Toward a Dynamical Pick and Place

Robert R. Burridge*, Alfred A. Rizzi†, and Daniel E. Koditschek‡

Artificial Intelligence Laboratory
University of Michigan
Department of Electrical Engineering and Computer Science

Abstract

We report on our initial efforts to build robot feedback controllers that develop increased capability from simpler constituent controllers. Previous work with our three degree of freedom robot has resulted in a machine that exhibits various dynamically dexterous skills of superlative ability but very narrow behavioral scope. We focus here on the development of both a formalism and practice for the composition of constituent controllers. The composite should yield automatically purposive combinations of these skills that reach goals no one of the defining controllers could have achieved in isolation. The specific task we initially target, the "dynamical pick and place", requires the robot to acquire balls that have been "randomly" thrown into its workspace and set them safely at rest in a specified location. We present here a brief overview of the constituent behaviors and a mechanism for their combination along with documentation of our preliminary empirical successes.

1 Introduction

We consider a sensor driven dynamical manipulation problem that seems analogous to the more familiar quasi-static Pick and Place. Past work in this area has resulted in an expanding family of working machines [14] that exhibit superlative dynamical dexterity in a narrow domain, as well as a growing body of theory to explain how [2, 13]. We hope by studying the present problem setting to both enlarge the domain of robot dexterity and extract from the algorithms that produce it a primitive but very robust sort of computational intelligence.

*Supported in part by the National Science Foundation under grant IRI-9123266 and in part by ARPA under grant B457.

†Supported in part by the National Science Foundation under grant IRI-9123266 and in part by the University of Michigan Center for Display Technology and Manufacturing.

‡Supported in part by the National Science Foundation under grant IRI-9123266.

1.1 Problem Statement

Our three degree of freedom robot is equipped with a flat paddle and a (60 Hz) stereo camera system (see [14, 13] for a complete description of this robot.). Its workspace will be cluttered with fixed obstacles — some suspended from the ceiling creating low "doors," and some protruding from the floor creating high "windows." The suspended obstacles will hang low enough that the paddle can only just pass through the doors when level with the ground. The protruding obstacles will be high enough that the robot can not carry the ball over them.

A ball will be thrown without warning into one of the free cells defined by the obstacles. The robot's task will be to acquire the ball, balance or bat it as required through the obstacles, and finally loft it into a destination receptacle. The work cell may be invaded by disturbances (we will poke at the ball with a stick as we do in the juggling work [13]) that take the ball far off its course and off the track of the sequence of maneuvers previously planned.

In this paper we outline our intended approach to this task and present preliminary results suggesting its feasibility. We have not yet implemented obstacles.

1.2 Background

Let b be the state of an environment and u the means by which a robot can change it according to the rule $b_{new} = f(b_{old}, u)$. Much work in robotics is concerned with developing plans, $u = \Pi(t; b_I)$, to bring b from a specified initial condition, b_I, to a desired final condition using time (t) as an explicit parameter. Such plans are often very sensitive to b_I, and rely strongly upon a predictive model for the world. Instead of introducing "exception handling" to overcome these difficulties, we are concerned with constructing time-independent feedback-driven autonomous systems where $u = \Phi(b)$.

Our focus on this problem is inspired in part by the

0-8186-7108-4/95 $4.00

HANDEY system developed by Lozano-Perez and colleagues [5], who emphasized the importance of developing task planning capabilities for situations where regrasping is necessary (although our problem is dynamical rather than quasi-static). Kak and colleagues [16] have created a quasi-static assembly environment in the tradition of HANDEY that builds a plan based upon sensed initial conditions. An execution process 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.

Research into reasoning about the interplay between sensing and recovery at the event level has been considerably stimulated by advent of the Ramadge-Wonham DES control paradigm [11]. Lyons [10, 9] proposes a formalism for encoding and reasoning about the construction of action plans and their sensor-based execution, but explicitly avoids problems wherein geometric sensor reports must be used to estimate progress and thereby stimulate the appropriate event transitions.

Despite the many dissimilarities in problem domain, models and representation, our work seems to relate most closely to the "fine-motion planning with uncertainty" literature in robotics (for example, as expounded in Latombe [7]) originated by Lozano-Perez and colleagues [8]. Indeed, our emphasis on robustness and error recovery as the driving consideration in robot task planning derives from the "LMT" framework and their original insights on fine motion planning. In their work, 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 funneling [4] action of sensorless compliance to rigid objects is replaced by the stability of general dynamical systems, but we borrow heavily from the notion of pre-image back-chaining, as will be seen in Section 4.

