must repeat until the domain of the goal controller is reached.

For the nonlinear problems that inevitably arise in robotics, however, invariant domains of attraction are difficult to derive in closed form, so the utility of the approach is called into question. Moreover, physical systems are inevitably "messier" than the abstracted models their designers introduce for convenience, and the question arises whether these ideas incur an overly optimistic reliance on closed-loop models. Modeling errors may, for instance, destroy the correctness of our method, allowing cycling among the controllers. Thus we must turn to empirical tests to verify that our formally correct method can produce useful controllers in practice.

The experimental environment for our juggling robot is highly nonlinear and "messy," and challenges the idealized view of the formal method in several ways. We have no closed-form Lyapunov functions from which to choose our domains of attraction. Instead, we are forced to use conservative invariant regions found using experimental data and numerical simulation. Also, there is significant local noise in the steady-state behavior of the system, so that even our "invariant" regions are occasionally violated. Nevertheless, as will be shown, our composition method proves to be very robust in the face of these problems. We are able to create a palette of controllers that successfully handles a wide range of initial conditions, persevering to the task goal while avoiding the obstacles, despite the significant noise along the way.

1.5. Organization of the Paper

This paper follows a logical development roughly suggested by Figures 1–4. In Section 2, we provide a high-level overview of the hardware and software, describe how entries in the controller palette (suggested by Fig. 1) arise, and introduce the *Juggle*, Φ_J, a controller used as a basis for all entries in our palette. We also introduce variants of Φ_J: catching, Φ_C; and palming, Φ_P. In Section 3, we introduce the notion of a safe controller suggested by Figure 2, as well as formally define the sequential composition depicted in Figure 3. In Section 4, we implement the partition suggested by Figure 4, and report on our empirical results.

2. The Physical Setting

2.1. The Hardware System

For the experiments described in this paper, we use the "Bühgler," a three-degree-of-freedom, direct-drive machine sketched in Figure 5a, which has been discussed in a variety of other settings (Whitcomb, Rizzi, and Koditschek 1993; Rizzi 1994; Rizzi and Koditschek 1996). Two cameras provide images of the ping-pong ball(s) at 60 Hz, and all computation, from the image processing to the generation of the analog signals to the motor amplifiers, is carried out on a network of 20 transputers.

Figure 5b shows the workspace from above, and depicts the various obstacles with which the robot must contend. The inner and outer edges of the annular workspace are caused by the "shoulder" offset and finite paddle length, respectively. The visual boundaries are considered as obstacles, because if the ball strays beyond these lines, the robot will lose track of it and fail to juggle it. Finally, a flat obstacle (which we refer to as the *beam*) is introduced that disconnects the horizontal workspace. The beam is very thin in the vertical dimension, protruding into the workspace just above the plane of zero height, and is situated as shown in Figure 5b. The robot may pass under it, and the ball may pass over it, but no contact is allowed between the robot and the ball in the region it occupies, and neither is allowed to touch it. Thus, in addressing the problem of how to bring a ball from one side of the beam to the other, we encounter a simple situation wherein a regrasp maneuver (see footnote 1) is necessary for the task to succeed.

Using the tangent-space notation of Abraham and Marsden (1978), we denote the position of the ball by $b \in \mathcal{B} := \mathbb{R}^3$, and the full state of the ball by $Tb = (b, \dot{b}) \in T\mathcal{B}$. Similarly, the position of the robot is denoted by $q = (\phi, \theta, \psi) \in \mathcal{Q} := S^3$, and the full state by $Tq = (q, \dot{q}) \in T\mathcal{Q}$. While the ball is in flight, we use the standard Newtonian flight model, ignoring all effects other than gravity (which is assumed to be a constant acceleration acting vertically downward). Robot–ball collisions are assumed to be instantaneous, and are modeled using Newton's restitution model. The reader is referred to the work of Rizzi (1994) for more detailed discussion of these models.

2.2. The Software System

The full software system is depicted in Figure 6, with the various blocks representing high-level subsystems. Rectangles represent the subsystems with considerable internal state: vision, observer, control, and actuator; while the circle in the middle represents a transformation that has no significant internal state.

Ball states, Tb, are interpreted at 60 Hz by the vision system, V (cameras and image processing), which produces image-plane locations at the same rate. These image-plane locations are passed to the observer, O, which triangulates estimates of ball position $\hat{b} \in \hat{\mathcal{B}}$, and uses standard observer methods to estimate the full state of the ball, $T\hat{b} = (\hat{b}, \hat{\dot{b}}) \in T\hat{\mathcal{B}}$. The estimated ball state, $T\hat{b}$, is fed back to V, to direct its focus of attention, and also interpolated to produce estimates at 330 Hz for the rest of the system. This nonlinear V–O sensing loop is discussed in detail and shown to be asymptotically stable in another work (Rizzi and Koditschek 1996). In the present work, we have simply assumed that convergence, $T\hat{b} \to Tb$, occurs at a time scale much faster than that relevant to the larger loop, an assumption which occasionally proves invalid, as discussed in Section 4.4.3.

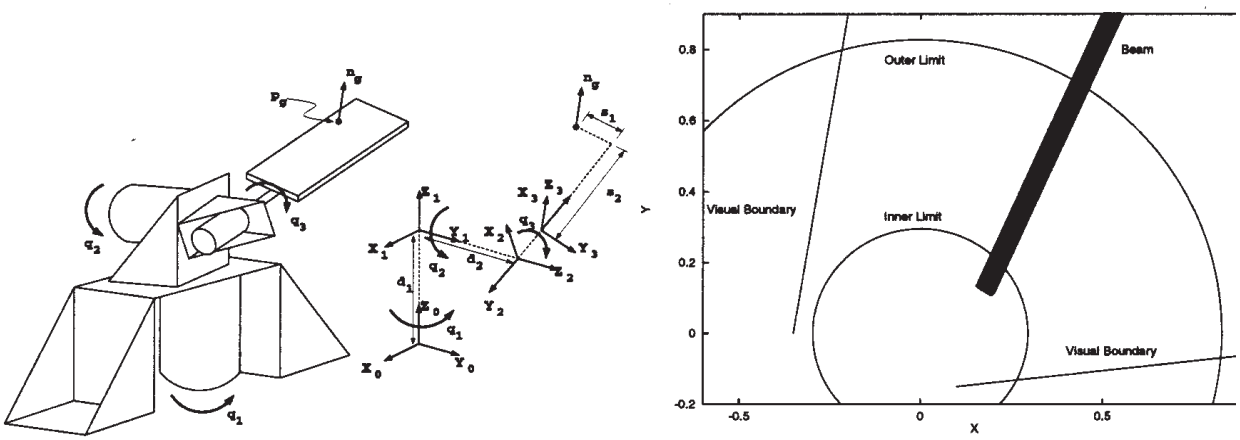

Fig. 5. The Bühgler Arm. The three joint values, q_1, q_2, and q_3 are referred to as ϕ, θ, and ψ, respectively, in this paper (a). The horizontal workspace with obstacles: the beam, inner and outer paddle limits, and the boundary of the visible workspace (b).

Estimated ball states, $T\hat{b}$, are passed through a memoryless transformation, $m : T\hat{\mathcal{B}} \rightarrow \mathcal{Q}$, to produce reference robot positions, $r = m(T\hat{b})$. Since m is analytic, this block also produces $\dot{r}$ and $\ddot{r}$ for use by the control system. Following the work of Buehler, Koditschek, and Kindlmann (1990a), the m used for juggling leads to robot motion that appears to be a distorted reflection of the ball's motion, and we call it a *mirror law*. The juggling mirror law used in this paper was developed by Rizzi, and a detailed description is available (Rizzi 1994). The mirror law uses nine key parameters that prescribe the spatial location and feedback gains for the juggle behavior. In this work, we are mostly interested in modifying the location of the *set point*, $\mathcal{G}$ (denoting the three parameters used to prescribe horizontal position and the apex height of the ball's desired periodic vertical trajectory), thereby defining juggling control strategies that are active in different regions of the workspace. We also modify the vertical energy gains to create the palming behavior. The juggling mirror law is discussed in more detail in Section 2.5, and we introduce control laws for other behaviors in Section 2.6.

The reference robot states created by the mirror law are fed to an inverse-dynamics joint-space controller (Whitcomb 1992; Whitcomb, Rizzi, and Koditschek 1993), C, which produces torque values, τ. The torques are used by the actuator block, A (amplifiers, motors, etc.), which generates true robot-joint states, $Tq \in T\mathcal{Q}$. The robot states are sensed and returned to C for feedback control. The C–A control/actuation loop has been shown to be globally asymptotically stable, so that $Tq \rightarrow Tr$ whenever $\ddot{r}$ is continuous. As for the sensor block, V–O, we have assumed that the transients of the C–A loop have died out by the time of any robot–ball interaction, and, hence, that $Tq = Tr$. Again, in practice, the transients are generally rapid enough to support this simplified reasoning; however, in some situations we inevitably pay a price in performance. This topic is further explored in Section 4.4.3.

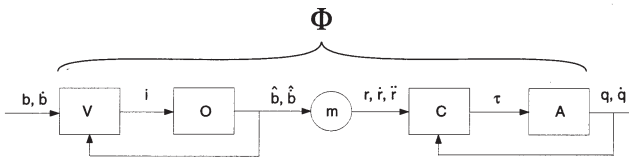

Fig. 6. Flow chart showing the various functional blocks of the system: vision, V; observer, O; mirror law, m; control, C; and actuation, A. The parameters of interest to this paper all reside in m.

Together, all the blocks depicted in Figure 6 form a dynamical plant, or filter, relating inputs Tb to outputs Tq. We denote this filter by Φ, and loosely refer to it as a "controller."

2.3. The Closed-Loop System

Figure 7 depicts the interconnected robot-environment (ball) system that arises when a falling ball excites the visual field of the hardware and software system just described. We denote the resulting closed-loop dynamics F_Φ, where the subscript reflects the dependence of the robot's reactions, and hence the evolution of the coupled robot-environment states, on the blocks of Φ depicted in Figure 6. It turns out that the repetitive continuous trajectories of this closed-loop system can be much more readily understood by passage to an induced discrete-event sampled mapping—the "return map," of the periodic orbits, that we denote by f_Φ. This map (more exactly, its iteration considered as a dynamical system) represents the

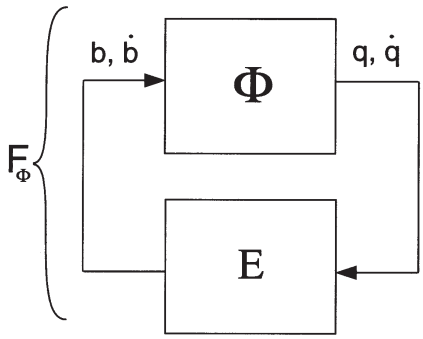

Fig. 7. The closed-loop dynamics, F_Φ, induced by Φ and the environment, E.

focus of analysis and experiment throughout the remainder of the paper. We now describe in some detail the manner in which the robot controller, Φ, gives rise to the closed-loop dynamics, F_Φ, and the relationship of these continuous dynamics to the central object of study, f_Φ.

