A One Degree of Freedom Juggler in a Two Degree of Freedom Environment

M. Bühler, D. E. Koditschek, and P. J. Kindlmann [1]

Center for Systems Science
Yale University, Department of Electrical Engineering

Abstract

We develop a formalism for describing and analyzing a very simple representative of a class of robotic tasks which require "dynamical dexterity", among them the task of juggling. We introduce and report on our preliminary empirical experience with a new class of control algorithms for this task domain that we call "mirror algorithms".

1 Introduction

We are interested in robotic task domains involving dynamical environments. In particular, we feel that a number of novel control problems arise in the context of coupling computer controlled mechanical manipulators to dynamically active environments whose input structure changes in response to the robot's actions. In this paper we consider a simple representative from a range of robotic tasks associated with dexterous capabilities that might be grouped under the general rubric of "juggling". We understand this term to include those tasks requiring throwing and catching, or (as in this paper) beating and batting, or any other interaction with an object (or multiple objects) which would otherwise fall freely in the earth's gravitational field. Such tasks share the property of presenting non-trivial dynamical environments whose characteristics change intermittently subject to excitation from the robot. It seems fair to say that the only systematic work in this realm to date has been the pioneering research of Raibert [3].

In a previous paper [1], we developed a formalism for describing and analyzing a particular task called the "vertical one-juggle". This involves controlling the trajectory of a single body — a "puck" — constrained to lie on a ("frictionless") plane turned into the earth's gravitational field by repeated impacts with a bar actuated at a revolute joint — a one degree of freedom "robot" — in such a fashion that the puck, regardless of its initial position and velocity, eventually attains a stable periodic orbit passing through some arbitrarily specified apex point on the juggling plane. In that paper, we undertook merely to characterize the "environmental control problem." Namely, adopting the phenomenological point of view of the puck, we examined the controllability and stabilizability of the fourth order dynamical system resulting from its response to the two possible robot inputs: time and velocity of impact. The intent was to understand the consequences of any logically possible impact sequence. This abstract characterization completely ignored the fact that the robot is itself a dynamical control system. Left unexamined were the questions of whether a particular schedule of impacts could be achieved by a real robot, and how the robot might be best be commanded to achieve that schedule.

In this paper we present our preliminary empirical experience with a computer controlled one degree of freedom revolute robot and frictionless puck on a near-vertical plane. We first attempt to implement an environmental control policy which obtains from a local linear analysis of the abstract environmental control problem. Impact schedules resulting from this policy, while provably correct (they achieve local stablity around any desired vertical one-juggle), are seen to lack robustness (in a sense to be made precise below), and no robot control procedure that we have employed to date to implement such a schedule succeeds even in keeping the puck aloft, much less in effecting a stable vertical one-juggle. We next introduce a family of surfaces in the six dimensional space resulting from the cross product of the second order robot phase space with the fourth order puck phase space. Trajectories in the puck phase space, when "lifted" onto this surface and then projected down to the robot phase space, yield explicit robot reference trajectories. The lifted and projected trajectories have an intuitively appealing character which leads us to name this procedure the *mirror algorithm*. When our robot is forced toward this surface, that is, when we force it to track the trajectory specified by a mirror algorithm applied to a free falling puck, the result is a successful vertical one-juggle. We present data from a variety of experiments which demonstrate this success. We defer to a paper presently in preparation the proof that the mirror algorithm results in a correct environmental control law.

The paper is organized as follows. In the next section we provide a description of the experimental apparatus in question, and develop a simplified mathematical model of the robot-puck dynamical interaction. In Section 3 we review the earlier results presented in [1] which formalize the notion of a "vertical one-juggle" and prove that it is achievable. Finally, in Section 4 we describe the results of our experiments to date.

[1]This work has been supported in part by PMI Corporation, GMF Robotics Corporation, Inmos Corporation and the the National Science Foundation under grant DMC-8552851, a Presidential Young Investigator Award held by the second author.