Traditionally this literature focuses on quasi-static problems, with control actions restricted to piecewise constant velocity vectors. 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 a closed loop policy, Φ. Moreover, we never develop an explicit disturbance model. Our experience building working controllers [6] teaches us that disturbances arising from modeling, sensor and calibration errors are countered by the local structural stability properties of stable dynamical systems. We also desire the system to recover from large, arbitrary and unanticipated perturbations (as long as they are relatively rare).

Although the problems explored here address higher-level issues of task execution, there is a strong relation to previous work in dexterous manipulation (in particular "robot juggling") performed in our laboratory. We believe that these ideas can be extended to build a variety of useful dexterous machines similarly single-minded in their pursuit of the task behavior and ability to surmount unanticipated obstacles along the way [14, 1, 13].

2 Setup

In the problem with which we are concerned, there is a robot, which we can control directly, and an environment (the ball in our experiments), which can only be manipulated through contact with the robot. The task is to devise a strategy for the robot that drives the environment to a goal state, or set of states.

2.1 Definitions and Notation

Let $b \in \mathcal{B} \approx \mathbb{R}^6$ be the full state of the ball in Cartesian coordinates. Let $r \in \mathcal{R} \approx \mathbb{R}^3 \times \mathbb{R}^3$ be the state of the robot in joint space.

2.1.1 Ball Flight

The ball in flight will be modeled by the Newtonian dynamics with gravity pointing along the z-axis with magnitude $-g$. Due to the simplicity of the ball flight dynamics, we can derive a closed form expression for the ball position at time t in the future as a function of present state: $b(t) = F^t(b)$. When the ball and paddle collide we use the standard restitution model of collisions (See Synge and Griffith [15] for a discussion of restitution models). In short, we assume that only the ball's velocity component normal to the paddle is affected, while neither the tangential component nor the robot is altered by impact.

Unless the ball and robot are in continuous contact, it is natural to divide the trajectory of the ball into epochs of time punctuated by collisions. The k^{th} epoch starts with the ball in state b_k, and ends immediately after the next impact, in state b_{k+1}. The motion of the robot during the k^{th} epoch will be denoted $r_k(t)$. The duration of the k^{th} epoch will be τ_k.

2.1.2 Controllers and Their Induced Return Maps

Since the robot has no effect on the ball except at contact, we will ignore from now on the trajectory of the actuator system, and only consider the state of the robot at the next impact. We are interested in robot strategies which are entirely dependent on the state of the ball, so we will denote the state of the robot immediately prior to the next impact by the shorthand $u_k = \Phi(b_k)$, and use the collision law to determine the induced effect on the ball.

Given the time to impact, τ_k, the flight model, $F^t(b_k)$, and the collision model, $C(b, r)$, we can now express the "return map"for the post-impact state of the ball:

$$b_{k+1} = C(F^{\tau_k}(b_k), r_k(F^{\tau_k}(b_k))) := f_\Phi(b_k). \quad (1)$$

Suppose there is an attracting set[1], $\mathcal{G}$, arising from iteration of f_Φ. In our methodology, the specific goal of a controller, Φ, is encoded by $\mathcal{G}$, and it follows that the domain of attraction of Φ to $\mathcal{G}$ is given by

$$\mathcal{D}_\Phi(\mathcal{G}) = \{b \in \mathcal{B} | f_\Phi{}^\infty(b) \in \mathcal{G}\}. \quad (2)$$

2.2 The Experimental Apparatus

All of the experimental work described in this paper has been implemented on the Bühgler robot described in [13, 14]. This machine senses ball positions using 2 CCD cameras located above and outside the workspace, and senses impacts using a microphone attached to the paddle. Although space limitations prevent an exhaustive list of modifications to the setup of [13, 14], we will note that both a window manager and dynamical observers were modified to allow the ball to be thrown into the workspace rather than carefully presented.

The state estimates from the sensor system are fed through a nonlinear transformation, $M(b)$, to arrive at a desired reference trajectory for the robot, r_k. This is in turn passed through a smoothing "follow-through" generator, and then to a robot controller. The controllers we use are chosen from the class of inverse dynamics controllers constructed by Whitcomb [17].

Throughout our work, we assume that the observer has correctly converged on the true ball state, and that the robot is accurately tracking the reference trajectory by the time of impact. Our lab experience consistently supports this assumption.