2.3.1. Collapse of Dimension to Ball Apex States

The robot controller, Φ, has been expressly designed to cause all of the robot's internal state to "collapse" onto a lower-dimensional local embedding of the environment's state in a manner determined by the mirror law, m. In consequence, after initial transients, the closed-loop dynamics, F_Φ, can be written in the coordinates of the ball apex states, as we now explain. A much more thorough presentation of these ideas may be found in an earlier work (Buehler, Koditschek, and Kindlmann 1990b).

Consider the "total system" configuration space, $\mathcal{X} := \mathcal{B} \times \mathcal{Q} \times \hat{\mathcal{B}}$, and its tangent space, $\mathcal{TX} = \mathcal{TB} \times \mathcal{TQ} \times \mathcal{T}\hat{\mathcal{B}}$, that represents the state space for the coupled robot-environment pair. Given the physical models introduced above, the dynamics of this coupling, denoted X_Φ, are induced by Φ, Newtonian ball flight, and impacts between robot and ball. In particular, the coupling between robot and environment occurs via collisions that take place on the *contact set*, $\mathcal{C} \subset \mathcal{X}$, comprising those configurations where the ball and the robot's paddle are touching. Our simple impact model introduced in Section 2.1 amounts to a morphism (Hirsch 1976) of $\mathcal{T}_\mathcal{C}\mathcal{X}$ into itself [7]—a rule by which a new ball velocity at contact is determined by a linear combination of its old velocity and that of the robot's paddle normal, with positions instantaneously unchanged.

7. The ball's velocity changes discontinuously while its position remains fixed. In contrast, since the mirror law specifies robot positions that depend on the ball velocities, the robot's position is commanded to change discontinuously after each impact (Rizzi and Koditschek 1993). Of course, this discontinuity can never be tracked "instantaneously" by the robot controller, and the impact event represents one of the chief events that belie our naive assumption in the next paragraph concerning the invariance of $\mathcal{M}$.

Define the *mirror surface* $\mathcal{M} \subset \mathcal{TX}$ to consist of those states satisfying simultaneously the constraint equations $T\hat{b} = Tb$ and $Tq = Tm(Tb)$.[8] The first of these corresponds to an error-free estimator (the internal states of V–O in Fig. 6 having converged), and the second corresponds to an error-free controller (the internal states of C–A in Fig. 6 having converged). Since $\mathcal{M}$ is attracting and invariant under X_Φ, we concentrate on the restriction dynamics, $X_\Phi \mid \mathcal{M}$, as mentioned in Section 2.2. We find it most convenient to do so in "ball coordinates," as follows. Notice that the mirror surface, $\mathcal{M}$, being the graph of a function

$$M : \mathcal{TB} \to \mathcal{TX}; \qquad Tb \mapsto (Tb, Tm(Tb), Tb), \tag{1}$$

is parameterized by $\mathcal{TB}$ using M as a global change of coordinates. Take F_Φ to be the representation of the restriction dynamics, $X_\Phi \mid \mathcal{M}$, expressed in $\mathcal{TB}$ coordinates via M. In symbols, denoting the respective flows (i.e., the time trajectories generated by the respective vector field) by the superscript t, we have $F_\Phi^t := M^{-1} \circ (X_\Phi \mid \mathcal{M})^t \circ M$.

The mirror law, m, is so designed that if the robot's paddle had infinite extent, and if the robot's visual apparatus had an unbounded view, then the ball's flight under gravity would ensure a return to the contact set after each collision. In ball coordinates, this amounts to asserting that $\mathcal{P} := M^{-1}(\mathcal{T}_\mathcal{C}\mathcal{X} \cap \mathcal{M}) \subset \mathcal{TB}$ is a global Poincaré section (Guckenheimer and Holmes 1983; Koditschek and Buehler 1991) for the relaxation oscillator F_Φ.[9] From the computational and conceptual points of view, it turns out that the collisions, $\mathcal{P}$, are less convenient to examine than the set of apex points, at which the ball's vertical velocity vanishes. Namely, decompose the ball's (Euclidean) configuration space into its one-dimensional vertical and two-dimensional horizontal components, $\mathcal{B} = \mathcal{V} \times \mathcal{H}$, with the induced state-space decomposition, $\mathcal{TB} = \mathcal{TV} \times \mathcal{TH}$. Denote by $F_\Phi^{-t_a}$ the map that takes a ball just before collision with the robot back to its previous apex. Then using this last change of coordinates,

$$F_\Phi^{-t_a} : \mathcal{P} \to \mathcal{TH} \times \mathcal{V},$$

we may study the iterates of the ball's return to apex on the transformed Poincaré section, $\mathcal{TH} \times \mathcal{V} \subset \mathcal{TB}$.

2.3.2. The Horizontal Return Map at Ball Apex, f_Φ

Denote by (f_Φ, g_Φ) the manner in which the ball's state at a previous apex gives rise to its state at the next apex—this is a

8. This is a convenient abuse of notation, since in reality, $Tq = Tm(TTb)$. However, for our simplified models $TTb = N(Tb)$, where N denotes the six-dimensional vector field of Newtonian flight in gravity. Hence, to be truly precise we would have to write $Tq = Tm \circ N(Tb)$, a usage whose complexity seems not to be justified by any gain in understanding and whose abandonment should not cause any confusion.

9. Since the paddle is finite and the cameras admit a relatively narrow viewing area, $\mathcal{P}$ is in fact only a local Poincaré section, and part of the obstacle-avoidance behavior to be discussed at length in the sequel involves modifying Φ so that $\mathcal{P}$ is indeed always visited forever into the future.

mapping from $\mathcal{TH} \times \mathcal{V}$ onto itself. Buehler, Koditschek, and Kindlmann (1990a, 1990b) showed that when the juggling mirror law is restricted to $\mathcal{V}$, g_Φ has global attraction to $\mathcal{G}_V$ (the vertical component of the set point). In contrast, only local attraction of (f_Φ, g_Φ) to $\mathcal{G}_H \times \mathcal{G}_V$ has been demonstrated to date.

Furthermore, Rizzi and Koditschek (1994) showed that f_Φ and g_Φ are nearly decoupled. Indeed, empirical results confirm that the horizontal behavior of f_Φ changes very little within a large range of apex heights, and that the vertical system is regulated much faster than the horizontal.

Thus, except where otherwise specified, we ignore $\mathcal{V}$ throughout this paper, and concentrate our attention on the evolution of f_Φ in $T\mathcal{H}$. Accordingly, for ease of exposition, we denote the horizontal goal set $\mathcal{G}$ (dropping the $\mathcal{H}$ subscript), a singleton subset of $T\mathcal{H}$ comprising a desired horizontal position at zero velocity.

2.4. Obstacles

In this section, we briefly describe the origin of the obstacles in $\mathcal{TH} \times \mathcal{V}$ with which we are concerned. We must start by reverting back to the full six-dimensional space, $T\mathcal{B}$, and the restriction flow F^t_Φ. Although we attempt to be fairly thorough in this section, some details are omitted in the interest of readability. The reader is referred to two other works (Rizzi 1994; Burridge 1996) for the missing details.

There are three types of physical obstacle with which we are concerned. The first consists of regions of the robot space that are "off limits" to the robot, $\mathcal{O}_\mathcal{Q} \subset \mathcal{Q}$, whether or not it is in contact with the ball. Robot obstacles include the floor, joint limits, etc., as well as any physical obstacle in the workspace, and have been studied in the kinematics literature. Recall that for a particular mirror law, m, each ball state induces a corresponding robot state. Stretching our terminology, we loosely define $m^{-1}(\mathcal{O}_\mathcal{Q}) = \{Tb \in T\mathcal{B} | m(Tb) \in \mathcal{O}_\mathcal{Q}\}$.

The second type of physical obstacle consists of subsets of ball state space which are off limits to the ball, $\mathcal{O}_{T\mathcal{B}} \subset T\mathcal{B}$, whether or not it is touching the robot. This set may include walls, the floor, visual boundaries, and any other objects in the workspace that the ball should avoid (such as the beam). Note that this region resides in the full state space of the ball, so for example, if the vision system fails for ball velocities above a certain value at certain locations, such states can be included in $\mathcal{O}_{T\mathcal{B}}$.

The final type of physical obstacle is defined as the set of ball states that are "lost." Intuitively, this set, which we call the *workspace obstacle*, $\mathcal{O}_{\mathcal{WS}}$, consists of all ball states on the boundary of the workspace that are never coming back.

We combine these three types into one general physical obstacle,

$$\mathcal{O} = m^{-1}(\mathcal{O}_\mathcal{Q}) \bigcup \mathcal{O}_{T\mathcal{B}} \bigcup \mathcal{O}_{\mathcal{WS}},$$

occupying the ball's configuration space.

These physical obstacles must be "dilated" in the (discrete-time) domain of f_Φ to account for physical collisions that may occur during the continuous-time interval between apex events. We proceed as follows.

For any ball state, Tb , denote by ${\tau_\mathcal{O}}^+(Tb)$ the time to reach the physical obstacle,

$${\tau_\mathcal{O}}^+(Tb) = \min\ t > 0 : F^t(Tb) \in T\mathcal{O},$$

and by ${\tau_\mathcal{O}}^-(Tb)$ the time elapsed from the last physical obstacle,

$${\tau_\mathcal{O}}^-(Tb) = \max\ t < 0 : F^t(Tb) \in T\mathcal{O}.$$

Similarly, denote the times to contact with the robot by

$${\tau_\Phi}^+(Tb) = \min\ t > 0 : M \circ F^t(Tb) \in T\mathcal{C},$$

and

$${\tau_\Phi}^-(Tb) = \min\ t < 0 : \exists Tx \in T\mathcal{C} : F^t \circ Imp(Tx) = Tb,$$

where *Imp* is the impact map.

A discrete obstacle is any apex state whose future continuous-time trajectory must intersect the obstacle, or whose past continuous-time trajectory may have intersected the obstacle,

$$\bar{\mathcal{O}} := \{Tb \in T\mathcal{H} \times \mathcal{V} : \tau^+_\mathcal{O}(Tb) < \tau^+_\Psi(Tb) \bigvee \tau^-_\mathcal{O}(Tb) > \tau^-_\phi(Tb)\}.$$

We will return to obstacles in the discussion on Safety, in Section 3.2.

2.5. Example: The Juggle Φ_J

The generic juggling behavior developed by Rizzi for the Bühgler is induced by a family of control laws in which a single ball is struck repeatedly by the robot paddle in such a way as to direct it toward a limit cycle characterized by a single zero-velocity apex point, or *set point*, $\mathcal{G} \subset \mathcal{H}$ (we ignore the vertical component here). We refer to members of this generic class of controllers as Φ_J. Under the assumptions made in Section 2.3, Φ_J reduces to its associated mirror law, m_J, which is detailed elsewhere (Rizzi and Koditschek 1993; Rizzi 1994) and described below.

2.5.1. The Mirror Law, m_J

For a given ball state, Tb, we begin by inverting the kinematics of the arm to calculate a robot position that would just touch the ball: $q(b) = (\phi_b, \theta_b, 0)$ (recall eq. (5)). We choose $\psi = 0$ to eliminate redundancy. This effectively gives us the ball in joint-space coordinates.