IEEE Catalog Number 88TH0234-5

2 The Empirical and Analytical Setting

2.1 Experimental Apparatus

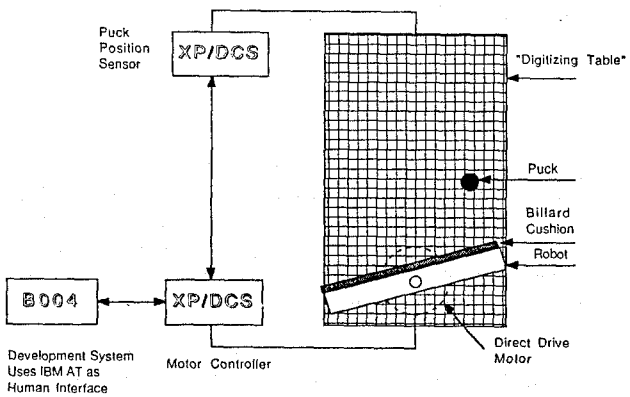

Figure 1: The Yale Juggler

The physical apparatus consists of a puck, which slides on an inclined plane and is batted successively by a simple "robot": a bar with billiard cushion rotating in the juggling plane as depicted in figure 1. All intelligent sensor and controller functions are performed by a three node distributed computational network formed from the Inmos transputer based Yale XP/DCS control node [2]. The Inmos T800 is a 32 bit 10 MIPs RISC chip which includes an onboard floating point unit capable of 1 Mflop performance. Each transputer provides four independent 5, 10 or 20 MHz serial DMA channels, so any node can communicate with up to four neighbors while simultaneously executing its own program with no affect upon either computational or communications rate (after initial start up overhead). Our XP/DCS CPU board complements the Transputer's modular and flexible character by providing fast external memory, support for the four serial communication links, two fiber optic links, and an I/O expansion connector. The board's backplane connector is pin compatible with the INMOS ITEM Development System. The plug-in I/O board enhances the Transputer's computational and communication power with a bidirectional latched 32 bit I/O bus with full handshaking support. Half of this board is allotted to a wire-wrap prototyping area allowing for easy customization to specific I/O needs. The cost of each mother/daughter board set at the time of writing is slightly over $2000.

In order to move the bar according to some puck dependent control algorithm, the puck's position and velocity in both directions on the plane must be measured. Presently, this is accomplished by placing an oscillator inside the puck and burying a grid in the juggling plane, thus imitating a big digitizing tablet. On the back of the plane, a simplified T8-based XP/DCS node is used as a smart sensor. It measures the voltages induced in the loops by the puck. The puck position in the plane is computed from the zero and first order moments. This information is used to estimate the puck's state: we use a standard linear observer to reduce measurement noise in position and velocity data. Each puck state measurement is communicated asynchronously via fiber optics to the Motor Controller Node. This sampling and communication process is performed at a rate of 1kHz. In the near future we intend to introduce a vision system in order to handle several pucks, or move off the plane into three space. We are fairly confident that the attendant decrease in sampling rate will not affect the experimental results significantly.

The main control node is the Motor Controller, dedicated to commanding a high torque dc servo actuator donated to the robotics laboratory by the PMI Corporation. This Yale XP/DCS node receives puck information from the sensor node, reports logging data to the logging node, and implements the control algorithm. Specifically, this involves interfacing to the motor, performing noise filtering, detecting puck states (up,top,down,impact), computing the desired motor position and velocity for the next impact, implementing a control strategy to achieve that, and predicting the puck states. Of course, numerous safety checks and considerable general housekeeping must be performed as well. The sampling time of this process is about 600 μsec.

For debugging and documentation, data about the puck as well as the motor are logged online to a third node on a standard INMOS B004 board in the IBM/AT.

