Sequential Composition of Dynamically Dexterous Robot Behaviors

R. R. Burridge
Artificial Intelligence Laboratory
EECS Department, College of Engineering
University of Michigan
1101 Beal Ave, Ann Arbor MI 48109-2110
burridge@eecs.umich.edu

A. A. Rizzi
Microdynamic Systems Laboratory
The Robotics Institute
Carnegie Mellon University
Pittsburgh PA 15213-3891
arizzi@bugle.msl.ri.cmu.edu

D. E. Koditschek
Artificial Intelligence Laboratory and Controls Laboratory
EECS Department, College of Engineering
University of Michigan
1101 Beal Ave, Ann Arbor MI 48109-2110
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 freespace formed when the ball and paddle remain in contact, forcing the machine to "let go" for a time in order to bring the ball to the desired state. The controller compositions that 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.

1 Introduction

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. The controllers are designed and then ordered 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 volume of the composite domain of attraction around the goal cell is thus as large as the union of the domains of all constituent controllers.

1.1 Statement of the Problem

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 dynamical interactions between robot and environment such as active balancing, hopping, throwing, catching, juggling, and other tasks requiring active regulation of kinetic energy. At the same time, we wish to explore how such dexterous behaviors can be marshalled toward goals whose achievement requires at least the rudiments of strategy.

We have for some time explored paddle juggling of a ping pong ball as a task domain that epitomizes dynamically dexterous behavior. The dynamics of the environment necessitate constant action from the robot, as the ball will fall to the floor unless repeatedly struck from below or balanced on the paddle surface. Moreover, the intermittent nature of the robot-ball interaction provides a focussed testbed for studying the hybrid (mixed continuous and discrete) control problems that seem to arise in many areas of robotics. In the work presented in this paper, we add a strategic aspect to the robot's task by introducing obstacles. The machine must acquire and contain a thrown ball, maneuver it through the workspace while avoiding obstacles that necessitate re-grasping[1] along the way, and finally bring it to rest on the paddle at a desired location. It seems appropriate to refer to the robot's task as "Dynamical Pick and Place."

[1] In general, re-grasping is required when the free configuration space attendant upon a specified grasp is disconnected, and the goal does not lie in the presently occupied connected component. As will be seen, in the present problem, the configuration space of the "palming" grasp disconnects the goal from the component where 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.

From the perspective of control theory, our notion of "behavior" comprises merely the closed loop dynamics of a plant operating under feedback. Yet here, 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, situations often arise in which different control laws will stabilize different local regions 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. However, the process of combination requires switching between control modes – an aspect of "hybrid" control theory that is not presently well established. This paper develops methods of switching among local controllers that lead to the creation of more complex controllers with formal stability properties, and this specific instance of stable hybrid control represents our present notion of behavioral composition.

In order 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 law induces very good regulation about a given goal, there will always be ball states that it can not contain: the goal has a limited domain of attraction. Those same "missing" states, however, might be successfully handled by retuning the parameters of the juggling control law: a new goal point, or perhaps different gain settings. In this paper we will resort solely to such 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 the established palette of juggling control laws with different goal points or gain tunings.

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-Perez 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 consideration 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 will be touched upon

below.

Comprehensive work by Kak and colleagues (Tung and Kak 1994) has yielded a contemporary quasi-static assembly environment in the best traditions of HANDEY. The system builds a plan based upon sensed initial conditions, spawns an execution process that senses exceptions to the planned evolution of states and produces 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 dynamic 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 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 proposes a formalism for encoding and reasoning about the construction of action plans and their sensor-based execution (Lyons 1993, Lyons and Hendriks 1994). In contrast to our interest, however, Lyons explicitly avoids 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 proposed here seems to bear the closest correspondence to the "fine-motion planning with uncertainty" literature in robotics (for example, as expounded in the excellent text of Latombe (Latombe 1991)) originated by Lozano-Perez and colleagues (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, each constant piece corrupted by a constant disturbance vector of known (bounded) magnitude and uncertain direction arising from a uniform distribution over the unit ball. In contrast, we are interested in Newtonian dynamical models, and our control primitive is not a constant action but the entire range of actions consequent upon the closed loop policy, Φ, to be

specified below.[2]

Differences in models notwithstanding, we have derived considerable motivation from the "LMT" framework which attempts to match systematically the low level controllers' goals and capabilities with the more abstract requirements that characterize a task. In the work of Lozano-Perez and colleagues (Donald 1987, Erdmann and Mason 1988), high level progress is made through a sequence of controller actions whose successful termination is ensured via careful choice of compliant motions in the presence of rigid objects. The sequence itself has been designed via a back chaining of motion pre-images. In our work, the funnelling action of sensorless compliance to rigid objects (Erdmann and Mason 1988) is replaced by the stability of a general dynamical system, but we borrow heavily from the notion of pre-image back-chaining, as will be seen in Section 3.

1.2.4 *Juggling*

In (Burridge, Rizzi, and Koditschek 1995(1)) we introduced the notions of dynamical safety and obstacle avoidance, while in (Burridge, Rizzi, and Koditschek 1995(2)) we discussed the controller composition algorithm mentioned in Section 3. In this paper, we further our work by showing how to compose controllers in such a manner as to avoid an obstacle that disconnects the workspace.

In previous work (Buehler, Koditschek and Kindlmann 1990(2), Rizzi and Koditschek 1994), we have paid a great deal of attention 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

[2] Moreover, we never "bother" to develop an explicit disturbance model even though we are specifically interested in operating in the empirical world with all its uncertainties and unmodelled phenomena. On the one hand, the theory and experience of building working controllers (Koditschek 1987, Whitcomb, Rizzi and Koditschek 1993) teaches us that the inevitable small but persistent disturbances arising from modeling errors, sensor inaccuracy and slight miscalibration are countered by the local structural stability properties of stable dynamical systems. On the other hand, large, arbitrary and unanticipated perturbations should be recoverable as long as they are relatively rare and do not violate the domain of attraction of every feedback policy at our disposal. In the end, of course, the only acceptable test of one or another model is to place the finished system in the physical setting for which it was designed.

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 the present work 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 was 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. Spin and air drag on a falling ball modelled as a point mass, or 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, when a parts tray is severely jostled or arrives unexpectedly empty then an assembly robot's world model is likely to have been grossly violated. Such large potentially catastrophic disturbances typically occur quite rarely (otherwise, they would have been accounted for systematically by the robot's designers!).

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. Analogously, 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.

In contrast, feedback controlled systems can be designed to confer a level of robustness against both types of disturbance without recourse to explicit disturbance models. On the 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, cannot destroy stability of the (possibly biased) goal. On the other hand, if the closed loop system induced by a

feedback controller exhibits a global (or merely "large") domain of attraction then large disturbances can never take it "by surprise", so 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.

1.3.2 *Feedback Strategies as Funnels*

A decade ago, Mason (Mason 1985) 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.[3] 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) as being more obviously programmable.

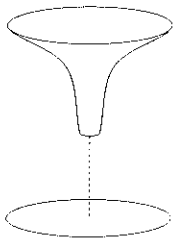


Figure 1: The Lyapunov Function as Funnel: Idealized graph of a positive definite function over its state space.

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 a closed loop dynamics on the domain (depicted as the x-y plane in these figures) all of whose motions lead down the funnel, then this 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 toward its bottom — the goal.

In Figure 2 we depict a complicated boundary such as might arise when the user specifies an "obstacle"

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

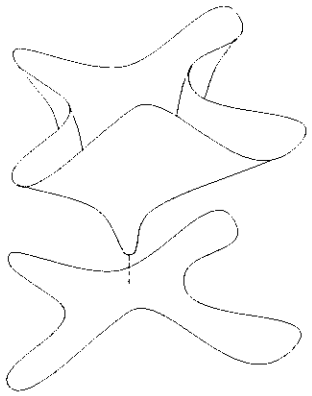


Figure 2: Ideal "funnel" capturing the entire "obstacle free" state space of the system.

— here, the set of states outside the bounded area that the user forbids us to enter.[4] Ideally, we would like a control strategy that "funnels" the entire state space of the system to the goal. We suggest this as well in the figure by sketching a complex funnel whose shape is exactly adapted to the domain.[5] Unfortunately, it is 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 1982, Koditschek 1992). Fortunately, we need not yet abandon the recourse to feedback.

1.3.3 *Sequential Composition as Preimage Backchaining*

Instead, we turn to another tradition introduced within the fine motion planning literature roughly one decade ago by Lozano-Pérez, Mason, and Taylor (Lozano-Pérez, Mason and Taylor 1984) — the notion of preimage back-chaining. Assume the availability of a palette of controllers whose induced funnels have targets that can be placed at will. Then we may deploy a set of multiple controllers whose combined (local) domains of attraction cover a large portion of the entire state space of the system. If such a set is rich enough that the domains overlap and goal sets lie within domains of other controllers, then it is possible to "backchain" away from the overall task goal, partitioning the state space into regions where the different controllers will be active, and arrive at a switching scheme between the controllers that will

[4] Of course, this picture dramatically understates the potential complexity of such problems since the free space need not be simply connected as depicted!