Next, we express informally a robot strategy that causes the paddle to respond to the motions of the ball in four ways. Here, ϕ_r, θ_r, and ψ_r refer to the commanded joint values, whereas ϕ_b and θ_b refer to the values necessary for the paddle to touch the ball:

1. $\phi_r = \phi_b$ causes the paddle to track under the ball at all times.
2. θ_r mirrors the vertical motion of the ball (as it evolves in θ_b): as the ball goes up, the paddle goes down, and vice-versa, meeting at zero height. Differences between the desired and actual total ball energy lead to changes in paddle velocity at impact.
3. Radial motion or offset of the ball causes changes in θ_r, resulting in a slight adjustment of the normal at impact, tending to push the ball back toward the set point.
4. Angular motion or offset of the ball causes changes in ψ_r, again adjusting the normal so as to correct for lateral position errors.

For the exact formulation of m_J, the reader is referred to Rizzi's earlier work (Rizzi 1994).

The mirror law has three parameters for each of the three cylindrical coordinates (angular, radial, and vertical): a set point, and two PD gains. In Section 2.6, we will set the vertical gains to zero to create qualitatively different behavior (palming), while in Section 4 we will generate a family of different juggling controllers by changing only the angular set point.

2.5.2. Analysis, Simulation, and Empirical Exploration of the Domain of Attraction, $\mathcal{D}(\Phi_J)$

The mirror law used in the experiments discussed in this paper does not permit a closed-form expression for the apex-apex return map. Recently we have made strides toward developing a mirror law that has a solvable return map (Rizzi and Koditschek 1994), yet still generates similar behavior. As is seen in the sequel, this would be very helpful to our methods, but the absence of closed-form expressions does not impede the ideas introduced in Section 1.3.

We define the *domain of attraction* of Φ to $\mathcal{G}$ by $\mathcal{D}(\Phi) = \{Tb \in T\mathcal{B} : \lim_{n\to\infty} f_\Phi^n(Tb) = \mathcal{G}\}$ in the absence of obstacles. For the choice of mirror-law parameters used in this paper, $\mathcal{D}(\Phi)$ contains nearly all of $T\mathcal{B}$ such that the ball is above the paddle. In Section 3.2, we will constrain this definition by incorporating all the obstacles from Section 2.4.

Because of the complexity of the system, it is difficult to ascertain the shape of the boundaries of $\mathcal{D}(\Phi_J)$ through systematic juggling experiments, especially with the presence of obstacles. Nevertheless, in Figure 8, we display experimental data used to formulate an approximation of what we call in Section 3.2 the *safe domain* for a juggling controller operating within the bounded workspace formed from the robot's paddle and visible region—the region of Figure 5b without the beam obstacle (Burridge, Rizzi, and Koditschek 1997). The particular choice of mirror-law parameters for this controller comes from previous empirical work. In these plots, the axes Rho and Phi refer to the horizontal polar coordinates of the ball relative to the robot's base.

To overcome this difficulty, and to speed up deployment, we created a numerical simulation of the juggler. In Figure 9, we display the same data as in Figure 8, but run through the numerical simulation of the robot operating under Φ_J rather than the real robot. Since all but one of the starting conditions that failed on the real robot failed in simulation, it is clear that the simulation provides a conservative approximation to the robot's true actions. This simulation is used extensively in Section 4 for finding conservative, invariant ellipsoids to approximate $\mathcal{D}(\Phi_J)$.

2.6. The Complete Palette

In Section 4, we deploy a selection of different controllers from the palette represented by changes in the mirror-law parameters. We start with a set of parameters that gives rise to a robust juggling controller, Φ_{J_0}. Next, we take advantage of the rotational symmetry of m_J, changing only the value for the angular component of the set point, $\bar{\phi}_b$, while rotating a numerically derived domain of attraction to match.

In addition to varying the set point for Φ_J, we use two other controllers based on the mirror law, but with choices of parameters that lead to qualitatively different behaviors.

2.6.1. Palming: Φ_P

If we set to zero the gains regulating the vertical energy of the ball, then we find that the ball will lose energy with each impact, finally coming to rest on the paddle. The lateral terms, however, will continue to stabilize the ball at the horizontal set point, $\mathcal{G} \in \mathcal{H}$. The closed-loop dynamics of this new behavior are significantly different from juggling, as the paddle is in continuous contact with the ball, exerting continuous control over it. We refer to this behavior generically as *palming*: Φ_P. The continuous interaction between ball and paddle allows us to tune the lateral PD gains higher for Φ_P than for Φ_J.

As our numerical simulator relies on the intermittent nature of contact in juggling, we cannot use it to simulate palming. Currently, we have neither an analytically nor a numerically derived domain of attraction for f_{Φ_P}, but empirical results have shown that the projection of the domain into $\mathcal{H}$ is comparable in size to that of f_{Φ_J}, if not larger. We use the same horizontal domain for palming as for juggling, but we only allow Φ_P to be active when the ball has very low vertical energy, as suggested by empirical tests.

2.6.2. Catching: Φ_C

The assumption that we may ignore the vertical subsystem becomes invalid for very low set points, and we find that Φ_J behaves poorly when the ball is nearly in continuous contact with the paddle. In these situations, the V–O subsystem may not notice small bounces quickly enough, and is very susceptible to noise. On the other hand, Φ_P has PD gains tuned

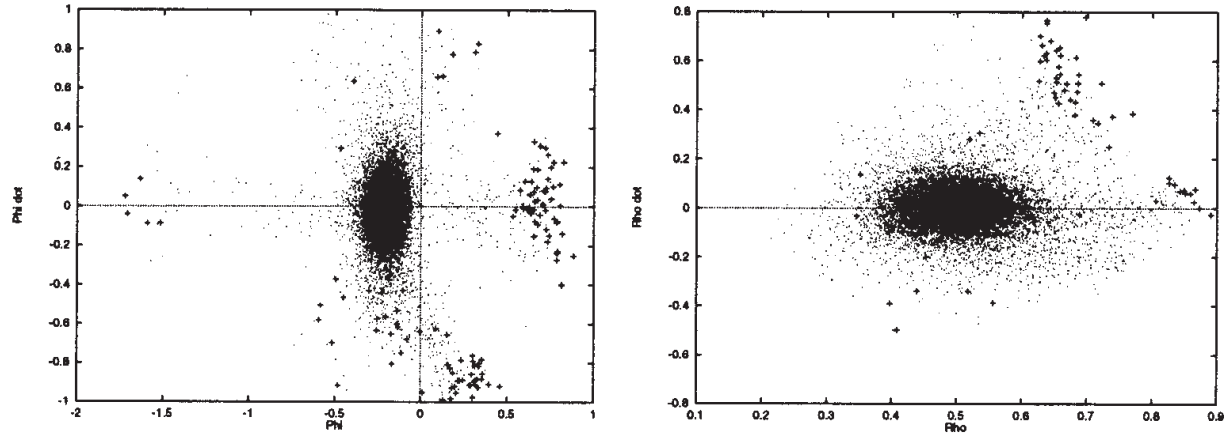

Fig. 8. Empirical data used to estimate the juggling domain, $\mathcal{D}(\Phi_J)$. Each dot (+ sign) represents in apex coordinates a ball trajectory that was successfully (unsuccessfully) handled under the action of Φ_J. Because of the properties of the vertical subsystem, most of these points are at nearly the same height, so only the horizontal coordinates are plotted.

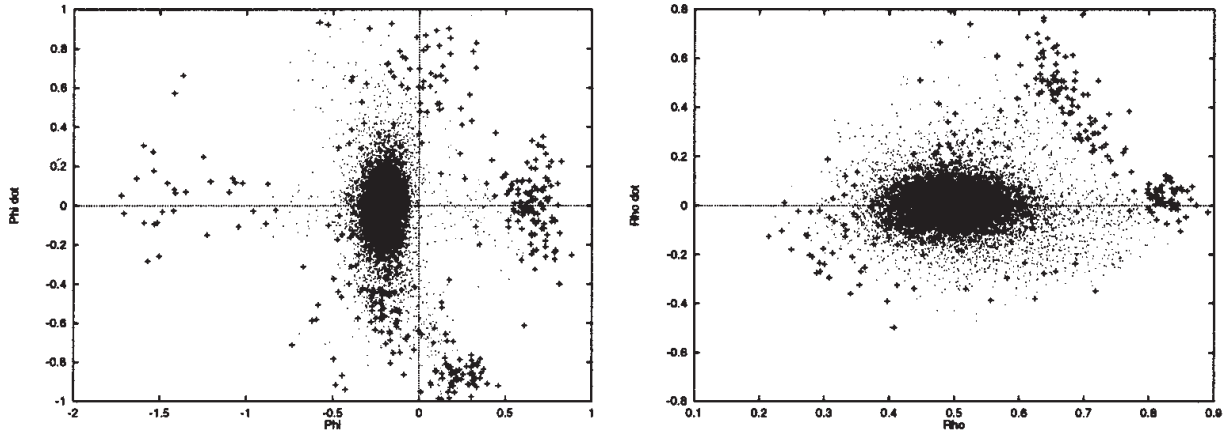

Fig. 9. Starting with the same set of data points as in Figure 8, we used the numerical simulator of Φ_J to predict whether the states would be successfully (dots) or unsuccessfully (+ signs) handled.

too aggressively for such bounces of the ball, and should be switched on only when the ball is essentially resting on the paddle. To guarantee a smooth transition from juggling to palming, we introduce a *catching* controller, Φ_C, designed to drain the vertical energy from the ball and bring it down close to rest on the paddle. Currently, a variant of Φ_J is used in which the vertical set point is very low, and the robot meets the ball higher than normal for Φ_J. For an intuitive view of how the domains for Φ_J, Φ_C, and Φ_P overlap, see Figure 15.

Owing to the mixed nature of this controller, we have neither analytical nor numerical approximations to its domain of attraction. Instead, we use a very conservative estimate based on the numerically derived $\mathcal{D}(\Phi_{J_0})$ of Section 4.1. Although this controller is not ideal, it removes enough of the vertical energy from the ball that we can safely turn on palming. Currently, catching is the dominant source of task failures, as we report in Section 4.4.

3. The Formal Idea: Safe Sequential Composition

The technique proposed here is a variant of the preimage backchaining idea introduced by Lozano-Pérez, Mason, and Taylor (1984), although their algorithm was used for compliant motion in quasi-static environments. We substitute sensory events for the physical transitions that characterized their control sequences.

3.1. Sequential Composition

Say that controller Φ_1 *prepares* controller Φ_2 (denoted $\Phi_1 \succeq \Phi_2$) if the goal of the first lies within the domain of the second: $\mathcal{G}(\Phi_1) \subset \mathcal{D}(\Phi_2)$. For any set of controllers, $\mathcal{U} = \{\Phi_1, ..., \Phi_N\}$, this relation induces a directed (generally cyclic) graph, denoted Γ.

Assume that the overall task goal, $\mathcal{G}$, coincides with the goal of at least one controller, which we call Φ_1: $\mathcal{G}(\Phi_1) = \mathcal{G}$.[10] Starting at the node of Γ corresponding to Φ_1, we traverse the graph breadth-first, keeping only those arcs that point back in the direction of Φ_1, thus ending up with an acyclic subgraph, $\Gamma' \subset \Gamma$. The construction of Γ' induces a partial ordering of $\mathcal{U}$, but the algorithm below actually induces a total ordering, with one controller being handled during each iteration of the loop.