[1] A closed, invariant set $\mathcal{G}$ is attracting if it has the property that there exists an open neighborhood, $\mathcal{N}(\mathcal{G}) \supseteq \mathcal{G}$, such that $f_\Phi{}^\infty(\mathcal{N}) \subseteq \mathcal{G}$.

3 The Constituent Controllers

The nonlinear transformation, M, above, follows the traditions of Buehler [2], and we will refer to it as a "mirror law" accordingly.

3.1 Mirror Laws as Fundamental Modes

For the purposes of this work, we have added several new mirror-style reference laws, M, to the original, M_J, which results in *juggling* - Φ_J. These result in *catching* - Φ_C, *palming* - Φ_P, *tossing* - Φ_T, and *placing* - Φ_K. In this paper, we only discuss juggling, catching and palming as the other behaviors are still in their infancy.

Juggling, Φ_J; The underlying mirror law, M_J, used to construct our juggling behavior is exactly that used in [13].

Catching, Φ_C; We have constructed a preliminary catching behavior based directly on M_J by choosing a set-point which represents an extremely low juggle (on the order of $10cm$). This results in the ball's vertical energy being quickly dissipated while its lateral position is still well regulated.

Palming, Φ_P; The machine has been endowed with this capability since its inception [13]. Implementation is accomplished by again re-working the template for M_J. In this case gains are adjusted such that this hitting portion of the pitch law has been removed.

3.2 Implementation

There is little yet that can be said analytically about these different controllers and their domains of attraction or ranges of acceptable "set points." The one degree of freedom juggling case has been analyzed successfully by Buehler and Koditschek [1], and Rizzi and Koditschek [12] have made progress toward the two and three dimensional cases. Lacking formal results for either the domains of attraction or viable ranges of set-points, we are nonetheless encouraged by the empirical results of [3] and believe that useful conservative estimates of these characteristics can be derived either computationally or experimentally.

In [3] we display statistical data demonstrating that the juggling and palming behaviors will robustly drive the ball to goal points located throughout large regions of the workspace. Once there, the regulation about the

fixed points is good, and the repeatability of average location (e.g., of apex) is very good.

4 A Manipulation Sequence

This section presents initial work on autonomous "sequencing" or "chaining" of the behaviors of Section 3 in order to accomplish a higher-level goal. In section 4.2, we demonstrate our initial attempt at implementing the ideas described below.

Presuming that we have a "task goal" and a finite set of behaviors, we wish to develop a directed graph based on the idea of *preparation* introduced in 4.1. Next we find a behavior whose goal set coincides with the task goal (for now, we assume that this is possible). Using a recursive algorithm similar to preimage backchaining [8], we then set up a partition of the ball's state space by moving away from the goal node of the graph in a breadth-first manner and choosing the appropriate subset of the domain of attraction of each behavior encountered.

4.1 Preparation Graphs and Composite Controller Design

Say that behavior Φ_1 *prepares* behavior Φ_2 if the goal set of Φ_1 lies within the domain of attraction of Φ_2. This relation may be symmetric or transitive, but need not be. For any set of behaviors, $\mathcal{S} = \{\Phi_1, \Phi_2, \ldots, \Phi_N\}$, there is a directed (possibly cyclic) graph, $\mathsf{G}_\mathcal{S}$ induced by the *prepares* relation. The nodes of $\mathsf{G}_\mathcal{S}$ represent the elements of $\mathcal{S}$, while the links represent the *prepares* relation.

If the overall task goal set ($\mathcal{G}$) coincides with the goal set of Φ_i, for some $i < N$, then by starting with Φ_i and recursively tracing the *prepares* relation backwards through the graph, we can find $\mathcal{S}_G \subseteq \mathcal{S}$, the set of all behaviors from whose domains the goal is achievable were the appropriate controllers applied at the correct times.

After finding Φ_i, next choose a partial ordering on $\mathsf{G}_\mathcal{S}$ which transforms it into an acyclic graph with all paths leading 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 $\bar{\mathcal{D}}_{\Phi_i} = \mathcal{D}_{\Phi_i}$, and $\mathcal{D}_{\mathcal{S}_G}(1) = \mathcal{D}_{\Phi_i}$.
2. Append to the back of the OPEN-LIST the list of all behaviors which *prepare* the first element, and have not previously been placed on the list.
3. Remove the first element of the OPEN-LIST.
4. For Φ_j, the new first element of OPEN-LIST, let $\bar{\mathcal{D}}_{\Phi_j} = \mathcal{D}_{\Phi_j} - \mathcal{D}_{\mathcal{S}_G}$, and let $\mathcal{D}_{\mathcal{S}_G}(N+1) = \mathcal{D}_{\mathcal{S}_G}(N) \cup \mathcal{D}_{\Phi_j}$.
5. Repeat 2,3 and 4 until OPEN-LIST is empty.