[5] In a previous series of papers with Rimon (Rimon and Koditschek 1992), the third author has addressed the design of such global funnels for purely geometric problems wherein the complicated portion of the domain is 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 Figure 11.

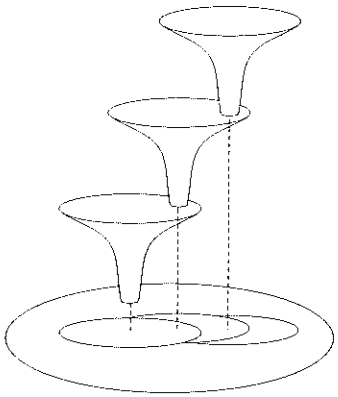


Figure 3: Sequential composition of funnels: the goal point of each previous controller lies within the domain of attraction induced by the next.

provably drive any state in the union of the various domains to the overall 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 appropriate region of state space, as sketched at the bottom of the figure. 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 partition, which is the only region to contain the goal set of its controller.

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

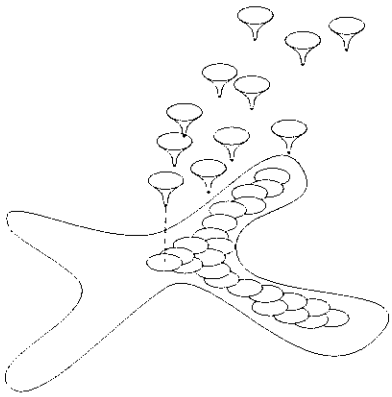


Figure 4: A suitable composition of local funnels partitions (a close approximation to) the free space state space into cells assigned to a unique controller.

This scheme affords the same robustness against disturbances discussed above. Local stability against

small disturbances is obtained because each funnel represents a Lyapunov function 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 state doesn't jump 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 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 state space, but they do not intersect any of the constituent domains of the partition of state space, then it follows that the state will never reach an obstacle while under the composite controller.

1.4 Contributions of the Paper

1.4.1 *Formal Presentation*

When written a little more carefully, as will be 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.

1.4.2 *Empirical Implementation*

Our experimental environment 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. Furthermore, 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 has proven to be very robust in the face of these problems. We were able to create a palette of controllers that successfully handled 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 Paper

The paper follows a logical development roughly suggested by Figures 1 through Figure 4. In Section 2 we provide a high level overview of the hardware and software, describe how entries in the controller palette (suggested by Figure 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 Physical Setting

2.1 Hardware System

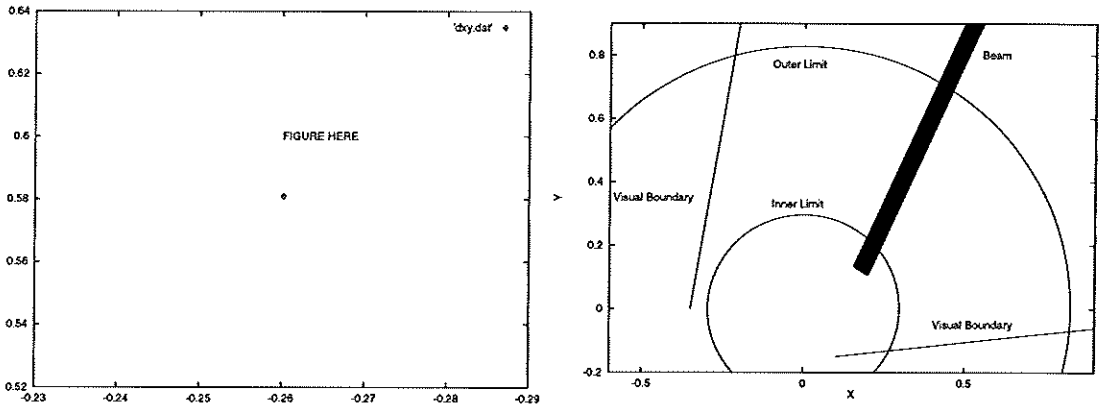


Figure 5: (a) The Bühgler Arm. (b) The horizontal workspace with obstacles.

For the experiments described in this paper, we use the "Bühgler," a three degree of freedom direct drive machine pictured in Figure 5(a), which we have discussed in a variety of other settings. Two cameras provide ball images at 60Hz, and all computation is done on a network of twenty transputers (Rizzi and Koditschek 1996, Rizzi 1994, Rizzi, Whitcomb and Koditschek 1992).

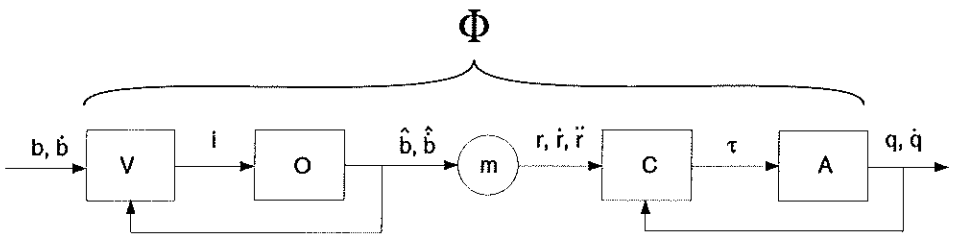


Figure 6: Flow chart showing the various state machines in the system.

Figure 5(b) shows an overhead view of the workspace, 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 obstacles because the robot will lose sight of the ball, thus failing its task, if the ball strays beyond these lines. Finally, a flat obstacle is introduced that disconnects the horizontal workspace. The robot may pass under the obstacle (which we will refer to as the *Beam*), and the ball may pass over it, but no contact between the two is possible in the region occupied by the Beam. 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 (see Footnote 1) maneuver is necessary for the task to succeed.

In this paper, we focus largely on the state of the ball, $(b, \dot{b}) \in \mathcal{B} \equiv I\!R^6$. For the most part, it will be convenient to decompose this state into the direct sum of horizontal and vertical subspaces, $\mathcal{B} = \mathcal{H} \oplus \mathcal{V}$, with $(h, \dot{h}) \in \mathcal{H}$, and $(v, \dot{v}) \in \mathcal{V}$, where $b = (h, v)$, $\dot{b} = (\dot{h}, \dot{v})$.

2.2 Software System

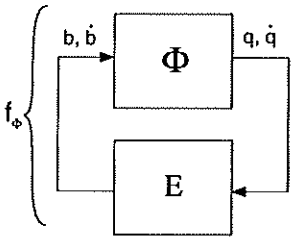
Our justification for limiting attention to the ball states is provided by the formal properties of the robot's sensing and control laws whose software implementation is depicted in Figure 6. Ball states, $(b, \dot{b}) \in \mathcal{B}$, are viewed by the vision system, V (cameras and image processing), which produces image plane locations, i. An observer, O, uses i to triangulate ball positions and produce estimates of the full state of the ball, $(\hat{b}, \hat{\dot{b}})$. These ball state estimates are 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 in (Rizzi and Koditschek 1996): for present purposes it suffices to note that the sensor feedback loop is globally asymptotically stable, so that $(\hat{b}, \hat{\dot{b}}) \to (b, \dot{b})$ whenever the ball enters the robot's field of view.

Estimated ball states, $(\hat{b}, \hat{\dot{b}})$, are passed through a memoryless transformation, m, to produce reference

robot states, $(r, \dot{r}, \ddot{r})$. We refer to m as a *mirror law.* The references, $(r, \dot{r}, \ddot{r})$, are fed to an inverse dynamics joint space controller (Whitcomb, Rizzi and Koditschek 1993, Whitcomb 1992), C, which produces torques, τ, for the actuator block, A (amplifiers, motors, etc.), which generates true robot joint states, $(q, \dot{q}) \in \mathcal{Q}$. It will sometimes be useful to decompose the robot position into its (joint space) constituents: $q = (\phi_r, \theta_r, \psi_r)$. The robot states are delivered to C for feedback control. Again, we note that the controller has been shown to be globally asymptotically stable, so that $(q, \dot{q}) \to (r, \dot{r})$ whenever $\ddot{r}$ exists.

In this paper we will completely ignore the transients in $V-O$ and $C-A$, and assume that $(\hat{b}, \dot{\hat{b}}) \approx (b, \dot{b})$ and $(q, \dot{q}) \approx (r, \dot{r})$, as these conditions are eventually guaranteed by the internal stability of our algorithm. This reduction of Φ (a dynamical controller) to m (a memoryless nonlinear transformation) affords the very simple composition method presented in Section 3. In practice, the transients are generally rapid enough to support this simplified reasoning. Of course, in some situations we inevitably pay a price in performance (see Section 4.4.3 for further discussion of this topic). In the sequel we will be concerned with various choices of m, but will use the symbol "Φ" to denote the various controllers that arise.