The experiences with the XP/DCS, the transputer and the development environment derived from this application are very encouraging. No single number can capture the ease of use and the little time spent with system overhead. Given the T800's intrinsic floating point capability, and the mathematical function library, formulas were programmed (in OCCAM, the C-like native compiler) almost directly from the blackboard with no attempt at code optimization. In spite of substantial calculations, and a great deal of data logging and error handling overhead, very high sampling rates were achieved. The system operates capably in a high EMI environment in consequence of the low cost 5MHz fiber optic units from Hewlett Packard built into the Yale XP/DCS boards.

2.2 A Simplified Mathematical Model

In the following discussion, refer to the sketch of the system given in Figure 2. Let p_j denote the ball position at impact; $\dot{p}_j$, the ball velocity just before; and $\dot{p}'_j$, the velocity just after impact, all as seen from frame $\mathcal{F}_0$ in the robot's base. As announced above, we ignore the robot's dynamics: actuator positions and velocities are assumed to be specified arbitrarily and are not at all influenced by the interaction with the ball. This derivation is based on the following additional simplifying assumptions.

First, we assume here that all interactions between ball and robot during impact can be modelled adequately as an instantaneous event. Namely, the velocity component of the ball perpendicular to the bar, after impact j is given as in [4],

$$ {}^1\dot{z}'_j = -\alpha\, {}^1\dot{z}_j + (1+\alpha)u_{2,j} \qquad (1) $$

where ${}^1\dot{z}'_j$ is the perpendicular velocity component of the ball after impact in the $\mathcal{F}_1$ coordinate frame, $\alpha \in (0,1)$ is the "coefficient of restitution", and $u_{2,j} = \|p_j\|\dot{\theta}$ is the linear velocity of the robot at the impact point p_j. Moreover, it is assumed that the puck's velocity component parallel to the robot bar are unchanged by the impact. Finally, spin effects and friction during flight are neglected as well.

Under these conditions, the velocity of the ball after impact in the $\mathcal{F}_1$ coordinate frame with origin at p_j is

$$ {}^1\dot{p}'_j = \begin{bmatrix} 1 & 0 \\ 0 & -\alpha \end{bmatrix} {}^1\dot{p}_j + \begin{bmatrix} 0 \\ 1+\alpha \end{bmatrix} u_{2,j} = A\, {}^1\dot{p}_j + b' u_{2,j}. \qquad (2) $$

To transform this back into $\mathcal{F}_0$ we use

$$R_j = \begin{bmatrix} \cos\theta_j & -\sin\theta_j \\ \sin\theta_j & \cos\theta_j \end{bmatrix}.$$

While $\theta_j = \arctan\frac{z_j}{x_j}$ is a function of the ball position at the moment of impact, we assume that the robot velocity is independent of the ball velocity, thus

$$^1p_j = R_j^T p_j,$$

and, (2) may be written as

$$\dot{p}_j' = M_j\dot{p}_j + b_j u_{2,j}, \tag{3}$$

where

$$M_j \triangleq R_j A R_j^T; \qquad b_j \triangleq R_j b'.$$

Denote the space of control inputs as $\mathcal{U} \triangleq \mathbb{R}^2$, the state space as $\mathcal{W} = \mathbb{R}^4$ and the complete state of the ball as

$$w = \begin{bmatrix} p \\ \dot{p} \end{bmatrix} = \begin{bmatrix} x \\ z \\ \dot{x} \\ \dot{z} \end{bmatrix}.$$

This defines a discrete dynamical system,

$$\begin{aligned} w_{j+1} &= f(w_j, u_j) = \begin{bmatrix} p_j + \dot{p}_j' u_{1,j} - a u_{1,j}^2 \\ \dot{p}_j' - 2a u_{1,j} \end{bmatrix} \\ &= \begin{bmatrix} p_j + (M_j\dot{p}_j + b_j u_{2,j}) u_{1,j} - a u_{1,j}^2 \\ M_j\dot{p}_j + b_j u_{2,j} - 2a u_{1,j} \end{bmatrix} \end{aligned} \tag{4}$$

with $a = \begin{bmatrix} 0 \\ \frac{1}{2}g \end{bmatrix}$.