At the end of this process, the regions $\bar{\mathcal{D}}_{\Phi_l}$ 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 behavior Φ_j exactly when the ball state lies within $\bar{\mathcal{D}}_{\Phi_j}$.

The domain of attraction for the goal set can now be considered to be not just $\mathcal{D}_{\Phi_i}$, but $\mathcal{D}_{\mathcal{S}_G} = \bigcup_{\Phi \in \mathcal{S}_G} \mathcal{D}_\Phi$. In this sense, the composite closed loop behavior can be thought of as arising from a new controller, $\Phi_{\mathcal{S}_G}$.

The technique proposed here is a variant of the preimage backchaining idea introduced by Lozano-Perez, Mason, and Taylor [8], but we choose to substitute sensory events for the physical transitions that characterized their control sequences.

4.2 An Instance: A Simple Pick and Place Task

Now consider the specific task of bringing the ball to rest on the paddle at a specified location from "randomly" thrown initial ball states. The behavioral repertoire is limited to the three controllers, Φ_J, Φ_C, and Φ_P discussed above. In the parameter space that represents their defining mirror laws, M_J, M_C, and M_P respectively, we have "hand tuned" three particular settings. Call such a choice of settings a "deployment."

For any deployment, Φ, we require the further information, $\mathcal{D}_\Phi$ and $\mathcal{G}_\Phi$, in order to apply the procedure of section 4.1. In this initial work, we obtain this information from empirical observation as follows.

Palm $\mathcal{D}_P$ and $\mathcal{G}_P$: The goal point, $\mathcal{G}_P$, is precisely the task goal point. The domain, $\mathcal{D}_P$, is the set of all ball states with low vertical energy ($\eta(b) < K_1$).

Catch $\mathcal{D}_C$ and $\mathcal{G}_C$: The goal point, $\mathcal{G}_C$, is also the task goal point. The domain, $\mathcal{D}_C$, is the set of all ball states with low horizontal position errors ($\phi(b) < K_2$), and low velocity errors ($\kappa(b) < K_3$), as well a bounded vertical energy ($\eta(b) < K_4$).

Juggle $\mathcal{D}_J$ and $\mathcal{G}_J$: The goal point, $\mathcal{G}_J$, is located 0.8m above the task goal point. The domain, $\mathcal{D}_J$, is

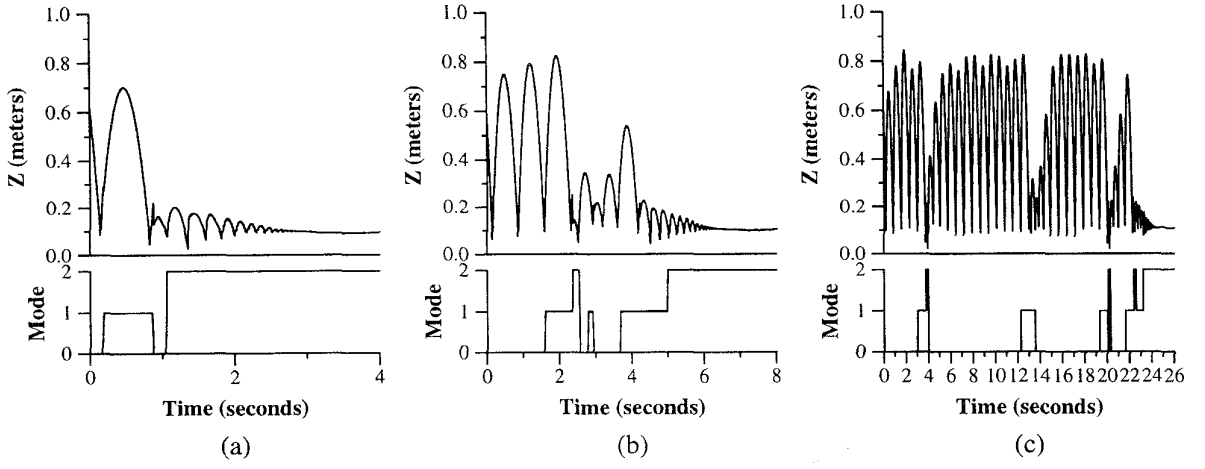

Figure 1: Behavior switching. (a) A quick example. (b) Several mode switches. (c) four attempts before success.

assumed to be the entire state space, as we used no other behaviors.