In the following algorithm, we recursively process controllers in $\mathcal{U}$ in a breadth-first manner leading away from Φ_1. We build $\mathcal{U}'(N) \subset \mathcal{U}$, the set of all controllers that have been processed so far, and its domain, $\mathcal{D}_N(\mathcal{U}')$, which is the union of the domains of its elements. As we process each controller, Φ_i, we define a subset of its domain, $\mathcal{C}(\Phi_i) \subset \mathcal{D}(\Phi_i)$, that is the cell of the partition of state space within which Φ_i should be active.

1. Let the queue contain Φ_1. Let $\mathcal{C}(\Phi_1) = \mathcal{D}(\Phi_1)$, $N = 1$, $\mathcal{U}'(1) = \{\Phi_1\}$, and $\mathcal{D}_1(\mathcal{U}') = \mathcal{D}(\Phi_1)$.
2. Remove the first element of the queue, and append the list of all controllers which *prepare* it to the back of the list.
3. While the first element of the queue has a previously defined cell, $\mathcal{C}$, remove the first element without further processing.
4. For Φ_j, the first unprocessed element on the queue, let $\mathcal{C}(\Phi_j) = \mathcal{D}(\Phi_j) - \mathcal{D}_N(\mathcal{U}')$. Let $\mathcal{U}'(N+1) = \mathcal{U}' \cup \{\Phi_j\}$, and $\mathcal{D}_{N+1}(\mathcal{U}') = \mathcal{D}_N(\mathcal{U}') \cup \mathcal{D}(\Phi_j)$. Increment N.
5. Repeat steps 2, 3, and 4 until the queue is empty.

At the end of this process, we have a region of state space, $\mathcal{D}_M(\mathcal{U}')$, partitioned into M cells, $\mathcal{C}(\Phi_j)$. When the state is within $\mathcal{C}(\Phi_j)$, controller Φ_j is active. The set $\mathcal{U}'(M) \subset \mathcal{U}$ contains all the controllers that can be useful in driving the state to the task goal. Note that the process outlined above is only used to impose a total order on the members of $\mathcal{U}'(M)$—the actual cells $\mathcal{C}(\Phi_j)$ need never be computed in practice. This is because the automaton can simply test the domains of the controllers in order of priority. As soon as a domain is found that contains the current ball state, the associated controller is activated, being the highest-priority valid controller.

The combination of the switching automaton and the local controllers leads to the creation of a new controller, Φ, whose domain is the union of all the domains of the elements of $\mathcal{U}'$ (i.e., $\mathcal{D}(\Phi) = \bigcup_{\Phi_i \in \mathcal{U}'} \mathcal{D}(\Phi_i)$), and whose goal is the task goal. The process of constructing a composite controller from a set of simpler controllers is denoted by $\Phi = \bigvee \mathcal{U}'$, and is depicted in Figure 10. In that figure, the solid part of each funnel represents the partition cell, $\mathcal{C}$, associated with that controller, while the dashed parts indicate regions already handled by higher-priority funnels.

3.2. Safety

It is not enough for the robot simply to reach the ball and hit it—the impact must produce a "good" ball trajectory that doesn't enter the obstacle, and is in some sense closer to the goal. Intuitively, we need to extend the obstacle to include all those ball states which induce a sequence of impacts that eventually leads to the obstacle, regardless of how many intervening impacts there may be. We do this by restricting the definition of the domain of attraction.

We define $\mathcal{D}_{\mathcal{O}}(\Phi)$, the *safe domain of attraction*, for controller Φ, given obstacle $\bar{\mathcal{O}}$, to be the largest (with respect to inclusion) positive-invariant subset of $\mathcal{D}(\Phi) - \bar{\mathcal{O}}$ under the map $f_\Phi(\cdot)$.

10. If the goal set of more than one controller coincides with $\mathcal{G}$, then we choose one arbitrarily (e.g., the one with the largest domain), and note that all of the others will be added to the queue in the first iteration of the algorithm.

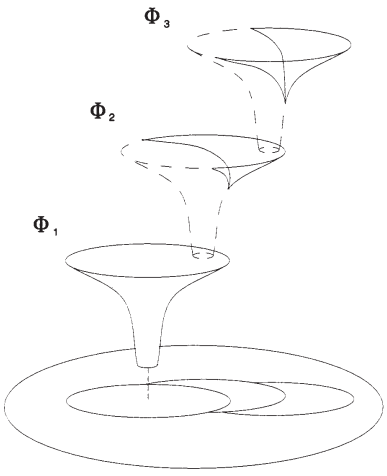

Fig. 10. The sequential composition of controllers. Each controller is only active in the part of its domain that is not already covered by those nearer the goal. Here, $\Phi_3 \succeq \Phi_2 \succeq \Phi_1$, and Φ_1 is the goal controller.

3.2.1. Safe Deployments

If all of the domains of the controllers Φ_j in a deployment are safe with respect to an obstacle, then it follows that the resulting composite controller, $\Phi = \bigvee \Phi_j$, *must* also be safe with respect to the obstacle. This is because any state within $\mathcal{D}_(\Phi)$ is by definition within $\mathcal{D}_{\mathcal{O}}(\Phi_j)$ for whichever Φ_j is active, and the state will not be driven into the obstacle from within $\mathcal{D}_{\mathcal{O}}(\Phi_j)$ under the influence of f_{Φ_j}.

4. Implementation: Deploying the Palette

4.1. Parameterization of Domains

While f_Φ is not available in closed form, its local behavior at goal $\mathcal{G}$ is governed by its Jacobian, Df_Φ, which might be computed using calculus and the implicit function theorem for simple $\mathcal{G}$. However, in practice (Buehler, Koditschek, and Kindlmann 1990b, 1994) we have found that the Lyapunov functions that arise from Df_Φ yield a disappointingly small domain around $\mathcal{G}$ relative to the empirical reality, and we wish to find a much larger, physically representative domain. Thus, we resort to experimental and numerical methods to find a large and usable invariant domain.

We created a typical juggling controller, Φ_{J_0}, by using an empirically successful set of parameters, with goal point $\mathcal{G}_0 = (\bar{\rho}_0, \bar{\phi}_0, \bar{\eta}_0) = (0.6, 0.0, 4.5)$. Using the numerical simulator introduced in Section 2.5, we then tested a large number of initial conditions, $(h_i, \dot{h}_i) = (x_i, y_i, \dot{x}_i, \dot{y}_i)$, covering relevant ranges of $T\mathcal{H}$ (the starting height, v_i, is the same for all runs, as we are not concerned here with the vertical subsystem, due to the decoupling of $\mathcal{H}$ and $\mathcal{V}$ discussed in Section 2.5). In Figure 11, we show samples of the "velocity spines" of these tests. In each plot, one initial velocity, $\dot{h}_i$, is chosen, and a range of positions is mapped forward under iterates of f_Φ. The shaded areas show the location of the first impact of all states that are successfully brought to $\mathcal{G}_0$ (denoted "$*$") without leaving the workspace serviced by the paddle. In the left-hand column, $\dot{y}_i = 0$, while $\dot{x}_i$ varies from large negative to large positive. On the right side, the roles of x and y are reversed. These plots demonstrate the rather complicated shape of the domain of attraction, as we have tried to suggest intuitively in Figure 2.

4.1.1. $\mathcal{D}(\Phi_{J_0})$: The Domain of Φ_{J_0}

Currently, we have no convenient means of parameterizing the true domain of Φ_{J_0}, pictured in Figure 11. Yet to build the sequential compositions suggested in this paper, we require just such a parameterization of domains. Thus, we are led to find a smaller domain, $\mathcal{D}_0$, whose shape is more readily represented, as introduced in Figure 1, and which is invariant with respect to f_{Φ_J} (i.e., $f_{\Phi_J}(\mathcal{D}_0) \subset \mathcal{D}_0$). Ellipsoids make an obvious candidate shape, and have added appeal from the analytical viewpoint. However, the ellipsoids we find are considerably larger than those we have gotten in the past from Lyapunov functions derived from Df_{Φ_J}. Inclusion within an ellipsoid is also an easy test computationally—an important consideration when many such tests must be done every update (currently 330 Hz).

To find an invariant ellipsoid, start with a candidate ellipsoid, $\mathcal{N}_{test} \subset T\mathcal{H}$, centered at $\mathcal{G}_0$, and simulate one iteration of the return map, f_Φ, to get the forward image of the ellipsoid, $f_{\Phi_J}(\mathcal{N}_{test})$. If $f_{\Phi_J}(\mathcal{N}_{test}) \subset \mathcal{N}_{test}$ then $\mathcal{N}_{test}$ is invariant under f_{Φ_J}. Otherwise, adjust the parameters of the $\mathcal{N}_{test}$ to arrive at $\mathcal{N}'_{test}$, and repeat the process. In Figure 12, we display four slices through the surface of such an invariant ellipsoid found after many trials (smooth ellipses), along with the same slices of its single-iteration forward image (dots). Although these pictures do not in themselves suffice to guarantee invariance, they provide a useful visualization of the domain and its forward image, and comprise the primary tool used in the search.[11] The invariant ellipsoid depicted in Figure 12 will be denoted $\mathcal{D}_0$ throughout the sequel.

Relying on the rotational symmetry of the mirror law, we now use Φ_{J_0} to create a family of controllers by varying only the angular component of the goal point, $\bar{\phi}$, keeping all other

11. Since $\mathcal{G}_0$ is an attractor, the existence of open forward-invariant neighborhoods is guaranteed. However, "tuning the shape" of $\mathcal{N}_{test}$ by hand can be an arduous process. This process will be sped up enormously by access to an analytically tractable return map. Even without analytical results, the search for invariance could be automated by recourse to positive-definite programming (a convex nonlinear programming problem) on the parameters (in this case, the entries of a positive-definite matrix) defining the shape of the invariant set. We are presently exploring this approach to generation of $\mathcal{N}_{test}$.