The quantity $u_{1,j}$ denotes the time elapsed between impact j and (j+1). Evidently, this interval uniquely specifies θ_{j+1}, the angle of the robot's joint at the next impact: we may regard $u_{1,j}$ as a robot control input to the environment. Recall, that $u_{2,j}$ is the linear velocity of the robot at the instant of impact at point p_j — a second control input. Thus, the robot acts upon the environment at intermittent times and positions which are a function of their mutual dynamical interaction.

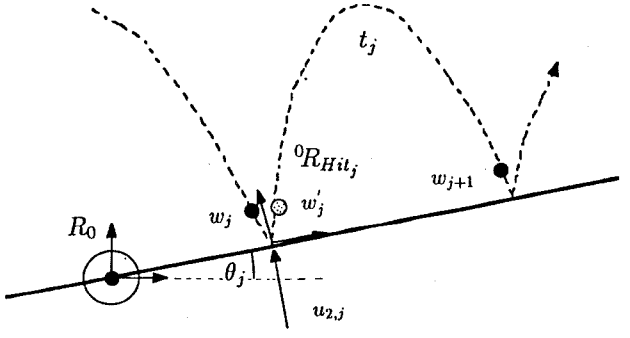

Figure 2: The Impact Event

3 The Environmental Control Problem

In this section we review the results presented in our earlier paper [1]. In Section 3.1 we provide a formal definition of the task, and in Section 3.2 we show that the task is achievable.

3.1 The Vertical One-Juggle

Probably the simplest systematic behavior of this environment imaginable (after the rest position), is a periodic vertical motion of the puck in its plane. Specifically, we would like to be able to specify an arbitrary "apex" point in the juggling plane, and from arbitrary initial puck conditions, force the puck to attain a purely vertical periodic trajectory which passes through that apex point at its zenith.

Since a purely vertical trajectory requires zero horizontal velocity, $\dot{x} = 0$, and a fixed vertical impact velocity, $\dot{z}^*$, from a specified impact height, $z = 0$, implies a specified zenith position, we are led to the following definition. Let the task subspace of the vertical one-juggle be the plane

$$\mathcal{T} \triangleq span\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\} = \{w \in \mathcal{W} : \dot{x} = 0,\ z = 0\}.$$

We will say that a feedback law, $g : \mathcal{W} \to \mathcal{U}$, constitutes a *vertical one-juggle* with respect to the task, $w^* \in \mathcal{T}$, if w^* is a fixed point of the closed loop system,

$$w^* = f_g(w^*); \qquad f_g(w) \triangleq f(w, g(w)),$$

and is a stable attractor of the resulting discrete dynamics.

Proposition 3.1 *Given the discrete dynamical control system, (4), and a point, $w^* \in \mathcal{W}$, there exists a feedback law, $g : \mathcal{W} \to \mathcal{U}$ such that w^* is a fixed point of the closed loop map,*

$$w^* = f_g(w^*),$$

if and only if

$$\begin{aligned} &(i) \quad w^* \in \mathcal{T}; \\ &(ii) \quad g(w^*) = u^* \triangleq \begin{bmatrix} -2/g \\ -\frac{1-\alpha}{1+\alpha} \end{bmatrix} \dot{z}^*. \end{aligned}$$

Proof: Consider the fixed point condition

$$w = f(w, g(w)) = \begin{bmatrix} p + \dot{p}' u_1 - a u_1^2 \\ \dot{p}' - 2a u_1 \end{bmatrix}$$

Elimination of $\dot{p}'$ directly results in

$$\begin{aligned} &(1) \quad \dot{x}^* = 0 \\ &(2) \quad u_1^*(w^*) = -\tfrac{2}{g}\dot{z}^*. \end{aligned}$$

With these conditions and

$$\dot{p}' = M\dot{p} + b u_2 = a u_1$$

we obtain two more conditions

$$\begin{aligned} &(3) \quad z^* = 0 \\ &(4) \quad u_2^*(w^*) = -\tfrac{1-\alpha}{1+\alpha}\dot{z}^* \end{aligned}$$

where conditions (1),(3)[(2),(4)] are equivalent to (i)[(ii)].