2.3 Closed Loop System



$$\begin{bmatrix} b(k+1) \\ \dot{b}(k+1) \end{bmatrix} = f_\Phi \left(\begin{bmatrix} b(k) \\ \dot{b}(k) \end{bmatrix} \right)$$

Figure 7: (a) The closed loop dynamics, f, induced by Φ. (b) The intermittency of the robot-environment interaction allows us to view the dynamics as a return map, with important implications vis à vis re-grasping (see footnote 1), as depicted in 14.

In Figure 7(a) the environment, E, is added to close the loop back from robot to ball. The resulting closed loop dynamics induced by a particular controller, Φ, are denoted f_Φ. Assumptions made in the previous section (that is, $(q, \dot{q}) \approx m(b, \dot{b})$) allow us to view f_Φ as evolving entirely within $\mathcal{B}$, the state space of the environment (ball). Note that the intermittent contacts characteristic of juggling promote a view of this closed loop from the perspective of a return map on a Poincaré section (Buehler, Koditschek and Kindlmann 1990(2)), thus f_Φ is a map rather than a vector field, as illustrated in Figure 7(b).

We are interested in encoding and achieving user goals taking the form of a desired set of world states, $\mathcal{G} \subset \mathcal{B}$. A controller, Φ, achieves those goals if $\mathcal{G}$ is attracting and invariant under f_Φ. If so, we would like to characterize its domain of attraction, denoted $\mathcal{D}_\mathcal{G}(\Phi) \subset \mathcal{B}$.

During the last five years we have developed an approach to robot controller design that has produced controllers with large domains of attraction to single set points by essentially reshaping the total energy exchanged between the robot and its environment, as depicted in Figure 1.

2.4 Example: The Juggle Φ_J

Our juggling behavior is induced by a control law in which a single ball is struck repeatedly by the robot paddle in such a way as to direct it toward a limit cycle of specified apex height at a desired horizontal position with no horizontal velocity. We refer to this controller as Φ_J. According to assumptions made in Section 2.2, Φ_J reduces to m_J, a memoryless (nonlinear) transformation of the ball state, which we now describe: (Rizzi 1994, Rizzi and Koditschek 1993).

2.4.1 *The Mirror Law, m_J*

For a given ball state, $(b, \dot{b})$, we begin by inverting the kinematics of the arm to calculate the robot position that would just touch the ball: $q(b) = (\phi_b, \theta_b, 0)$. For convenience, we choose zero for the third degree of freedom to eliminate redundancy. This effectively gives us the "ball in joint space coordinates."

Next, we express formulaically a robot strategy that causes the paddle to respond to the motions of the ball in four ways:

(i) $\phi_r = \phi_b$ causes the paddle to track under the ball at all times.

(ii) θ_r "mirrors" the vertical motion of the ball as expressed by the original planar mirror law (Buehler, Koditschek and Kindlmann 1990(2)).

(iii) Radial motion of the ball causes the paddle to raise and lower, resulting in the normal being adjusted to correct for radial deviation in the ball position.

(iv) Angular motion of the ball causes the paddle to roll, again adjusting the normal so as to correct for lateral position errors.

Define the ball's *vertical energy* and *radial distance* as

$$\eta_b \triangleq \gamma b_z + \frac{1}{2}\dot{b}_z^2 \quad \textit{and}, \quad \rho_b \triangleq \sqrt{b_x^2 + b_y^2} \tag{1}$$

respectively, where γ represents gravity. The complete mirror law combines these two measures with the set point apex position expressed in joint space coordinates: $\mathcal{G} \triangleq (\bar{\rho}_b, \bar{\dot{\phi}}_b, \bar{\eta}_b)$ to form the function

$$m_J(b, \dot{b}) \triangleq \begin{bmatrix} \phi_r \\ \theta_r \\ \psi_r \end{bmatrix} = \begin{bmatrix} \underbrace{\dot{\phi}_b}_{(i)} \\ \underbrace{-\theta_b(\kappa_0 + \kappa_1(\eta_b - \bar{\eta}_b))}_{(ii)} + \underbrace{\kappa_{00}(\rho_b - \bar{\rho}_b) + \kappa_{01}\dot{\rho}_b}_{(iii)} \\ \underbrace{\kappa_{10}(\phi_b - \bar{\phi}_b) + \kappa_{11}\dot{\phi}_b}_{(iv)} \end{bmatrix}. \tag{2}$$

There are nine parameters in (2): three representing the position of the desired set point (the desired velocity is zero), four for the PD gains for radial and angular control, and two for the gains regulating vertical energy. In Sections 2.5 and 4 we will create controllers that act in different ways on different areas of the workspace by changing the values of these parameters. At such times it will be useful to think of our controller as determined by an element of parameter space, $\Phi(\xi), \xi \in \Xi$, where $\xi = (\bar{\eta}_b, \bar{\rho}_b, \bar{\dot{\phi}}_b, \kappa_{00}, \kappa_{01}, \kappa_{10}, \kappa_{11}, \kappa_0, \kappa_1)$.

As we will see, different choices of the parameters, ξ, can lead to qualitatively different behavior of f_Φ. We will use the notation Φ_J to refer to a "typical" controller having parameters from the region of Ξ leading to juggling behavior with non-zero apex height and an attracting limit cycle with single set point apex $\mathcal{G}$.

2.4.2 *Analysis, Simulation, and Empirical Exploration of* $\mathcal{D}(\Phi_J)$.

It is useful to decompose the goal point, $\mathcal{G}$, into its horizontal and vertical components: $\mathcal{G} = (\mathcal{G}_H, \mathcal{G}_V)$, where $\mathcal{G}_H = (\bar{\rho}, \bar{\dot{\phi}})$, and $\mathcal{G}_V = \bar{\eta}$. In previous work, Rizzi *et. al.* (Rizzi and Koditschek 1994) have shown that the vertical and horizontal subsystems of f_{Φ_J} are essentially decoupled. This allows us to examine the two systems independently and combine the results.

Buehler *et. al.* (Buehler, Koditschek and Kindlmann 1990(2)) showed that the restriction of f_{Φ_J} to $\mathcal{V}$ has global attraction to $\mathcal{G}_V$. Currently, however, we can only show local attraction for the horizontal system. Empirical results show that $\mathcal{D}_H(\Phi_J)$ indeed changes very little as apex height changes within the relevant range of heights, and we focus largely on the horizontal system for the remainder of this paper. Henceforth, when we refer to $\mathcal{D}(\Phi_J)$, we will mean $\mathcal{D}_H(\Phi_J)$.

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 will be 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.

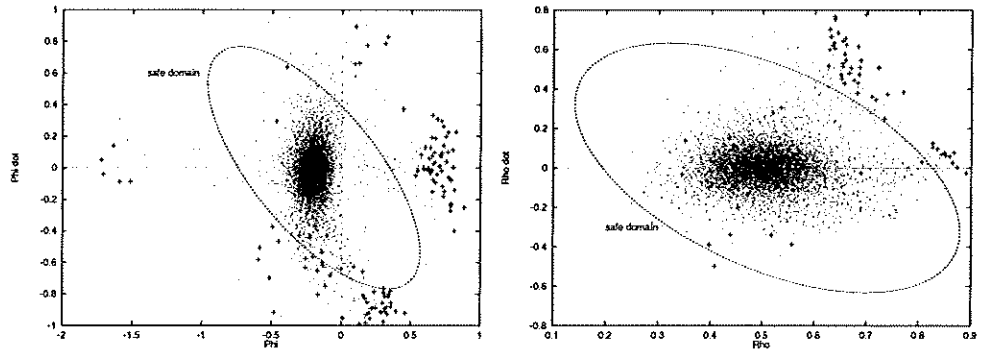


Figure 8: Empirically derived estimate of the horizontal juggling domain, $\mathcal{D}_H(\Phi_J)$. Note for the purposes of subsequent discussion that the symbol(s) "+" ("·") denote the points which failed (succeeded) in remaining within the robot's annular and visual workspace under the action of Φ_J.

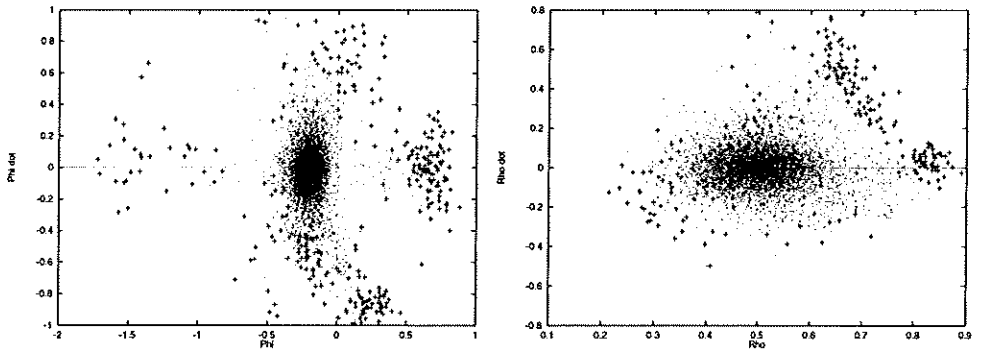


Figure 9: Numerically derived estimate for the horizontally safe juggling domain. The symbol(s) "+" ("·") denote(s) the points which failed (succeeded) in remaining within the robot's annular and visual workspace under the action of the numerical simulation of Φ_J.