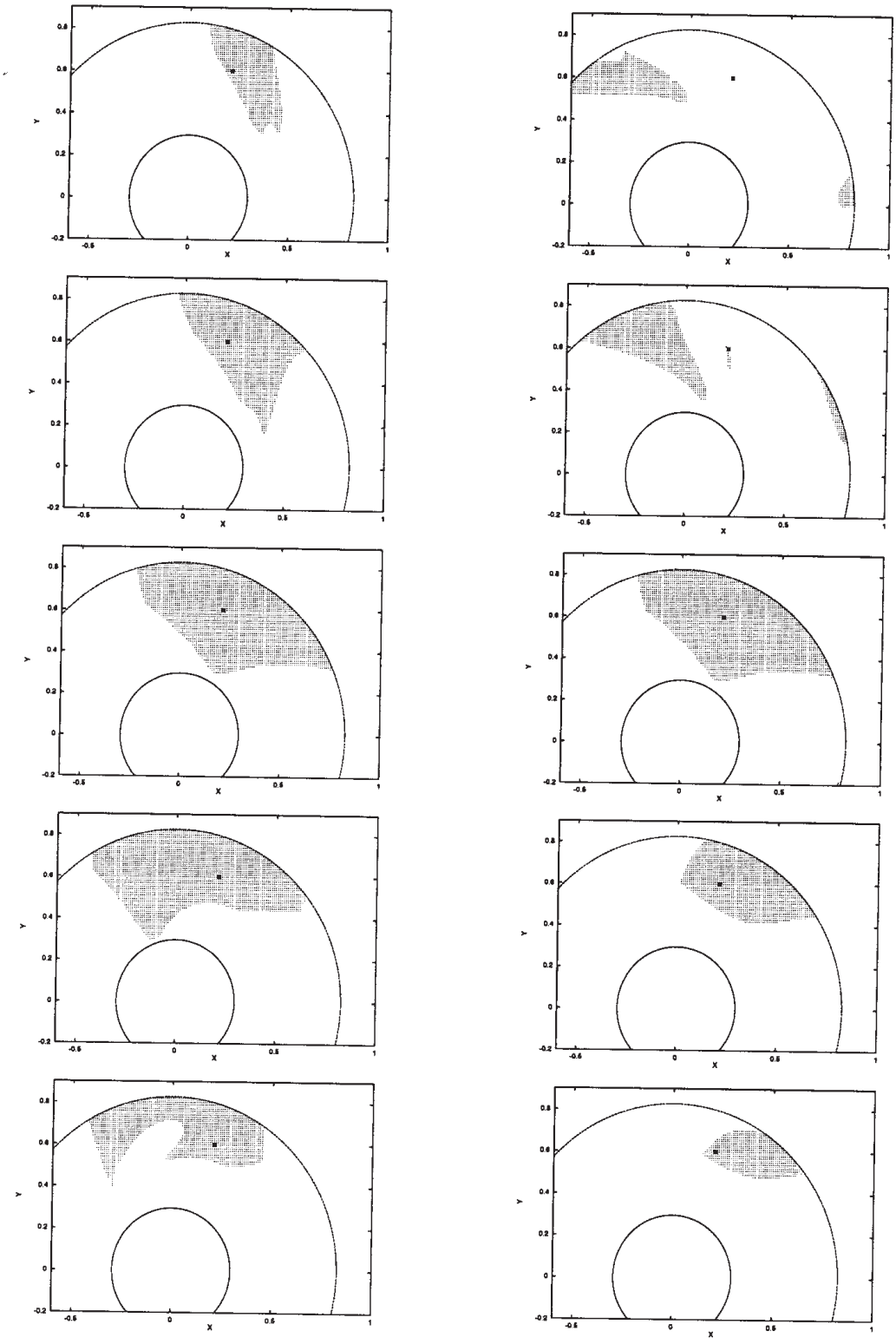

Fig. 11. Shaded regions denote initial conditions that were successfully contained in the workspace while being brought to the goal via iteration of f_{Φ_J}: varying initial $\dot{x}_i$ from negative (top) to positive (bottom) with $\dot{y}_i = 0$ (left column); varying initial $\dot{y}_i$ from negative to positive with $\dot{x}_i = 0$ (right column). The scale for all plots is meters.

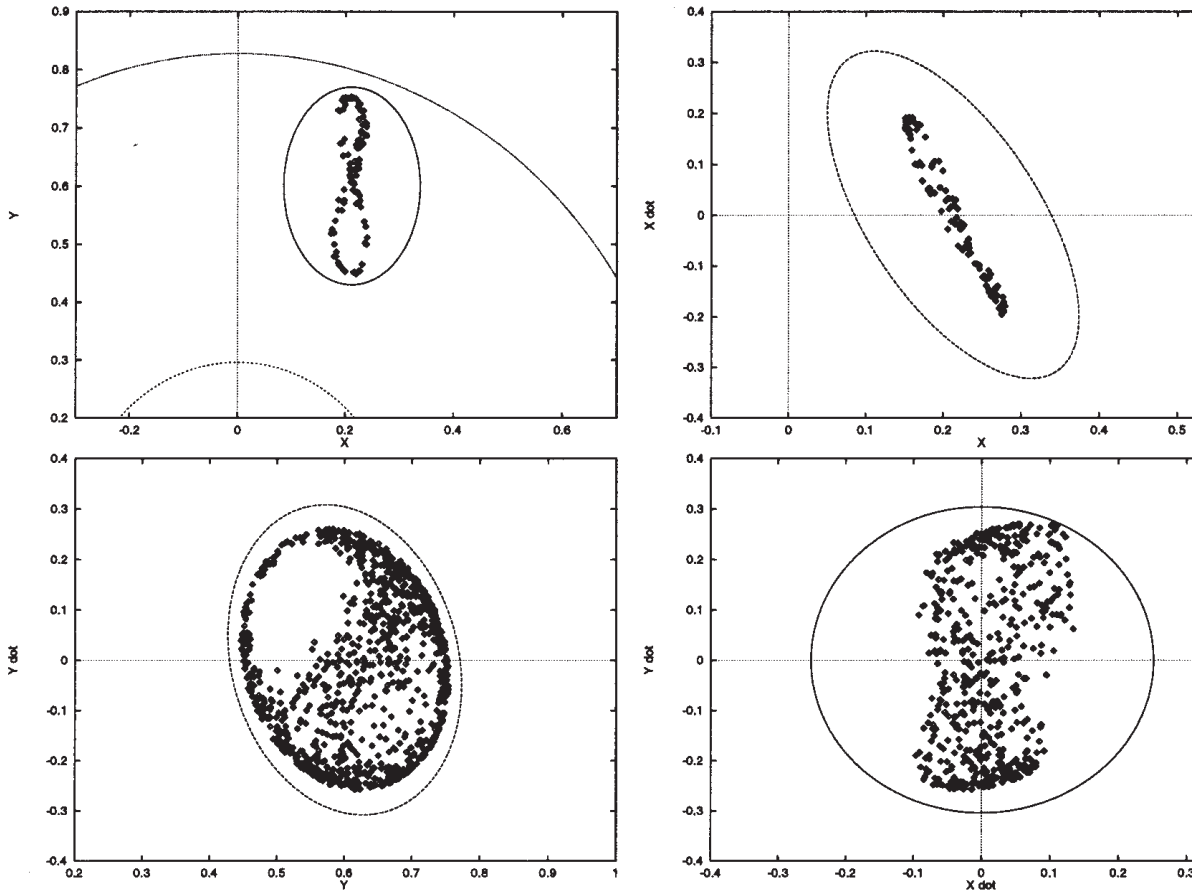

Fig. 12. Our 2-D slices of the invariant ellipsoid, $\mathcal{D}_0 \subset \mathcal{H}$ (smooth curves), and its one-bounce forward image, $f_\Phi(\mathcal{D}_0)$ (dots).

parameters fixed, and rotating $\mathcal{D}_0$ to match. We denote a member of this family by $\Phi_{J_0}(\bar{\phi})$, with goal $\mathcal{G}_0(\bar{\phi})$ and domain $\mathcal{D}_0(\bar{\phi})$.

4.1.2. $\mathcal{D}_{\text{paddle}}(\Phi_{J_0})$: Safety with Respect to the Workspace Boundaries

The largest safe domain with respect to the paddle's workspace is indicated by Figure 11. However, since there is no obvious way of parameterizing this set (recall that geometrically close approximations to this volume in $T\mathcal{H}$ will typically fail to be invariant with respect to f_{Φ_J}), we resort to the more conservative approximation, $\mathcal{D}_0$, shown in Figure 12.

4.1.3. $\mathcal{D}_{\text{beam}}(\Phi_{J_0})$: Safety with Respect to the Beam

In Figure 13a, we depict the simulated iterates of f_{Φ_J} again, but add the beam to the workspace. The zero-velocity slice of the safe domain, $\mathcal{D}_0$, has also been added for comparison (the ellipse with the dotted boundary). The gray area represents those initial states with zero initial horizontal velocity that are successfully brought to $\mathcal{G}_0$ without leaving the workspace or hitting the beam (compare to the middle pictures of Fig. 11). The dark region represents all states that hit the beam, either initially, or after a few bounces. Note that all states in this figure that don't hit the beam initially are successfully brought to the goal. This happens because $\mathcal{G}_0$ is so strongly attracting that states on the wrong side of the beam are simply knocked over it, whereas those already near the goal never stray far enough from it to reach the beam. In Figure 13b, we add horizontal velocity to our initial conditions, and note that secondary images of the beam appear, demonstrating the added complexity in the shape of the largest safe domain arising from the introduction of the beam.

Any $\Phi_{J_0}(\bar{\phi})$ whose domain does not intersect the beam or its preimages, as shown in Figure 13, will be safe with respect to the beam, as well as the paddle's workspace.

Because the beam divides the horizontal workspace into disjoint regions, we need to take advantage of the discrete

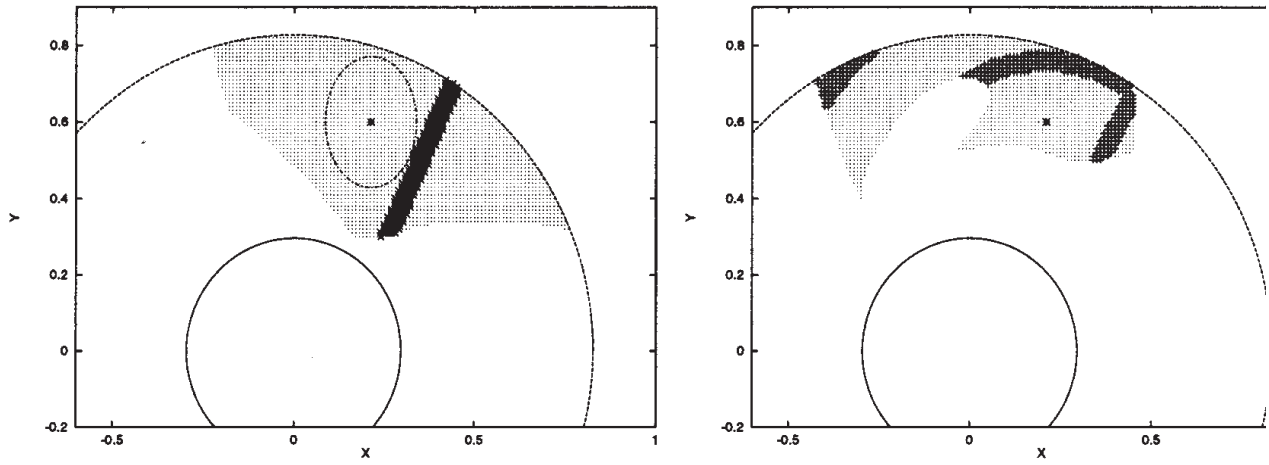

Fig. 13. The safe domain for Φ_{J_0} with the beam inserted. Light-gray areas represent successful initial states, while darker areas show states that eventually hit the beam. Zero initial velocity is shown (a), and the appropriate slice of $\mathcal{D}_0$ is added for comparison. For $\dot{x}_i = 1.5\frac{m}{s}$ (b), preimages of the beam are evident.

dynamics of f_{Φ_J} to cross it (recall from Section 2.3 that this is a return map). We do this by finding a disjoint, yet still invariant, safe domain for Φ_{J_0}. We generate a series of small ellipsoids starting on the far side of the beam, such that each maps forward into the next, and the last maps forward entirely into $\mathcal{D}_0$. We call this new disconnected, forward-invariant domain $\mathcal{D}_1$ throughout the sequel.