This result shows, on the one hand, that only a point in $\mathcal{T}$ may be fixed by feedback, and, on the other hand, that an appropriate constant, u^* may be found to fix *any* point of $\mathcal{T}$. We remark, in passing, that the same constant offset, u^* fixes *every* point on a line in the task plane, hence by itself, could not possibly stabilize any particular point on that line (here, stabilize is being used to denote the property of attractivity as well as stability).

3.2 Local Stabilizability of the Task Plane

We next observe that the system is locally controllable at any point in the task set. Since this implies local stablilizability — that is, the existence of an affine feedback law which arbitrarily places the poles of the linearized closed loop system — it serves to demonstrate that the vertical one-juggle as defined in Section 3.1 is logically achievable.

Proposition 3.2 *If*

$$w^* \in \mathcal{T} - 0,$$

and g fixes w^*,

$$f(w^*, g(w^*)) = w^*,$$

then system (4) is locally controllable at $(w^*, g(w^*))$.

Proof: According to Proposition 3.1 , g fixes w^* if and only if $g(w^*) = u^*$. Thus the system is locally controllable if and only if

$$\begin{aligned} A &= [D_w f](w^*, g(w^*)) = [D_w f](w^*, u^*) \\ B &= [D_u f](w^*, g(w^*)) = [D_u f](w^*, u^*) \end{aligned}$$

comprise a completely controllable pair.

Taking partial derivatives of (4) gives

$$A = \begin{bmatrix} 1 & 2v\frac{\dot{z}^{*2}}{x^*} & v\dot{z}^* & 0 \\ 0 & 1 & 0 & -\alpha v\dot{z}^* \\ 0 & 2\frac{\dot{z}^*}{x^*} & 1 & 0 \\ 0 & 0 & 0 & -\alpha \end{bmatrix}; \quad B = \begin{bmatrix} 0 & 0 \\ \dot{z}^* & (1+\alpha)v\dot{z}^* \\ 0 & 0 \\ -g & 1+\alpha \end{bmatrix}$$

where v denotes the constant

$$v = -\frac{2}{g}.$$

It suffices to show that four of the eight columns of the matrix (B, AB, A^2B, A^3B) are linearly independent. The four columns (B, A^2B) we consider are

$$\begin{bmatrix} 0 & 0 & \frac{4\dot{z}^{*3}}{gx^*}(2\alpha - 3) & \frac{8\dot{z}^{*3}}{g^2x^*}(1+\alpha)(3-\alpha) \\ \dot{z}^* & -\frac{2}{g}\dot{z}^*(1+\alpha) & \dot{z}^*(2\alpha^2 - 2\alpha + 1) & -\frac{2}{g}\dot{z}^*(1-\alpha+\alpha^2) \\ 0 & 0 & 4\frac{\dot{z}^2}{x^*}(1-\alpha) & -\frac{4\dot{z}^{*2}}{gx^*}(1+\alpha)(2-\alpha) \\ -g & 1+\alpha & -\alpha^2 g & \alpha^2(1+\alpha) \end{bmatrix}.$$

The determinant of this matrix,

$$\frac{16}{g^2}\frac{\dot{z}^{*6}}{x^{*2}}\alpha(1+\alpha)^2$$

is nonzero for any $w^* \in \mathcal{T} - 0$.