Due to the complexity of the system, it is difficult to "feel out" the boundaries of the domain of attraction for Φ_J through systematic juggling experiments. Nevertheless, in Figure 8, we display experimental data used to formulate an approximation of what we will come to 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 5(b) with no flat obstacle inserted (Burridge, Rizzi and Koditschek 1995(1)). The particular choice of parameters, ξ, for this controller came 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 Figure 8 run through the numerical simulation of the robot operating under Φ_J. Since all but one of the starting conditions that failed on the real robot failed in simulation, we are persuaded that the simulation provides a conservative approximation to the robot's true actions.

2.5 The Complete Palette

In Section 4 we will deploy a selection of different controllers from the palette represented by ξ in (2) with which to run our experiments. Most of those controllers will be copies of a particular $\Phi_J(\xi)$ which vary only in the angular component, $\bar{\phi}$, of the set point. Taking advantage of the rotational symmetry of m_J in (2), we will rotate a numerically derived domain of attraction for $\Phi_J(\xi)$ along with the set points, completing the requirements for our methods.

In addition to varying the set point for $\Phi_J(\xi)$, we will use two other controllers based on (2), but with choices of ξ that lead to qualitatively different behavior.

2.5.1 *Palming:* Φ_P

If we set to zero the gains regulating the vertical energy of the ball, (κ_0, κ_1), then we find that the ball will lose energy, finally coming to rest on the paddle. The lateral terms (iii, iv) of (2), however, will continue to stabilize the ball at the horizontal set point, $\mathcal{G}_H = (\bar{\rho}_b, \bar{\dot{\phi}}_b)$. 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 can not 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 will use the same horizontal domain for palming as for juggling, but the vertical energy component will be restricted to values near zero, as implied by

empirical tests.

2.5.2 *Catching:* Φ_C

Φ_J may behave poorly when the ball is nearly in continuous contact with the paddle. In these situations, the system may not notice small bounces, and is very susceptible to noise in the vision system and observer. Φ_P, on the other hand, may have PD gains tuned too aggressively for large bounces of the ball, and should be switched on only when the ball has low vertical energy.

In order 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 to rest on the paddle. Currently, a variant of Φ_J is used in which the vertical set point, $\bar{\eta}_b$, is very low, and the robot meets the ball higher than normal. For an intuitive view of this, see Figure 15.

Due to the hybrid nature of this controller, we have no analytic or numerical results for its domain of attraction. Instead, we use a very conservative estimate based on the numerically-derived $\mathcal{D}(\Phi_J)$ 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.

3 Formal Idea: Safe Sequential Composition

The technique proposed here is a variant of the preimage backchaining idea introduced in (Lozano-Pérez, Mason and Taylor 1984), although their algorithm was used for compliant motion in the face of velocity uncertainty with respect to a generalized damper model of dynamics. We choose to 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: $\mathcal{G}(\Phi_i) = \mathcal{G}$. After finding Φ_i, choose a partial ordering on Γ which transforms it into an acyclic graph with all paths leading down to Φ_i. For example, consider a breadth-first search back from Φ_i, as outlined in the following

steps:

1. Let the OPEN-LIST contain Φ_i. Let $\bar{\mathcal{D}}_{\Phi_i} = \mathcal{D}_{\Phi_i}$, $N = 1$, and $\mathcal{D}_{\mathcal{S}_G}(N) = \mathcal{D}_{\Phi_i}$.

2. Append to the end of the OPEN-LIST the list of all controllers which *prepare* the first element, and have not previously been placed on the OPEN-LIST.

3. Remove the first element of the OPEN-LIST.

4. For Φ_j, the new first element of OPEN-LIST, let $\bar{\mathcal{D}}_{\Phi_j} = \mathcal{D}_{\Phi_j} - \mathcal{D}_{\mathcal{S}_G}(N)$, and let $\mathcal{D}_{\mathcal{S}_G}(N+1) = \mathcal{D}_{\mathcal{S}_G}(N) \cup \mathcal{D}_{\Phi_j}$. Increment N.

5. Repeat 2,3 and 4 until OPEN-LIST is empty.

At the end of this process, the regions $\bar{\mathcal{D}}_{\Phi_i}$ will be cells in a partition of $\mathcal{D}_{\mathcal{S}_G}(M)$, where M is the number of cells in the partition. The automaton will choose controller Φ_j exactly when the ball state lies within $\bar{\mathcal{D}}_{\Phi_j}$.

The domain of each controller in $\mathcal{D}_{\mathcal{S}_G}$ is partitioned in such a manner that the automaton will only switch to a new controller if the ball moves into the domain of a controller which was processed earlier from OPEN-LIST. Since the attracting goal set of every controller (except Φ_i) lies in the domain of another closer to the goal, it is inevitable that $\bar{\mathcal{D}}_{\Phi_i}$ finally be reached, and thus the task goal set will be approached. This process is depicted in Figure 10, where the funnels represent the various domains, each attracting to the goal set at the middle of the funnel, and the partition of state space is shown on the horizontal plane below them.

The domain of attraction for the goal set can now be considered to be not just $\mathcal{D}_{\Phi_i}$, but $\mathcal{D}_{\mathcal{S}_G} = \bigcup_{\Phi_l \in \mathcal{S}_G} \mathcal{D}_{\Phi_l}$. The process of constructing a composite controller from a set of "simpler" controllers is denoted by $\Phi = \bigvee \mathcal{U}_\mathcal{G}$, and is depicted in Figure 10. In that figure, the only active part of each funnel is the part not over the higher-priority funnels.

3.2 Obstacles and Safety

There are two types of obstacle with which we are concerned. The first consists of regions of the configuration space "off limits" to the robot, $\mathcal{O}_\mathcal{R}$, 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. The second type of obstacle consists of subsets of ball state space

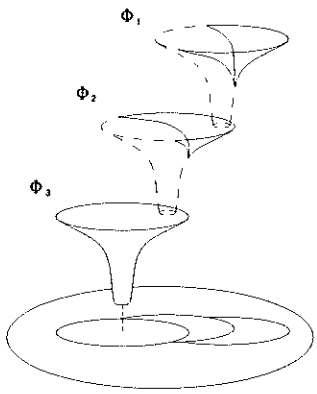


Figure 10: Sequential composition of controllers: Each controller is only active in the part of its domain not already covered by those nearer the goal. Here, $\Phi_1 \succeq \Phi_2 \succeq \Phi_3$, and Φ_3 is the goal controller.

which are "off limits" to the ball, even when it is not in contact with the robot. This type of obstacle is particularly interesting in the context of intermittent contact, which results in discrete dynamics, and has not been studied methodically to date. The system is considered to have failed if it allows the ball to enter any such regions, collectively denoted $\mathcal{O}_\mathcal{B}$.

Recall that for a particular mirror law, m, each ball state induces a corresponding robot configuration. We define a workspace obstacle, $\mathcal{O}$, as the union of ball state obstacles and the set of ball states that induce the robot into the robot obstacles, which we will loosely call $m^{-1}(\mathcal{O}_\mathcal{R})$. Thus $\mathcal{O} = \mathcal{O}_\mathcal{B} \bigcup m^{-1}(\mathcal{O}_\mathcal{R})$.

3.2.1 *Definition of Safety*

The "time to obstacle", $\tau_\mathcal{O}$, of a particular ball state is defined as the time it would take to reach $\mathcal{O}$ in the absence of the paddle. Similarly, the time until next impact with the paddle given a particular control law, Φ, is denoted τ_Φ.

A controller Φ is *safe with respect to* $\mathcal{O}$ if there can be found an invariant open subset in $\mathcal{D}(\Phi)$ such that the next impact with the robot always occurs before the ball reaches an obstacle (i.e., $\tau_\Phi < \tau_\mathcal{O}$) for all states within the subset. We denote such a *safe domain* by $\mathcal{D}_\mathcal{O}(\Phi)$.

Notice that for any obstacle and controller, a safe domain is invariant during flight by definition. Thus, once Φ has been selected, only a violation of our models — e.g., a perturbation during flight or impact, an overly large observer error, etc. — can cause a transition out of $\mathcal{D}_\mathcal{O}(\Phi)$.

3.2.2 *Safe Deployments*

If all the constituent controllers, Φ_i, in a deployment are safe with respect to an obstacle, then it follows that the resulting composite controller, $\Phi = \bigvee \Phi_i$, *must* also be safe with respect to the obstacle. This is because any state within $\mathcal{D}(\Phi)$ is also within $\mathcal{D}(\Phi_j)$ for the current active Φ_j, and Φ_j will not drive the state into the obstacle from within $\mathcal{D}(\Phi_j)$.