In principle, finding $\mathcal{D}_1$ is not much more difficult than finding $\mathcal{D}_0$. First we choose another controller close to the beam on the far side, $\Phi_{J_0}(\bar{\phi}_{pre})$, and take a small neighborhood, $\mathcal{N}_1$, surrounding its goal, $\mathcal{G}_0(\bar{\phi}_{pre})$. This is the *launch window*. Next, we take the forward image of $\mathcal{N}_1$ under one iteration of the discrete return map, and find an ellipsoid, $\mathcal{N}_2$, that completely contains it. We then map $\mathcal{N}_2$ forward, and continue this process until we find an ellipsoid, $\mathcal{N}_n$, whose forward image is completely contained in $\mathcal{D}_0$. By construction, the union of ellipsoids, $\mathcal{D}_1 := \mathcal{D}_0 \bigcup \mathcal{N}_1 \bigcup \cdots \bigcup \mathcal{N}_n$, is invariant under f_{Φ_J}, and safe with respect to the beam and the workspace boundaries.

In practice, differences between the true robot behavior and the simulator have proven too substantial for the small neighborhoods, $\mathcal{N}_i$, to effectively contain the trajectory of the ball during the activation of the trans-beam controller. Instead, we have been forced to build our $\mathcal{D}_1$ domain via direct experimentation. We inspect the actual experimental distribution of apex points (that should have been in $\mathcal{N}_2$, according to the simulation), and create an empirically derived $\mathcal{N}'_2$. We then repeat this process for $\mathcal{N}_3$, arriving at an empirically derived $\mathcal{N}'_3$, and then again to create $\mathcal{N}'_4$. For the present case, this turns out to be completely contained in $\mathcal{D}_0$. The empirically derived trans-beam domain, $\mathcal{D}_1$, used in the experiments below has four parts: $\mathcal{N}_1$, $\mathcal{N}'_2$, $\mathcal{N}'_3$, and $\mathcal{D}_0$, and is shown in Figure 14, projected onto four orthogonal planes in $T\mathcal{H}$.

4.2. Composition of Domains

The robot's task is to capture and contain a ball thrown into the workspace on one side of the beam, negotiate it over the beam without hitting it, and bring it to rest on the paddle at location $\mathcal{G}_T = (0, \bar{\rho}_0, \bar{\phi}_T)$. We now describe the deployment created by hand, using the controllers constructed in the previous sections to achieve this task.

4.2.1. Deployment

The only controllers in our palette capable of stabilizing the dynamics about $\mathcal{G}_T$ are the stable palming controllers with goal point $\mathcal{G}_T$, so we choose one with empirically successful parameters: $\Phi_P(\mathcal{G}_T)$. The palming domain is a large ellipsoid in $T\mathcal{H}$, but its projection onto $\mathcal{V}$ is much smaller, thus we only allow palming for states with very low vertical energy. We also use a catching controller, Φ_C, that has a small domain at $\mathcal{G}_T$, and that successfully reduces the ball's vertical energy from typical juggling values to values near zero (i.e., palming values). This introduces the need for $\Phi_{J_0}(\bar{\phi}_T)$—a juggler that attracts juggled balls to $\mathcal{G}_T$, but with nonzero vertical energy. Note that by our construction, $\Phi_{J_0}(\bar{\phi}_T) \succeq \Phi_C(\mathcal{G}_T) \succeq \Phi_P(\mathcal{G}_T)$, but $\Phi_{J_0}(\bar{\phi}_T) \not\succeq \Phi_P(\mathcal{G}_T)$, so the system must go through catching from juggling to reach palming. This idea is depicted intuitively in Figure 15: in the left-hand figure, Φ_J draws a flat circular region of high-energy states toward

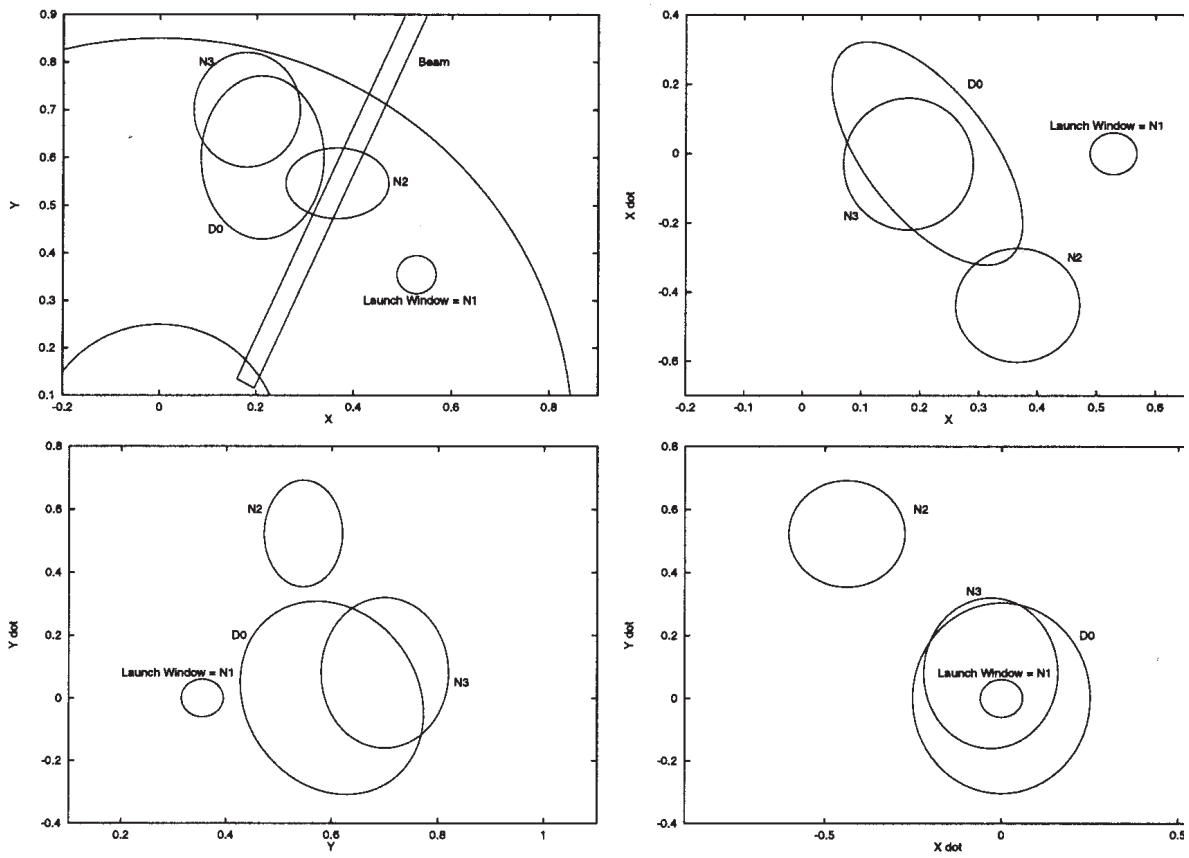

Fig. 14. Four projections of the union of the disjoint ellipsoids, $\mathcal{D}_1 := \mathcal{D}_0 \bigcup \mathcal{N}_1 \bigcup \mathcal{N}_2 \bigcup \mathcal{N}_3$, created by analysis of empirical data. Notice the velocity components of $\mathcal{N}_2$, which consists of apex points over the beam.

its goal, $\mathcal{G}_J$, which lies within $\mathcal{D}_C$. In the right-hand figure, Φ_C draws its (ellipsoidal) domain down in vertical energy to its goal, which lies within $\mathcal{D}_P$, a flat circular region of low energy.

Starting with $\Phi_{J_0}(\bar{\phi}_T)$, we add new jugglers with new values of $\bar{\phi}$, such that each new goal lies just within the last domain (which was a rotated copy of $\mathcal{D}_0$), repeating this process until we reach $\bar{\phi}_0$, and the beam blocks further progress. At this point, we use $\mathcal{D}_1$, the trans-beam domain, for $\Phi_{J_0}(\bar{\phi}_0)$, as described in Section 4.1.3.

Finally, we create another sequence of rotationally symmetric controllers on the far side of the beam, starting with one whose goal lies at $\mathcal{G}_0(\bar{\phi}_{pre})$, and continuing until the edge of the visible workspace blocks further progress. The entire palette of controllers is shown in Figure 16. Note that all domains are simply rotated copies of $\mathcal{D}_0$ except for the four ellipsoids surrounding the beam, which make up the invariant domain, $\mathcal{D}_1$, for the trans-beam controller of Figure 14. Some of the interesting parameters of this deployment are listed in Table 1.

Table 1. The Full Deployment, with Controller Types, Goal Points, and Domain Types[a]

Ellipses	Type	Goal: $\bar{\phi}$	Domain Type
1	Φ_P	0.3	$\mathcal{D}_P$
2	Φ_C	0.3	$\mathcal{D}_C$
3	Φ_J	0.3	$\mathcal{D}_0$
4	Φ_J	0.15	$\mathcal{D}_0$
5–8	Φ_J	0.0	$\mathcal{D}_1$
9	Φ_J	-0.64	$\mathcal{D}_0$
10	Φ_J	-0.81	$\mathcal{D}_0$
11	Φ_J	-0.97	$\mathcal{D}_0$
12	Φ_J	-1.12	$\mathcal{D}_0$
13	Φ_J	-1.26	$\mathcal{D}_0$
14	Φ_J	-1.4	$\mathcal{D}_0$

a. All goal points have the same radial component, $\bar{\rho}_0 = 0.6$, so we list here only the angular component, $\bar{\phi}$. Ellipse numbers correspond to those in Figure 16.

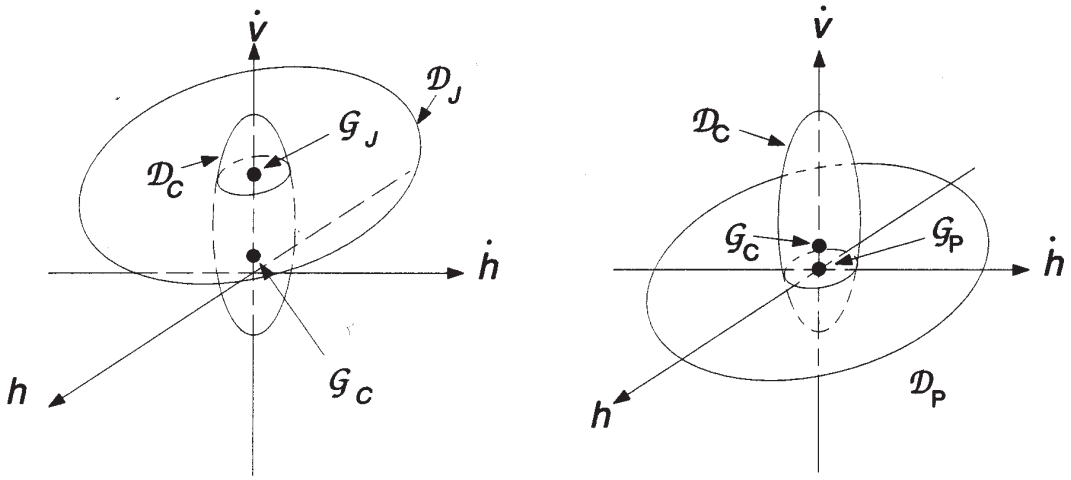

Fig. 15. Juggle to catch to palm: the juggle drives all of its domain to $\mathcal{G}_J$, which lies within the domain for the catch, but with nonzero vertical energy (a); the catch reduces the vertical energy of the ball, bringing it down into the domain for palming (b).