□

According to linear control theory, if (A, B) is a completely controllable pair then for any desired set of poles whose complex elements appear in conjugate pairs,

$$\Lambda = \{\lambda_i\}_{i=1}^4 \subset \mathbf{C}$$

there exists a matrix, $K_\Lambda \in \mathbb{R}^{2\times 4}$ such that the closed loop spectrum achieves that set,

$$spectrum\,(A + BK_\Lambda) = \Lambda.$$

Now suppose that the feedback algorithm, g, is chosen to be

$$g(w) \triangleq u^* + K_\Lambda(w - w^*). \tag{5}$$

Since

$$\begin{aligned} [D_w f_g](w^*) &= [D_w f](w^*, g(w^*)) + [D_u f](w^*, g(w^*))[D_w g](w^*) \\ &= A + BK_\Lambda, \end{aligned}$$

it follows that any K_Λ for which $\Lambda \subset \mathcal{D}^1 \subset \mathbf{C}$(the open unit disk in the complex plain) yields a feedback law, g, which achieves the vertical one-juggle as defined in Section 3.1.

A few remarks are now in order. The local nature of our task definition — asymptotic stability of a fixed point — afforded an easy proof that it may actually be attained. Local notions have played an historically predominant role in systems theory for just this reason. Indeed, decades worth of successful activity in the theory and practice of linear control are a testament to the suitability of this point of view. However, as is well known, fixed parameter local conditions may be deceptively reassuring.

To begin with, the determination that a particular equilibrium state is asymptotically stable provides very little help in estimating the domain of attraction and the domain of "containment" around that state. The former is comprised of those initial conditions which tend asymptotically toward the equilibrium state, while the latter is comprised of those initial conditions which are guaranteed to remain in some specified neighborhood of the equilibrium state. Since our juggling plane, in contrast to its analytical model developed in Section 2.2, is not truly infinite, achieving a particular containment region will be crucial to the success of any real juggling strategy, regardless of its domain of attraction.

If it is stable, trajectories are assured to be contained in a sufficiently small neighborhood of an equilibrium state by the guarantee that they originate in some still smaller neighborhood. The relationship between the region of origin and region of containment may be conservatively estimated by recourse to a Lyapunov function. However, a constructive proceedure for determining Lyapunov functions is available in general only for linear systems. Thus, if the feedback law is determined using local linear techniques, estimating the domain of containment may be conveniently accomplished only in a conservatively small neighborhood of the equilibrium state which may be of little practical use.

A second important and well known property of asymptotically stable linear systems is that of structural stability. As long as the closed loop linear system is "hyperbolic" — in this case, as long as its eigenvalues are not on the unit circle in $\mathbf{C}$— sufficiently small perturbations in its parameters will not cause a loss of stability. Yet structural stability, a necessary attribute of any physically manifested process, is not yet sufficient to ensure a physically prescribable process: synthesis requires robustness. A robust process admits a uniform measure of perturbation over a set of data points affording a means of determining just how small "small deviations" from the nominal may be. Since the parameters of the ultimate closed loop system with spectrum Λ are chosen according to a numerical procedure which finds a matrix K_Λ, as a function of the pair A, B, a controllable but *ill conditioned* pair may result in large departures of the closed loop parameters from their desired values either because the numerical procedure is very sensitive or because of small departures in implementation from the numerically determined value of K_Λ. Thus, pole placement in the face of ill conditioned data is not robust.

4 Robot Implementation

The preceding analysis was intended as an abstract justification of the possibility of achieving the vertical one-juggle by addressing the effect of arbitrary impact sequences upon the robot's environment. In this section we describe our preliminary efforts to implement a successful vertical one-juggle with the physical apparatus sketched in Section 2.1. First, in Section 4.1, we describe the failure of our efforts to turn the discrete impact rule (5) into a practicable juggling algorithm. Then, in Section 4.2, we introduce a new synthesis procedure for realizing a vertical one-juggle which explicitly takes the robot control problem into account. This procedure accomplishes the specified task: we provide data from representative runs of successful vertical one-juggles.