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. However, in practice (Buehler, Koditschek and Kindlmann, 1994, Buehler, Koditschek and Kindlmann 1990(1)) 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, ξ_0, with goal point $\mathcal{G}_0 = (\bar{\rho}_0, \bar{\dot{\phi}}_0, \bar{\eta}_0) = (0.6, 0.0, 4.5)$. Using the simulator introduced in Section 2.4.2, 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 $\mathcal{H}$ (the starting height, v_i, is the same for all runs, as we are not concerned here with the vertical subsystem, owing to the decoupling of $\mathcal{H}$ and $\mathcal{V}$ discussed in Section 2.4.2). 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 are reversed. These plots demonstrate the rather complicated shape of the domain of attraction, as we have tried to suggest intuitively in Figure 2.

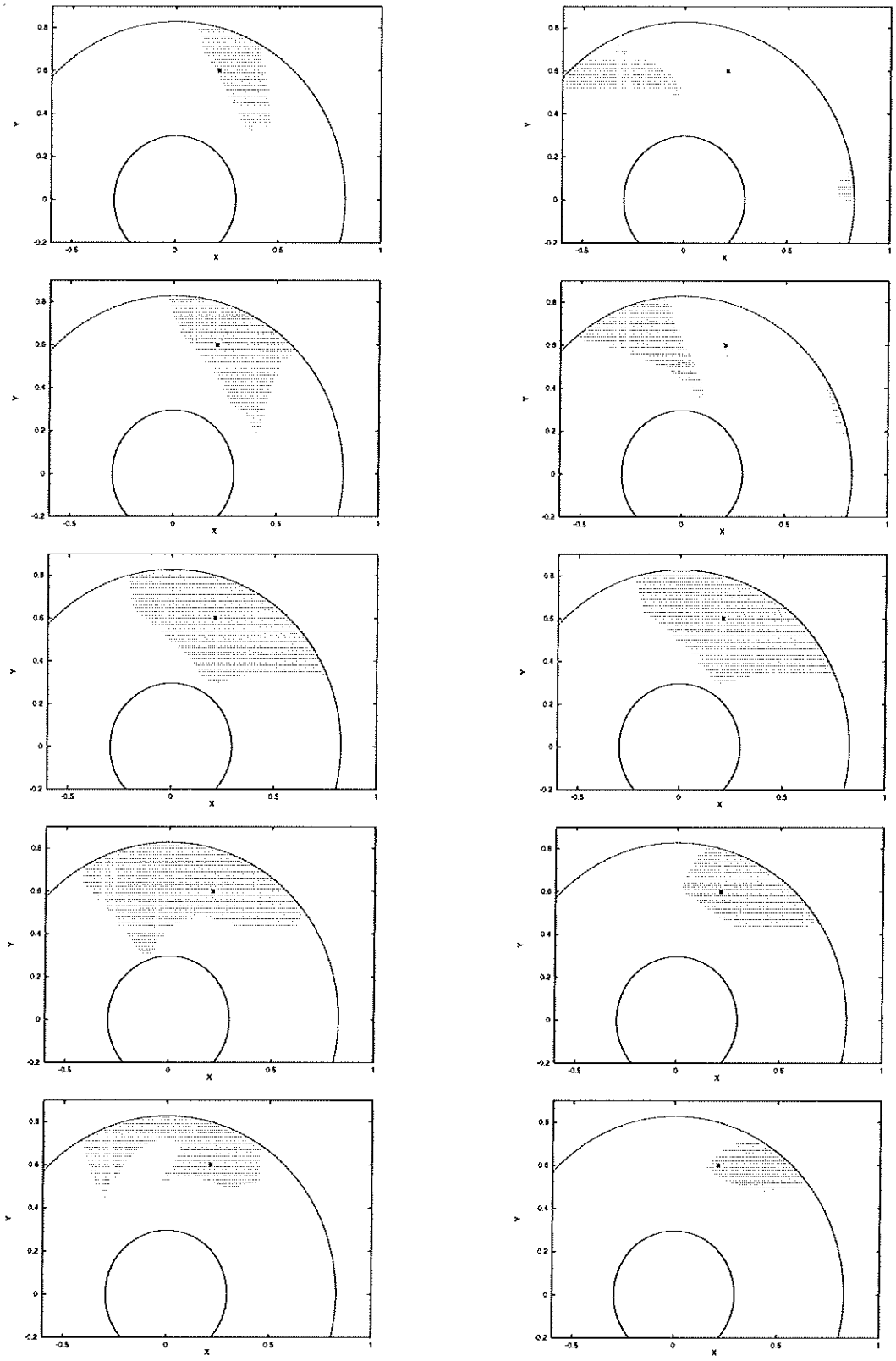


Figure 11: Velocity "spines" – projections onto the zero-velocity plane of slices of $\mathcal{H}$ at regularly spaced velocities – of the invariant attracting domain of Φ_J. Shaded regions denote initial conditions that were successfully contained in the workspace while being brought to the goal via iterates of f_{Φ_J}. (a) Left Column: Varying initial $\dot{x}_i$ from negative (top) to positive (bottom) with $\dot{y}_i = 0$. (b) Right Column: Varying initial $\dot{y}_i$ from negative to positive with $\dot{x}_i = 0$.

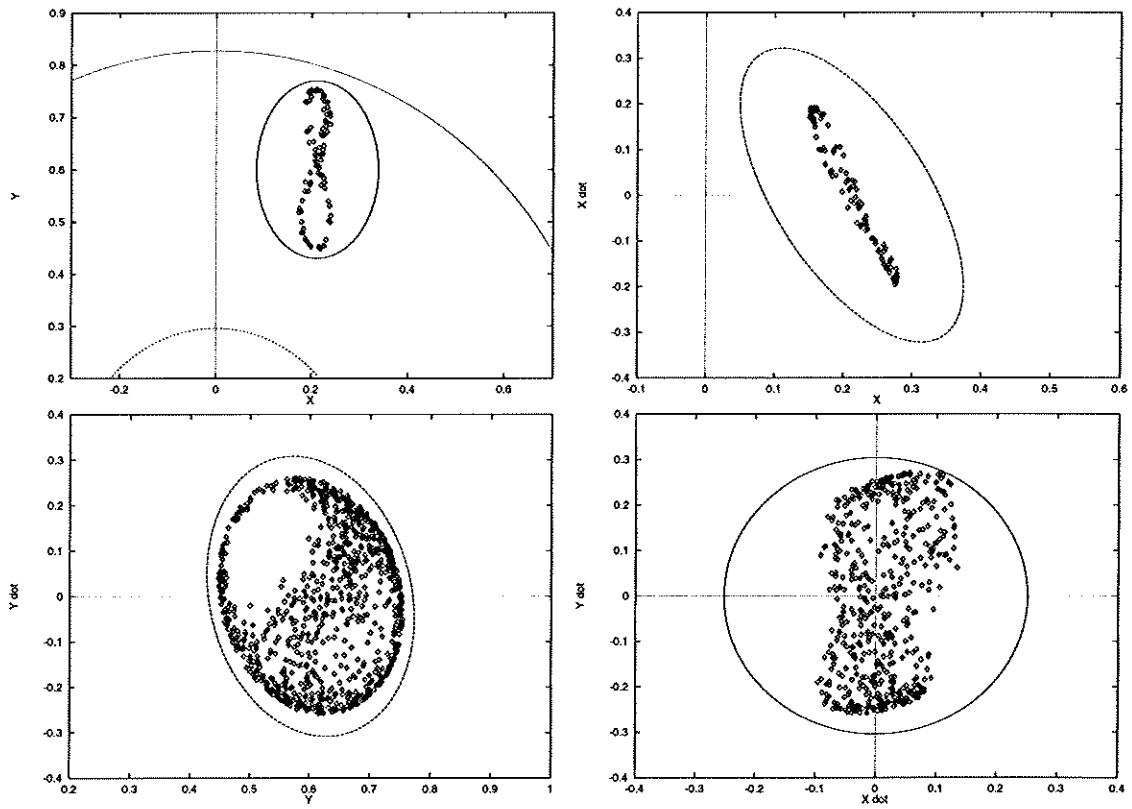


Figure 12: Four 2-D slices of the invariant ellipsoid, $\mathcal{D}_0 \subset \mathcal{H}$ (smooth curves), and its one-bounce forward image, $f_\Phi(\mathcal{D}_0)$ (dots).

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, yet which enjoys the invariance property with respect to f_{Φ_J}, namely that $f_{\Phi_J}(\mathcal{D}_0) \subset \mathcal{D}_0$. Ellipsoids make an obvious candidate shape, and have added appeal from the analytical viewpoint (although these will be 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).