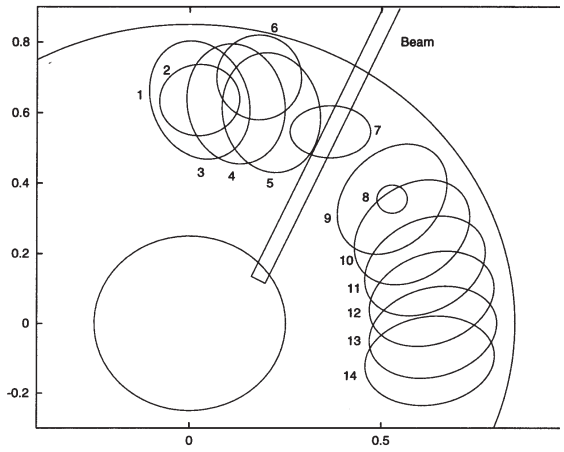

Fig. 16. A theoretically sound deployment using the jump controller of Figure 14.

Every member of this deployment was chosen by hand to ensure that a path exists from all the domains to the task goal state. The algorithm of Section 3.1 leads in a straightforward manner to a linear tree with no branches, thus the ideal trajectory of the system should be a stepwise descent from the first cell of the partition entered through all the intermediate cells, and finally to the catching and then to the palming cell, where it should remain.

4.2.2. A Hierarchy of Controllers

The deployment and controller-switching scheme we have just described relies for its formal properties on the abstracted domain and goal properties of the constituent return maps. In turn, the invariance and attracting properties of f_Φ arise from abstractions whose validity depends on the underlying tracking and convergence properties of the continuous controller and observer. Thus, we have a hierarchy of controllers (and, hence, of abstraction), where each level functions on the basis of certain assumptions about the quality of performance of the lower levels. As we see in Section 4.4.3, however, there will be situations where the transients in the robot's tracking of the reference are too large for it to get to the ball despite the mirror law, thus violating the underlying assumptions at the lower level. Moreover, we see in Section 4.3 that the noise in the system sometimes causes the ball to stray from all the domains of the deployment, despite their theoretical invariance, violating the properties assumed by the higher level.

The entire deployment of Section 4.2.1 may be thought of as generating a single (complex) controller, Φ, with its own goal and domain, which could be used as an element of a still more complex deployment. When this is done, there will be assumptions about the ability of Φ to safely move the ball across the beam and to its goal, and so the process continues.

In summary, a hierarchical system will function only as well as its constituent abstractions can be realized empirically. Since no model is ever perfect, all theoretical guarantees are subject to operating assumptions. It is important to keep in mind the lower-level assumptions being made, so that empirical settings wherein such assumptions fail can be identified.

4.3. Characteristics of the Test Runs

In Figure 16, we depict the entire deployment of the previous section by superimposing the zero-velocity slices of all the various domains onto the horizontal workspace, along with

projections of the nonzero velocity parts of $\mathcal{D}_1$. There are a total of 14 ellipsoids for 11 different controllers; the union of 4 of them is associated with the trans-beam controller's domain, $\mathcal{D}_1$, with the same ellipsoid being used for the final juggling and palming controllers (numbers 1 and 3 in the figure). This is our hand-crafted realization of the concept depicted in Figure 4.

4.3.1. A Typical Run

Figure 17a shows the trace of a typical run, with the active controller plotted against time, while Figure 17b shows the same run projected onto the horizontal workspace, along with the deployment of Figure 16. Typically, the ball was introduced into a cell of the partition that had very low priority, such as ellipses 13 or 14 in Figure 16. The robot brought it fairly quickly, within a few bounces, to the controller closest to the beam on the far side (ellipses 9), where it stayed for some time, waiting for the ball to enter the launch window, $\mathcal{N}_1$ (ellipses 8), of the trans-beam controller's domain, $\mathcal{D}_1$.

At this point the "messiness" of the system mentioned in Section 1.4 is most evident, as the observer (recall Fig. 6) sometimes "hallucinates" that the ball state is within $\mathcal{N}_1$ (ellipses 8), thereby activating the trans-beam controller, and then changes its estimate, which takes the ball state back out of $\mathcal{N}_1$, turning on the prebeam controller (ellipses 9) again (these spurious transitions are labeled "launch attempts" in Fig. 17a). Although the eventual action of the robot at impact (the dots in Fig. 17) is almost always correct, the third degree of freedom, ψ_r, visibly shifts back and forth during the flight of the ball, due to the large difference in goal location between the pre- and trans-beam controllers (ellipses 9 and 8). Eventually, the true ball state enters $\mathcal{N}_1$, the trans-beam controller (ellipses 8) becomes active, and the task proceeds.

In the particular run shown in Figure 17, the ball overshoots controllers 5 and 4, and goes straight from ellipses 6 to ellipses 3. The first catch attempt is unsuccessful, with the ball exiting $\mathcal{D}_C$ (ellipses 2) before entering $\mathcal{D}_P$ (ellipses 1), so the juggle (ellipses 3) turns back on; after a couple of hits, Φ_C turns on again, and the ball is brought down to the paddle successfully.

4.3.2. Descriptive Statistics

In Table 2, we list the percentage of successful trials within each of three sets of experiments, which are described below. We also show for all successful trials the mean and standard deviation of the length of time from introduction to prelaunch mode, time spent there before launch, time from launch to palming, and total time from start to finish. Additionally, we show the total number of regressive steps considered. The "hallucinated" mode switches result from controller transitions based on false observer transients, which change back after the observer settles. All regressive steps actually taken were associated with failed catch attempts, where the ball moved out of $\mathcal{D}_C$ before it was in $\mathcal{D}_P$, and sometimes moved to juggles further away than ellipses 3. The regressive steps, both those "hallucinated" and those taken, remind us of the gulf between the ideal assumption, $T\hat{b} \approx Tb$, and reality, as discussed in Sections 2.1 and 4.4.3.

4.4. Deployment of Invariant Sets

We ran three sets of experiments, each with 60 trials. We started off by carefully introducing the ball to a theoretically correct deployment. Next, we added a set of "catchalls"—controllers with domains not demonstrably safe, yet practically effective—and once again carefully introduced the ball. We also modified $\mathcal{D}_C$ to be more conservative, to reduce the number of failures due to catching, letting a catchall surrounding it save the ball if it strayed out before reaching $\mathcal{D}_P$. Finally, we ran the second deployment again, but this time with wild, almost antagonistic, introductory tosses of the ball. The results of these experiments we now discuss in detail, and summarize in Table 2.

4.4.1. Theoretically Correct Deployment

In our first set of experiments, the deployment was precisely that of Figure 16—an exact implementation of the formal algorithm presented in Section 3.1. The ball was carefully introduced into the workspace 60 times, 49 of which (82%) were successfully negotiated along the workspace, over the beam, and safely brought to rest on the paddle. The ball was knocked out of all safe domains five times while going over the beam, and six times during the transition from catching to palming, which is still being refined.

Some statistics concerning the various segments of a typical run are given in the first column of Table 2. Note that more than half of the total time is spent waiting for the ball to enter the launch window and get knocked over the beam, and differences in the length of this delay account for most of the variance in total task time. This results from the launch window being small relative to the noise in the robot-ball system.

The imperfect success rate for this formally correct control scheme belies the exact accuracy of our models and assumptions. Nevertheless, despite the "messiness" of our physical system, the relatively high success rate validates the utility of our approach in a practical setting.

4.4.2. Adding Safety Nets

Given the inevitable infidelities of the robot and ball with respect to the ideal models of ball flight, impacts, and robot kinematics, a physical implementation will remain roughly faithful to the simulation results, but inevitably depart frequently enough and with significant enough magnitude that some further "safety net" is required to bring balls that start (or unexpectedly stray) outside the domain of the composite back into it.

Using Figure 11 as a guide, we added very-low-priority controllers with larger domains of attraction. These

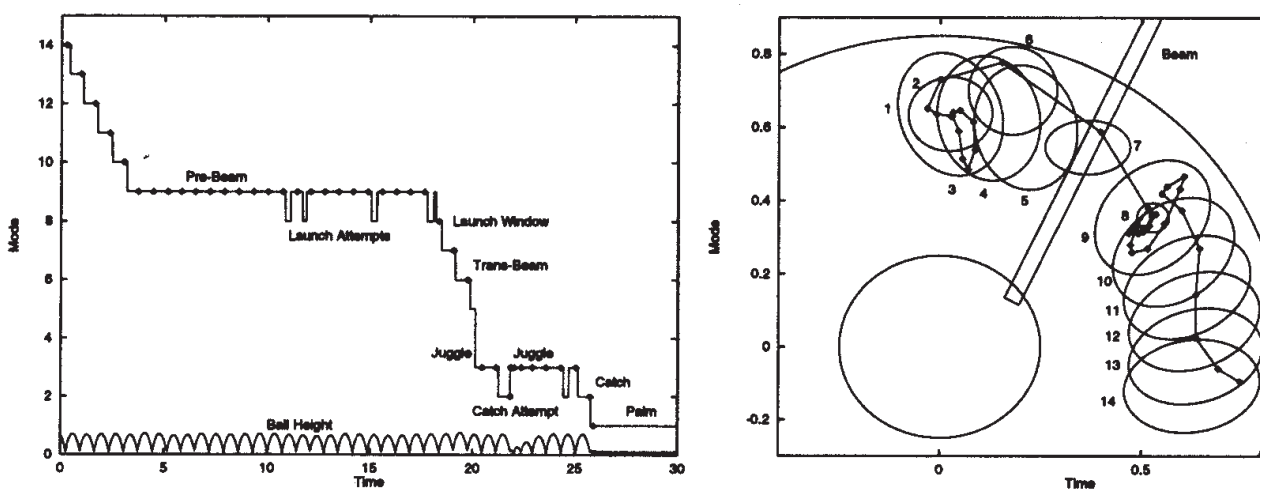

Fig. 17. The cell number or mode—also the ellipse number from Figure 16—plotted against time for a typical run, with a trace of the ball's vertical position added for reference (a). The dots represent the mode at the moment of impact. The horizontal projection of the same trace, superimposed on the deployment of Figure 16 (b). The dots show the apex positions.