We now report on our initial empirical experience with the composite feedback controller just constructed.

In figure 1, we show three traces of the vertical position of the ball as a function of time, coupled with the behavioral mode of the system. In the mode traces, 0 represents juggling, 1 catching, and 2 is the palming mode.

In figure 1a, the ball is dropped into the workspace from above the goal point. As a result, the only criterion keeping the system in juggling mode is the energy maximum for the catch. Immediately after the first collision, the energy is low enough for the system to try to catch the ball. At the second collision, however, the observer predicts the post-impact conditions incorrectly, which sends the system back to juggling (due to lateral error, not shown). The observer quickly corrects to the actual ball trajectory, which has low enough energy to send the system straight to palming, where the ball quickly loses its remaining energy.

In figure 1b, the ball is not introduced so nicely, and the robot juggles it a few times before the lateral errors are small enough for an attempted catch. This time, the observer takes too long to notice the second bounce, predicting that the ball has fallen further than it did in reality. Thus, $\eta(b)$ falls low enough for palming to be switched on, but new ball data quickly send the system back to juggling. The lateral errors briefly dip low enough for the system to switch to catching, then back to juggling, but finally catching turns on, and then palming.

In figure 1c, the system tries to catch three times and fails before finally bringing the ball down to the paddle on the fourth attempt. Once the system is in palming mode and the ball has settled on the paddle, it will remain there.

The third example shown here demonstrates the robustness of this approach to planning. The system has no memory, and no "plan" in the traditional sense. It will jump from any node to any other based on the sensed ball state, regardless of how or whether those nodes are connected in $G_{\mathcal{S}}$. A traditional planner might need pre-programmed exception handling to go from palming mode back to catching or juggling, as such jumps violate the world model. The dynamical stability of each of our behaviors guarantees that the system will continue to be inexorably drawn toward the goal state regardless of any unexpected changes in state.

For a more complex workspace, the selection of the set points, gains, etc. for the various behaviors will be more difficult. We have begun to look at the problem of automatic "deployment," where the system autonomously chooses where to place the set points so as to produce a favorable $G_{\mathcal{S}}$.

5 Conclusion: Looking Ahead to Dynamical Obstacle Avoidance

We are developing versions of each of our behaviors that will insure that the ball does not penetrate an obstacle. Since these "safe" controllers are not yet complete at the time of this writing, we have no em-

pirical results to report. In [3] we outline our notion of *safety* that provides the key ingredient in their design.

If one re-reads this paper and substitutes for each Φ the safe version, Φ_S and substitutes for D_Φ the induced restriction dynamics $f_\Phi|_{\mathcal{D}_{\Phi S}}$, then all the same definitions and algorithmic procedures are applicable. Clearly, however, the domains, $D_{\Phi S}$, can be expected to be much smaller, and the corresponding problem of deployment much more difficult.

In the near future we plan to consider the "obstacle" of all those ball states that can never again be batted because they are out of reach, or unalterably becoming so. We will attempt to build a "containing" version of the sequenced manipulation of section 4. In this scenario, a ball would be thrown into the workspace and the robot would first insure that it is brought "under control" so as not to fly out of reach before settling into its descent toward the palming goal. Moreover, the robot would be capable of containing balls that we try to poke out of its workspace.

More realistic obstacles will raise even more interesting versions of the deployment problem. We hope to work on doors and windows soon.

References

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

[2] Martin Bühler. *Robotic Tasks with Intermittent Dynamics.* PhD thesis, Yale University, New Haven, CT, May 1990.

[3] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Dynamical pick and place. Technical Report CSE-TR-235-95, University of Michigan, Ann Arbor, MI, 48105, April 1995.

[4] M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE J. Rob. and Aut.*, 4(4):635–642, Aug 1988.

[5] Tomas Lozano-Pérez et al. Handey: A robot systems that recognizes, plans, and manipulates. In *Proceedings IEEE Int. Conf. Robotics and Aut.*, pages 843–849, 1987.

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

[7] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer, Boston, MA, 1991.

[8] Tomás Lozano-Perez, Matthew T. Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–23, 1984.

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

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

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

[12] A. A. Rizzi and D. E. Koditschek. Further progress in robot juggling: Solvable mirror laws. In *Int. Conf. Rob. and Aut.*, pages 2935–2940, 1994.

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

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

[15] J. L. Synge and B. A. Griffith. *Principles of Mechanics.* McGraw Hill, London, 1959.

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

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