Starting with a candidate ellipsoid, $\mathcal{N}_{test} \subset \mathcal{H}$, centered at $\mathcal{G}_0$, 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 "shape" 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 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.[6] 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, (2), 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 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 $\mathcal{D}_{PADDLE}(\Phi_{J_0})$: *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 $\mathcal{H}$ 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 $\mathcal{D}_{BEAM}(\Phi_{J_0})$: *Safety with respect to the Beam*

In Figure 13(a), 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 (ellipse with 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 Figure 11). The dark region represents all states which 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 13(b), 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

[6] 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 $\mathcal{N}_{test}$.

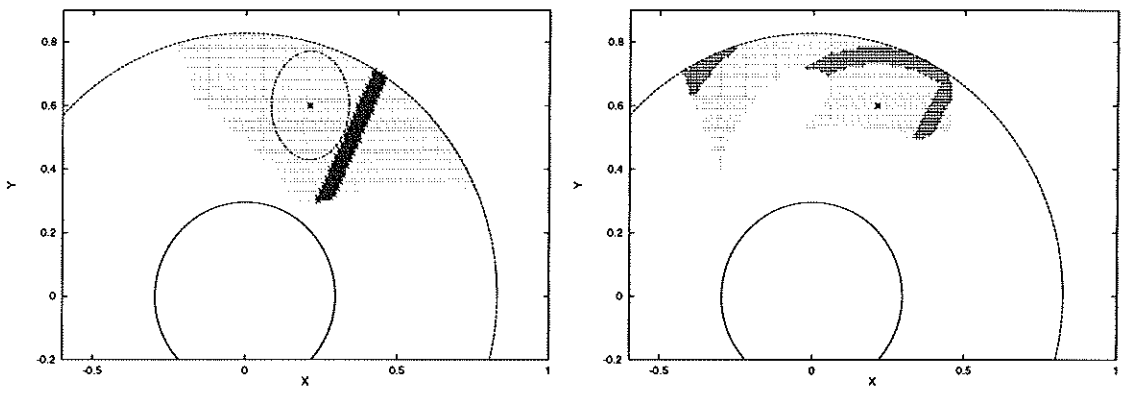


Figure 13: Safe domain for Φ_{J_0} with the Beam inserted. Light gray areas represent successful initial states, darker areas show states that eventually hit the Beam. (a) Zero initial velocity. The appropriate slice of $\mathcal{D}_0$ has been added for comparison. (b) $\dot{x}_i = 1.5\frac{m}{s}$, showing preimages of the Beam.

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 dynamics of f_{Φ_J} to cross it (recall from Section 2.3 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 will call this new disconnected forward invariant domain $\mathcal{D}_1$ throughout the sequel.

In principle, finding a $\mathcal{D}_1$ is not much more difficult than finding a $\mathcal{D}_0$. 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 will be 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 have found an ellipsoid, $\mathcal{N}_n$, whose forward image is completely contained in $\mathcal{D}_0$. By construction, the union of ellipsoids, $\mathcal{D}_1 := \mathcal{D}_0 \bigcup \mathcal{N}_1 \bigcup \ldots \bigcup \mathcal{N}_n$, 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

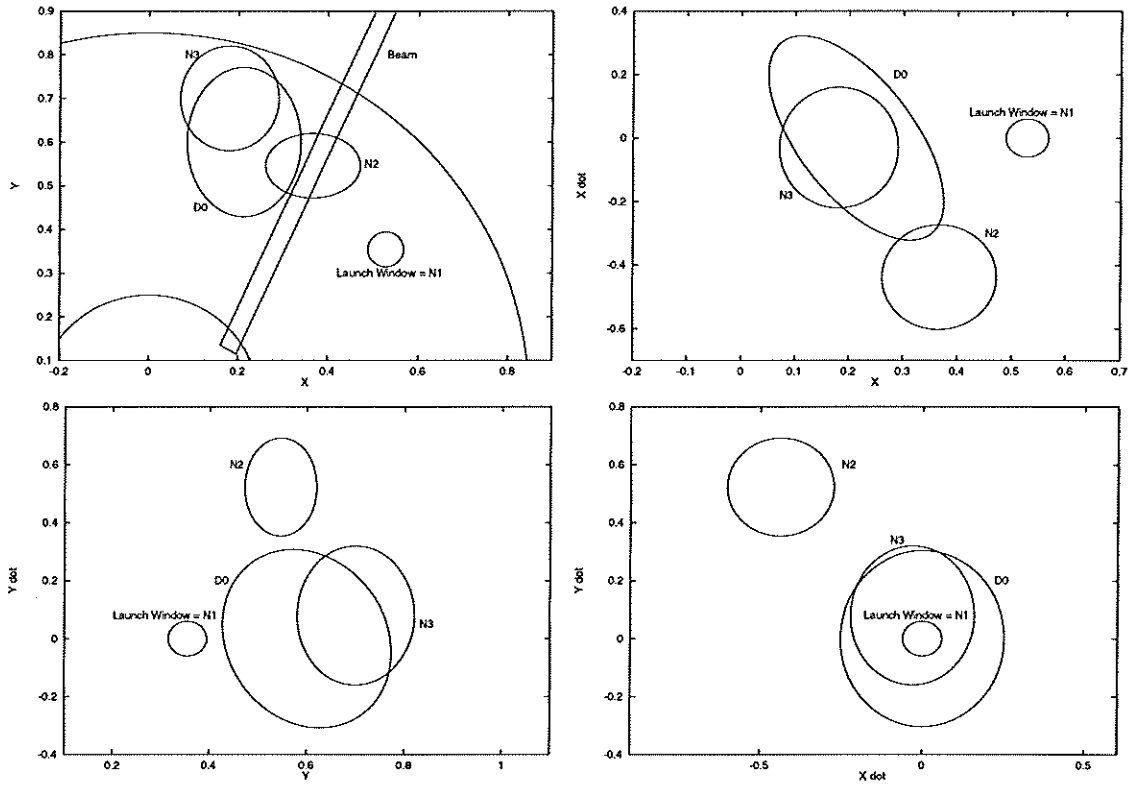


Figure 14: Four projections of the union of the disjoint ellipsoids, $\mathcal{D}_1 := \mathcal{D}_0 \bigcup \mathcal{N}_1 \bigcup \mathcal{N}_2 \bigcup \mathcal{N}_3$, created by analysis of empirical data.

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 turn 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 $\mathcal{H}$.

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 $\mathcal{H}$, but its projection onto $\mathcal{V}$ is much smaller. We only allow palming for those states with very low vertical energy. We also use a catching controller, Φ_C, that has a small domain at $\mathcal{G}_T$, and 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 non-zero 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 $\Phi_{J_0}(\bar{\phi}_T) \not\preceq \Phi_P(\mathcal{G}_T)$, 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 region of states toward its goal, which lies within $\mathcal{D}_C$. In the right hand figure, Φ_C draws its domain down (in vertical energy) to its goal, which lies within $\mathcal{D}_P$.

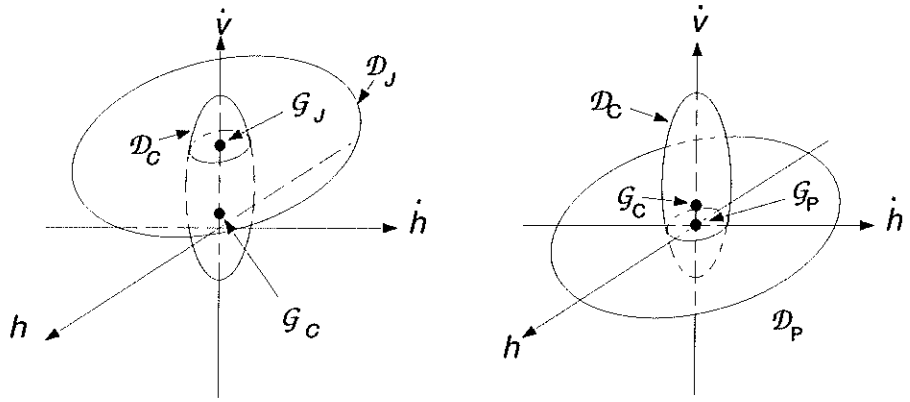


Figure 15: Juggle to Catch to Palm: (a) The Juggle drives all of its domain to $\mathcal{G}_J$, which lies within the domain for the Catch, but with non-zero vertical energy. (b) The Catch reduces the vertical energy of the ball, bringing it down into the domain for Palming.

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

Finally, we created 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

Ellipses	Type	Goal:ϕ	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	$\mathcal{D}_0$

Table 1: The full deployment, with controller types, goal points and domain types. All goal points have the same radial component, $\bar{\rho}$, so we display here only the angular component, $\bar{\phi}$. Ellipse numbers correspond to those in Figure 16.

blocked 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 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.

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 the palming cell, where it should remain.