Table 2. Statistics Characterizing Three Sets of 60 Trials Each[a]

Statistic	Experiment 1	Experiment 2	Experiment 3
Number of Successful Runs	49 (82%)	57 (95%)	23 (38%)
Time (sec) to Reach			
Prelaunch Mode: Mean	3.954	2.775	1.276
Standard Deviation	2.086	1.283	0.428
Time (sec) Spent in			
Prelaunch Mode: Mean	9.811	8.368	13.671
Standard Deviation	10.020	5.879	8.498
Time (sec) from Launch to			
Palming: Mean	3.871	5.837	4.351
Standard Deviation	1.821	4.456	2.204
Time (sec) from Throw to			
Palming: Mean	17.636	16.980	19.299
Standard Deviation	10.190	7.809	8.300
Regressive Mode Switches:			
"Hallucinated"	98	71	34
Actually Taken	4	0	0
Failed Catch Attempts:			
Recovered	6	52	7
Lost	6	3	1

a. The first three rows give a breakdown for the successful runs of the total time into three segments: throw to prelaunch, time spent in prelaunch, and launch to palming. The fourth row has the same statistics for total time from throw to palm. Regressive mode switches are "hallucinated" if the mode switches back before impact.

controllers have domains that in some cases extend beyond the end of the paddle (but not into the beam), and are not theoretically invariant. Nonetheless, the idea behind our second set of experiments was to use such controllers as backups for the regular deployment, capturing the ball if it strayed away from the controllers of Experiment 1, and channeling it back into the theoretically sound portion of the deployment. In addition, the presence of the catchall controllers allowed us to reduce the size of $\mathcal{D}_C$, thus avoiding some of the failures resulting from bad catches.

The ball was again carefully introduced 60 times. Three failures resulted from bad transitions from catching to palming, but the remaining 57 runs (95%) were successful. The success of these catchall controllers confirms the simulation results of Figure 11, suggesting that there exist larger invariant domains than the conservative $\mathcal{D}_0$, or even $\mathcal{D}_1$.

In the second column of Table 2, we give statistics concerning these test runs. Note that the more aggressive domains bring the ball to the prelaunch mode faster, and actually launch the ball slightly faster. However, the more conservative domain for catching results in longer delays before palming, with many more failed catch attempts, although the catchalls nearly always provided a safe recovery.

4.4.3. Testing the Limits

The final set of tests involved significantly wilder introductions of the ball. Several new controllers were added to handle high lateral velocities of balls thrown into the workspace. Although the base motor is quite powerful, we found that the robot could react quickly enough to contain only those throws landing relatively near its paddle. For those landing further away, the mirror law generated a good reference trajectory, but the robot was not physically able to track it in time (the ball lands—reaches $z = 0$—in roughly 10 camera frames, or 0.17 sec). Of the balls that were thrown well and landed near the paddle, including some with large horizontal velocity, 23 of 25 (92%) were successfully contained and negotiated to the final state. In fact, the robot succeeded 23 of 24 times when its favorable initial configuration enabled it to hit the ball more than twice.

This experiment demonstrates, again, the problems that may arise when the underlying assumptions (in this case, that the robot is successfully tracking the mirror-law reference) are no longer valid, as discussed in Section 4.2.2. Since the mirror law generates reasonable trajectories during the normal course of juggling, it would suffice to ensure that the robot start in a state from which it can handle any ball state within any domain of the deployment. This was not possible within our taxing framework; thus the ball had to be introduced near the paddle for the robot to contain it.

The other breakdown of our assumptions comes when $T\hat{b}$ is not tracking Tb correctly at the moment of impact. This leads to the incorrect choice of partition, and thus control law, leading to undesirable behavior. Since our observer is quite good, this only occurs when the ball meets the paddle soon after leaving it, which only happens during catches. Thus, we see that when the ball is thrown in such a manner as to allow the robot to track the mirror law, it is the failure of the observer to track the ball that is the dominant source of failure.

5. Conclusions

We have described an approach to creating switching control algorithms that provides a theoretical guarantee of the safety of the resultant controller, based on the properties of the constituent controllers. Although the methods proposed in this paper require theoretical constructs that may be difficult or impossible to obtain exactly, we have shown that conservative approximations can be used in their place with very good success. We believe that the developing systems discipline described herein may be extended to build a variety of useful dexterous machines that are similarly single-minded in their pursuit of the user's goal behavior and ability to surmount unanticipated perturbations along the way.

Although the robot is running with a deployment of many controllers, it is difficult for a casual observer to tell that there is more than one controller. The machine's motion seems coordinated, the progress of the ball toward its goal seems reasonably timely (see Table 2), and the response to obstacles, and especially unanticipated disturbances (such as an experimenter deflecting the ball in flight), looks dexterous.

There are several interesting directions of further study that are highlighted by these experiments. Clearly, we need to develop automated methods for finding larger invariant domains than the extremely conservative $\mathcal{D}_0$ found in Section 4.1. This would allow fewer members in the deployment. It would also be useful to extend the model introduced in Section 2 to explicitly include robot states in the design of a deployment of Section 3, avoiding some of the problems mentioned in Section 4.4.3.

The deployment created and used in Section 4 was carefully hand-crafted to ensure that there was a path from the starting region to the goal. We would like to develop automatic methods for creating deployments from a given palette of controllers, such as the rotationally symmetric family of controllers stemming from $\mathcal{D}_0$. If the system were able to automatically go from $\mathcal{G}$ to Φ to $\mathcal{D}$, then it could choose its own deployment by backchaining away from the task goal in a manner analogous to the method used in Section 4.2.1.

We are also exploring ways to move the controller smoothly from starting point to task goal. At any point in time, the local controller's goal would be moving toward the task goal by descending a potential field such as those discussed by Rimon and Koditschek (1992). At any moment in time, there would be trade-offs between the size of the domain and the velocity of the local goal toward the task goal.

Acknowledgments

We are particularly grateful for the helpful and insightful comments of Professors Elmer Gilbert and Mike Wellman. The works of R. R. Burridge and A. A. Rizzi are supported in part by NSF grant IRI-9396167. D. E. Koditschek's work is partially supported by NSF grants IRI-9396167, CDA-9422014, and IRI-9510673.

References

Abraham, R., and Marsden, J. E. 1978. *Foundations of Mechanics*. Reading, MA: Benjamin/Cummings.

Brockett, R. W. 1983. Asymptotic stability and feedback stabilization. In Brockett, R. W., Millman, R. S., and Sussman, H. J. (eds.) *Differential Geometric Control Theory*. Birkhäuser, pp. 181–191.

Buehler, M., Koditschek, D. E., and Kindlmann, P. J. 1990a. A simple juggling robot: Theory and experimentation. In Hayward, V., and Khatib, O. (eds.) *Experimental Robotics I*. New York: Springer-Verlag, pp. 35–73.

Buehler, M., Koditschek, D. E., and Kindlmann, P. J. 1990b. A family of robot control strategies for intermittent dynamical environments. *IEEE Control Sys. Mag.* 10(2):16–22.

Buehler, M., Koditschek, D. E., and Kindlmann, P. J. 1994. Planning and control of a juggling robot. *Intl. J. Robot. Res.* 13(2):101–118.

Burridge, R. R. 1996. Sequential composition of dynamically dexterous robot behaviors. PhD thesis, Department of Electrical Engineering and Computer Science, University of Michigan.

Burridge, R. R., Rizzi, A. A., and Koditschek, D. E. 1995. Toward a dynamical pick and place. *Proc. of the 1995 Intl. Conf. on Intell. Robots and Sys.*, vol. 2. Los Alamitos, CA: IEEE Computer Society Press, pp. 292–297.

Burridge, R. R., Rizzi, A. A., and Koditschek, D. E. 1997. Obstacle avoidance in intermittent dynamical environments. In Khatib, O., and Salisbury, J. K. (eds.) *Experimental Robotics IV*. New York: Springer-Verlag, pp. 43–48.

Erdmann, M. A. 1984. On motion planning with uncertainty. MS thesis and Technical report AI-TR-810, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Guckenheimer, J., and Holmes, P. 1983. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. New York: Springer-Verlag.

Hirsch, M. W. 1976. *Differential Topology*. New York: Springer-Verlag.

Ish-Shalom, J. 1984. The CS language concept: A new approach to robot motion design. *Proc. of the 1984 IEEE Conf. on Decision and Control*. Los Alamitos, CA: IEEE, pp. 760–767.

Koditschek, D. E. 1992. Task encoding: Toward a scientific paradigm for robot planning and control. *J. Robot. Autonomous Sys.* 9:5–39.

Koditschek, D. E., and Buehler, M. 1991. Analysis of a simplified hopping robot. *Intl. J. Robot. Res.* 10(6):587–605.

Latombe, J. 1991. *Robot Motion Planning*. Boston, MA: Kluwer.

Lozano-Pérez, T., Jones, J. L., Mazer, E., and O'Donnell, P. A. 1987. Handey: A robot system that recognizes, plans, and manipulates. *Proc. of the 1987 IEEE Intl. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE, pp. 843–849.

Lozano-Pérez, T., Mason, M. T., and Taylor, R. H. 1984. Automatic synthesis of fine-motion strategies for robots. *Intl. J. Robot. Res.* 3(1):3–23.

Lyons, D. M. 1993. Representing and analyzing action plans as networks of concurrent processes. *IEEE Trans. Robot. Automat.* 9(3):241–256.

Lyons, D. M., and Hendriks, A. J. 1994. Planning by adaption: Experimental results. *Proc. of the 1994 IEEE Intl. Conf. on Robot. and Automat.* Washington, DC: IEEE, pp. 855–860.

Mason, M. T. 1985. The mechanics of manipulation. *Proc. of the 1985 IEEE Intl. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE, pp. 544–548.

Ramadge, P. J., and Wonham, W. M. 1987. Supervisory control of a class of discrete event processes. *SIAM J. Control Optimiz.* 25(1):206–230.

Rimon, E., and Koditschek, D. E. 1992. Exact robot navigation using artificial potential fields. *IEEE Trans. Robot. Automat.* 8(5):501–518.

Rizzi, A. A. 1994. Dexterous robot manipulation. PhD thesis, Yale University.

Rizzi, A. A., and Koditschek, D. E. 1993. Further progress in robot juggling: The spatial two-juggle. *Proc. of the 1993 IEEE Intl. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE, pp. 919–924.

Rizzi, A. A., and Koditschek, D. E. 1994. Further progress in robot juggling: Solvable mirror laws. *Proc. of the 1994 IEEE Intl. Conf. on Robot. and Automat.* Washington, DC: IEEE, pp. 2935–2940.

Rizzi, A. A., and Koditschek, D. E. 1996. An active visual estimator for dexterous manipulation. *IEEE Trans. Roboti. and Automat.* 12(5).

Rizzi, A. A., Whitcomb, L. L., and Koditschek, D. E. 1992. Distributed real-time control of a spatial robot juggler. *IEEE Computer* 25(5).

Tung, C. P., and Kak, A. C. 1994. Integrating sensing, task planning, and execution. *Proc. of the 1994 IEEE Intl. Conf. on Robot. and Automat.* Washington, DC: IEEE, pp. 2030–2037.

Whitcomb, L. L. 1992. Advances in architectures and algorithms for high-performance robot control. PhD thesis, Yale University.

Whitcomb, L. L., Rizzi, A. A., and Koditschek, D. E. 1993. Comparative experiments with a new adaptive controller for robot arms. *IEEE Trans. Robot. Automat.* 9(1):59–70.

Whitney, D. E. 1977. Force-feedback control of manipulator fine motions. *ASME J. Dyn. Sys.* 98:91–97.