4.1 Impact Schedules Resulting from the Linearized Discrete Dynamics

For a variety of task points, $w^* \in \mathcal{T}$, we chose a variety of spectra, Λ, in the open unit disk, $\mathcal{D}^1$, determined K_Λ using a numerical procedure, and determined an ideal impact schedule according to the affine feedback law, g, described by (5). We then induced our robot to deliver a close approximation of this impact schedule via an ad hoc procedure described below. In no case of this series of experiments did we observe a successful vertical one-juggle. Even the best runs ended with the puck striking the boundaries of the juggling plane after three successive impacts. Our explanation for this failure now follows.

First, numerical simulation showed that the domain of containment of the idealized closed loop discrete dynamical system resulting from (5) was unacceptably small. For certain arbitrarily chosen values of w^*, pole assignments, Λ, could be found that resulted in a large domain of attraction. However, we could find no pole assignments for physically realizable settings of w^* which achieved a domain of attraction whose diameter along the $\dot{z}$ axis of $\mathcal{W}$ was greater than 6 % of the value $\dot{z}^*$. In all of these simulations, the trajectories within the domain of attraction leave the physical boundaries of actual juggling plane. For typical settings, the domain of containment within the juggling plane was no larger than 2 % of the desired fixed point magnitude. This is smaller than the error tolerance of the puck position sensing sensing system. We conclude that the linearized analysis of even this simplified discrete nonlinear system is inadequate to the desired task. In formal terms, the very definition of the task in 3.1 may be too weak.

Second, for all values of w^* examined, numerical tests revealed that the pair (A, B), while completely controllable, was poorly conditioned. We conclude that our experimental gain settings were not sufficiently close approximations of K_Λ even to guarantee stability, much less containment.

Of course, the key to success or failure of even a practicable feedback scheme stemming from the discrete analysis would depend upon the details of how the robot is commanded to implement the impact schedule required by (5). For, as we have previously described, the analysis of the environmental control problem in Section 3, provides an abstract characterization of what is logically possible assuming that the robot is a perfect "feedback agent". It remains entirely silent concerning the manner in which a particular impact schedule is achieved. Several different implementation procedures were attempted; the best performance seemed to result from the following procedure:

1. Since the feedback algorithm (5) is based upon the states of the puck just before impact, a "start-up" procedure at time $j = 1$ is required. For simplicity, we start from the desired apex point, and assume that the first impact occurs at $\hat{w}_1 = w^*$: the robot is commanded to hit the puck with a velocity obtained from applying the feedback law (5) to the estimated impact state: $u_{j=1} = g(\hat{w}_1)$. Of course, in this case, $g(w^*)$ yields $u_{2,j=1} = u_2^*$ given in Proposition 3.1.
2. Just before the actual first impact, we measure and estimate the true state of the puck before impact, w_1, and evaluate the feedback law, $g(w_1)$ to get a desired time inverval to next hit, $\hat{u}_{1,j=1}$. Now we use the dynamical model (4) to predict $\hat{w}_2$ with u_2 set to the actual measured impact velocity, and u_1 set at the nominal value, $\hat{u}_{1,j=1}$.
3. The desired velocity of the next impact, $u_{2,j=2}$, is determined by applying the feedback law to the predicted next impact point, $g(\hat{w}_2)$. Now $\hat{w}_2$ prescribes the robot's angle, θ_2, at second impact, and, together with $u_{2,j=2}$, prescribes the robot's angular velocity, $\dot{\theta}_2$ at second impact.
4. To implement the desired robot states for the second impact, we use a simple open loop fixed torque control strategy which works as follows. Acceleration from a rest position to $\dot{\theta}_2$ takes time t_{acc}. Together with θ_2, this yields the robot's rest position. During the puck flight, its states are predicted t_{acc} in advance. When the predicted position crosses θ_2, the robot starts accelerating until the second impact occurs. Measurements made after the fact show that this strategy yields satisfactory accuracy.
5. The procedure continues using the measured estimate of the forthcoming impact, w_j, to obtain a value for $u_{1,j}$ from $g(w_j)$, (5), and a predicted next impact state, $\hat{w}_{j+1}$, from the model (4), to generate a value for $u_{2,j+1}$ from $g(\hat{w}_{j+1})$.