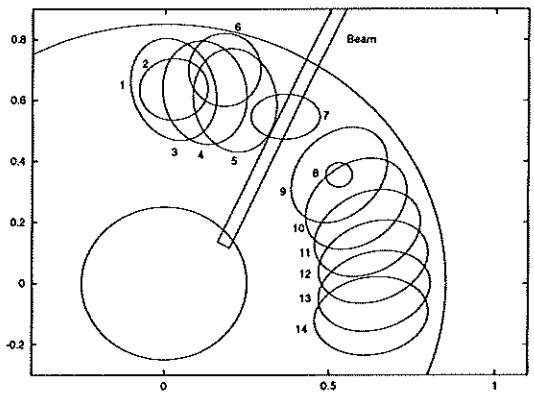


Figure 16: A "theoretical" deployment using the jump controller of Figure 14.

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 will 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 will see in Section 4.4.1 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 then 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 non-zero velocity parts of $\mathcal{D}_1$. There are a total of fourteen ellipsoids for eleven different controllers, the union of four of them associated with the trans-Beam controller's domain, $\mathcal{D}_1$, and the same ellipsoid being used for the final juggling and palming controllers (numbers 1 & 3 in the figure). This is our hand-crafted realization of the concept depicted in Figure 4.

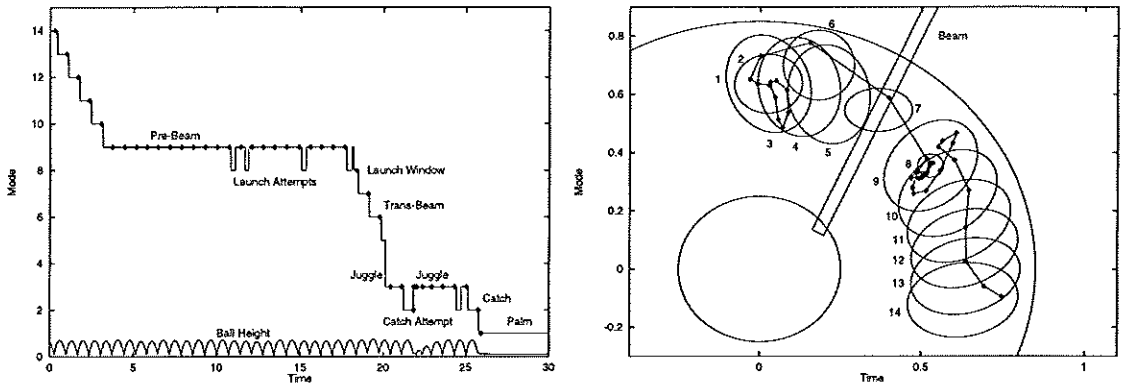


Figure 17: (a) Cell (ellipse) number plotted against time for a typical run, with a trace of the ball's vertical position added for reference. The dots represent the mode at the moment of impact. (b) Horizontal projection of the same trace, superimposed on the deployment of Figure 16. The dots show the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

4.3.1 *A Typical Run*

Figure 17(a) shows the trace of typical run, with active controller plotted against time, while Figure 17(b) 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 (9), where it stayed for some time, waiting for the ball to enter the launch window, $\mathcal{N}_1$ (8), of the trans-Beam controller's domain, $\mathcal{D}_1$.

At this point the "messiness" of the system mentioned in Section 1.4.2 was most evident, as the observer (recall Figure 6) would "hallucinate" the ball state within $\mathcal{N}_1$ (8), activating the trans-Beam controller, then change its estimate and take the ball state back out of $\mathcal{N}_1$, turning on the pre-Beam controller (9) again (these spurious transitions are labelled "Launch Attempts" in Figure 17(a)). Although the action of the robot at impact (dots in Figure 17) was almost always correct, the third degree of freedom, ψ_r (recall eqn.(2)), would visibly shift back and forth during the flight of the ball due to the large difference in goal location between the pre- and trans-Beam controllers (9 and 8). Eventually, the true ball state entered $\mathcal{N}_1$, the trans-Beam controller (8) became active, and the task proceeded.

In the particular run shown in Figure 17, the ball overshot controllers (5) and (4), going straight from (6) to (3). The first catch attempt was unsuccessful, with the ball exiting $\mathcal{D}_C$ (2) before entering $\mathcal{D}_P$ (1), so the juggle (3) turned back on, and after a couple of hits, Φ_C turned on again, and the ball was

	Experiment 1	Experiment 2	Experiment 3
Number of successful runs:	49 (82%)	57 (95%)	23 (38%)
Time (s) to reach Pre-launch mode : mean	3.954	2.775	1.276
std. dev.	2.086	1.283	0.428
Time (s) spent in Pre-launch mode : mean	9.811	8.368	13.671
std. dev.	10.020	5.879	8.498
Time (s) from Launch to Palming : mean	3.871	5.837	4.351
std. dev.	1.821	4.456	2.204
Time (s) from Throw to Palming : mean	17.636	16.980	19.299
std. dev.	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

Table 2: Statistics characterizing three sets of 60 trials each. The first three rows give a breakdown for the successful runs of the total time into three segments: Throw to Pre-launch, time spent in Pre-launch, 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.

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 to be described below. We also show for all successful trials the mean and standard deviation of the length of time from introduction to pre-launch mode, time spent there before launch, time from launch to palming, and total time from start to finish. We also show the total number of regressive steps considered. Those "hallucinated" 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 (3). The regressive steps, both those "hallucinated" and those taken, remind us of the gulf between the ideal assumption, $(\hat{b}, \hat{\dot{b}}) \approx (b, \dot{b})$, and reality, as discussed in Sections 2.2 and 4.2.2.

4.4 Deployment of Invariant Sets

We ran three sets of experiments, each with sixty trials. We started off carefully introducing the ball to a theoretically correct deployment. Next, we added a set of "catch-alls" – 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 in order 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 have summarized in Table 2.

4.4.1 *Theoretically Correct Deployment*

In our first set of experiments, the deployment was exactly 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. Five times the ball was knocked out of all safe domains 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 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 strays away from the controllers of Experiment 1, and channelling it back into the theoretically sound portion of the

deployment. In addition, the presence of the catch-all 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 catch-all 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 pre-launch 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 allowed 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 only react quickly enough to contain throws that landed 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 s). 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 in order for the robot to contain it.

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 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 in order to ensure that there was a path from the starting region to the goal. We would like to develop automatic methods for creating deployments given families 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 in (Rimon and Koditschek 1992). At any moment in time, there would be tradeoffs between the size of the domain and the velocity of the local goal toward the task goal.

References

[1] Brockett, R. W., 1983. Asymptotic stability and feedback stabilization. In *Differential Geometric Control Theory*, Brockett, R. W., Millman, R. S., and Sussman, H. J. editors, Birkhäuser, pages 181–191.

[2] Buehler, M., Koditschek, D. E., and Kindlmann, P. J., 1994. Planning and control of a juggling robot. *The International Journal of Robotics Research*, 13(2):101–118.

[3] Buehler, M., Koditschek, D. E., and Kindlmann, P. J., 1990. A Simple Juggling Robot: Theory and Experimentation. In *Experimental Robotics I*, Hayward, V., and Khatib, O. editors, Springer-Verlag, pages 35–73.

[4] Buehler, M., Koditschek, D. E., and Kindlmann, P. J., 1990. A family of robot control strategies for intermittent dynamical environments. *IEEE Control Systems Magazine*, 10:16–22.

[5] Burridge, R. R., Rizzi, A. A., and Koditschek, D. E., 1995. Obstacle avoidance in intermittent dynamical environments. In *Proceedings Fourth International Symposium on Experimental Robotics*, pages 43–48.

[6] Burridge, R. R., Rizzi, A. A., and Koditschek, D. E., 1995. Toward a dynamical pick and place. In *Proceedings IROS*, pages 2:292–297.

[7] Donald, B. R., 1987. *Error Detection and Recovery for Robot Motion Planning with Uncertainty.* PhD thesis, Massachusetts Institute of Technology.

[8] Erdmann, M. A., and Mason, M. T., 1988. An exploration of sensorless manipulation. *IEEE Journal on Robotics and Automation*, 4(4):635–642.

[9] Erdmann, M. A., 1984. On motion planning with uncertainty. Technical Report AI-TR-810 (MS Thesis), Cambridge, Massachusetts Institute of Technology.

[10] Ish-Shalom, J., 1984. The cs language concept: A new approach to robot motion design. In *Proceedings 23rd IEEE Conference on Decision and Control*, pages 760–767.

[11] Koditschek, D. E., 1992. Task encoding: Toward a scientific paradigm for robot planning and control. *Journal of Robotics and Autonomous Systems*, 9:5–39, 1992.

[12] Koditschek, D. E., 1987. Robot control systems. In *Encyclopedia of Artificial Intelligence*, Shapiro, S. editor, John Wiley and Sons, Inc., pages 902–923.

[13] Latombe, J., 1991. *Robot Motion Planning.* Kluwer, Boston, MA.

[14] Lozano-Pérez, T., et al, 1987. Handey: A robot systems that recognizes, plans, and manipulates. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 843–849.

[15] Lozano-Pérez, T., Mason, M. T., and Taylor, R. H., 1984. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–23.

[16] Lyons, D. M., and Hendriks, A. J., 1994. Planning by adaption: Experimental results. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 855–860.

[17] Lyons, D. M., 1993. Representing and analyzing action plans as networks of concurrent processes. *IEEE Transactions on Robotics and Automation*, 9(3):241–256.

[18] Mason, M. T., 1985. The mechanics of manipulation. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 544–548.

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

[20] Rimon, E., and Koditschek, D. E., 1992. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 8(5):501–518.

[21] Rizzi, A. A., and Koditschek, D. E., 1993. Further progress in robot juggling: The spatial two-juggle. In *Proceedings IEEE International Conference on Robotics and Automation.*

[22] Rizzi, A. A., and Koditschek, D. E., 1994. Further progress in robot juggling: Solvable mirror laws. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2935–2940.

[23] Rizzi, A. A., and Koditschek, D. E., 1996. An active visual estimator for dexterous manipulation. *IEEE Transactions on Robotics and Automation* (to appear).

[24] Rizzi, A. A., 1994. *Dexterous Robot Manipulation.* PhD thesis, Yale University.

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

[26] Tung, C. P., and Kak, A. C., 1994. Integrating sensing, task planning and execution. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2030–2037.

[27] Whitcomb, L. L., 1992. *Advances in Architectures and Algorithms for High Performance Robot Control.* PhD thesis, Yale University.

[28] Whitcomb, L. L., Rizzi, A. A., and Koditschek, D. E., 1993. Comparative experiments with a new adaptive controller for robot arms. *IEEE Transactions on Robotics and Automation*, 9(1):59–70.

[29] Whitney, D. E., 1977. Force feedback control of manipulator fine motions. *ASME Journal of Dynamical Systems*, 98:91–97.

Contents

1 Introduction 2

1.1 Statement of the Problem . . . 2

1.2 Previous Literature . . . 3

1.2.1 *Pick and Place in Cluttered Environments* . . . 3

1.2.2 *Hybrid Control and Discrete Event Systems* . . . 4

1.2.3 *Error Detection and Recovery* . . . 4

1.2.4 *Juggling* . . . 5

1.3 Overview of the Approach . . . 6

1.3.1 *Feedback Confers Robustness* . . . 6

1.3.2 *Feedback Strategies as Funnels* . . . 7

1.3.3 *Sequential Composition as Preimage Backchaining* . . . 8

1.4 Contributions of the Paper . . . 10

1.4.1 *Formal Presentation* . . . 10

1.4.2 *Empirical Implementation* . . . 10

1.5 Organization of Paper . . . 11

2 Physical Setting 11

2.1 Hardware System . . . 11

2.2 Software System . . . 12

2.3 Closed Loop System . . . 13

2.4 Example: The Juggle Φ_J . . . 14

2.4.1 *The Mirror Law, m_J* . . . 14

2.4.2 *Analysis, Simulation, and Empirical Exploration of $\mathcal{D}(\Phi_J)$.* . . . 15

2.5 The Complete Palette . . . 17

2.5.1 *Palming: Φ_P* . . . 17

2.5.2 *Catching:* Φ_C 18

3 Formal Idea: Safe Sequential Composition **18**

3.1 Sequential Composition 18

3.2 Obstacles and Safety 19

3.2.1 *Definition of Safety* 20

3.2.2 *Safe Deployments* 21

4 Implementation: Deploying the Palette **21**

4.1 Parameterization of Domains 21

4.1.1 $\mathcal{D}(\Phi_{J_0})$*: The Domain of* Φ_{J_0} 23

4.1.2 $\mathcal{D}_{PADDLE}(\Phi_{J_0})$*: Safety with respect to the Workspace Boundaries* 24

4.1.3 $\mathcal{D}_{BEAM}(\Phi_{J_0})$*: Safety with respect to the Beam* 24

4.2 Composition of Domains 26

4.2.1 *Deployment* 27

4.2.2 *A Hierarchy of Controllers* 29

4.3 Characteristics of the Test Runs 29

4.3.1 *A Typical Run* 30

4.3.2 *Descriptive Statistics* 31

4.4 Deployment of Invariant Sets 31

4.4.1 *Theoretically Correct Deployment* 32

4.4.2 *Adding "Safety Nets"* 32

4.4.3 *Testing the Limits* 33

5 Conclusions **34**

List of Figures

1 The Lyapunov Function as Funnel: Idealized graph of a positive definite function over its state space. 7

2 Ideal "funnel" capturing the entire "obstacle free" state space of the system. 8

3 Sequential composition of funnels: the goal point of each previous controller lies within the domain of attraction induced by the next. 9

4 A suitable composition of local funnels partitions (a close approximation to) the free space state space into cells assigned to a unique controller. 9

5 (a) The Bühgler Arm. (b) The horizontal workspace with obstacles. 11

6 Flow chart showing the various state machines in the system. 12

7 (a) The closed loop dynamics, f, induced by Φ. (b) The intermittency of the robot-environment interaction allows us to view the dynamics as a return map, with important implications vis à vis re-grasping (see footnote 1), as depicted in 14. 13

8 Empirically derived estimate of the horizontal juggling domain, $\mathcal{D}_H(\Phi_J)$. Note for the purposes of subsequent discussion that the symbol(s) "+" ("·") denote the points which failed (succeeded) in remaining within the robot's annular and visual workspace under the action of Φ_J. 16

9 Numerically derived estimate for the horizontally safe juggling domain. The symbol(s) "+" ("·") denote(s) the points which failed (succeeded) in remaining within the robot's annular and visual workspace under the action of the numerical simulation of Φ_J. 16

10 Sequential composition of controllers: Each controller is only active in the part of its domain not already covered by those nearer the goal. Here, $\Phi_1 \succeq \Phi_2 \succeq \Phi_3$, and Φ_3 is the goal controller. 20

11 Velocity "spines" – projections onto the zero-velocity plane of slices of $\mathcal{H}$ at regularly spaced velocities – of the invariant attracting domain of Φ_J. Shaded regions denote initial conditions that were successfully contained in the workspace while being brought to the goal via iterates of f_{Φ_J}. (a) Left Column: Varying initial $\dot{x}_i$ from negative (top) to positive (bottom) with $\dot{y}_i = 0$. (b) Right Column: Varying initial $\dot{y}_i$ from negative to positive with $\dot{x}_i = 0$. 22

12 Four 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). 23

13 Safe domain for Φ_{J_0} with the Beam inserted. Light gray areas represent successful initial states, darker areas show states that eventually hit the Beam. (a) Zero initial velocity. The appropriate slice of $\mathcal{D}_0$ has been added for comparison. (b) $\dot{x}_i = 1.5\frac{m}{s}$, showing preimages of the Beam. 25

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. 26

15 Juggle to Catch to Palm: (a) The Juggle drives all of its domain to $\mathcal{G}_J$, which lies within the domain for the Catch, but with non-zero vertical energy. (b) The Catch reduces the vertical energy of the ball, bringing it down into the domain for Palming. 27

16 A "theoretical" deployment using the jump controller of Figure 14. 28

17 (a) Cell (ellipse) number plotted against time for a typical run, with a trace of the ball's vertical position added for reference. The dots represent the mode at the moment of impact. (b) Horizontal projection of the same trace, superimposed on the deployment of Figure 16. The dots show the apex positions. 30

List of Tables

1 The full deployment, with controller types, goal points and domain types. All goal points have the same radial component, $\bar{\rho}$, so we display here only the angular component, $\bar{\phi}$. Ellipse numbers correspond to those in Figure 16. 28

2 Statistics characterizing three sets of 60 trials each. The first three rows give a breakdown for the successful runs of the total time into three segments: Throw to Pre-launch, time spent in Pre-launch, 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. 31

List of Footnotes

1. In general, re-grasping is required when the free configuration space attendant upon a specified grasp is disconnected, and the goal does not lie in the presently occupied connected component. As will be seen, in the present problem, the configuration space of the "palming" grasp disconnects the goal from the component where 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.

2. Moreover, we never "bother" to develop an explicit disturbance model even though we are specifically interested in operating in the empirical world with all its uncertainties and unmodelled phenomena. On the one hand, the theory and experience of building working controllers (Koditschek 1987, Whitcomb, Rizzi and Koditschek 1993) teaches us that the inevitable small but persistent disturbances arising from modeling errors, sensor inaccuracy and slight miscalibration are countered by the local structural stability properties of stable dynamical systems. On the other hand, large, arbitrary and unanticipated perturbations should be recoverable as long as they are relatively rare and do not violate the domain of attraction of every feedback policy at our disposal. In the end, of course, the only acceptable test of one or another model is to place the finished system in the physical setting for which it was designed.

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

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

5. In a previous series of papers with Rimon (Rimon and Koditschek 1992), the third author has addressed the design of such global funnels for purely geometric problems wherein the complicated portion of the domain is 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 Figure 11.

6. 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 $\mathcal{N}_{test}$.