Neither this strategy nor any of its variations resulted in a viable vertical one-juggle. Even had the linearized analysis resulted in a robust feedback law with a physically realizable region of containment we now list some of the reasons for our feeling that it fails to provide a satisfactory synthetic framework in the present task domain.

An initially attractive feature of the discrete analysis is that it confirms the intuition that only state information at impact should be required for a successful juggling algorithm — that full trajectory information is redundant. Conceptual appeal notwithstanding, in reality state information at or very near the impact event is exceedingly difficult to measure. Our experimental apparatus, for example, derives estimates of the state at impact by recourse to full state information during flight. The discrete analysis is "blind" to such realities: it does not "tell" the designer how best to use the continuous data.

Just as they necessarily rely upon the accuracy of the impact state measurement, algorithms resulting from the discrete analysis critically depend upon the accuracy of the impact parameters and idealizations of the model — the coefficient of restitution; the validity of the zero friction assumption; a zero puck diameter — because this information is explicitly used to compute the next desired impact states and the next robot inputs. These features do not contribute to a practicable scheme. It is necessary, for example, to correct for the error between the next predicted and the actual impact position. How to do that, is left to the designer.

Finally, and obviously, as pointed out in the very beginning of the paper, this algorithm only specifies desired robot control inputs to the environment at the moment of impact. It is completely up to the designer to solve the robot control problem, and this leads to the kind of ad hoc implementations exemplified by steps 1 - 5 above. We seek, instead, some automatic procedure for generating successful and provably correct robotic juggling behavior.

4.2 The Mirror Algorithm and its Implementation

We now introduce a procedure which meets the goals stated in the previous paragraph. To appreciate the intuitive origins of the new algorithm, consider first the problem of a prismatic one degree of freedom robot in a one degree of freedom environment. That is, there is a single puck (of unit mass) constrained to fall in only the vertical direction, and a piston which moves up and down to stike it precisely. If it is desired to bring the puck to a periodic orbit (allowing only instantaneous contact to preclude simply resting the puck on a fixed piston) whose zenith is z_d, then note that the total energy of the puck,

$$\eta = \frac{1}{2}\dot{z} + gz,$$

when it attains this zenith at zero velocity is

$$\eta_d = gz_d.$$

Supposing that r measures the height of the robot, consider a robot trajectory determined by the following rule

$$r = -\{\kappa_0 + \kappa_1[\eta_d - \eta(z,\dot{z})]\}z = -\kappa(w)z. \tag{6}$$

The reader may note that an impact occurs either in the case of exact tracking $r(t) \equiv z(t)$, or otherwise, only when both the robot and the puck achieve zero height. In the latter case, the robot describes a distorted "mirror" reflection of the puck's trajectory. The reader may note as well that since we neglect friction during the puck's flight, we may assume that $\dot{\eta} \equiv 0$, hence,

$$\dot{r} = -\kappa(w)\dot{z}.$$

Now passing to the discrete impact system, it is now not hard to show that with

$$\kappa_0 \triangleq \frac{1-\alpha}{1+\alpha},$$

we satisfy the fixed point condition of Proposition 3.1 (in the appropriately smaller dimensional phase space). Further analysis reveals the appropriate choice of κ_1 to stabilize this fixed point. Thus, we have a procedure which makes explicit use of the full puck trajectory, completely specifies the robot's behavior, and results in a provably correct vertical one-juggle, when the robot and environment have the same "cartesian" degrees of freedom.

We now describe the manner in which this idea "scales" to the case of a one degree of freedom revolute robot operating in the two degree of freedom cartesian environment presented in Section 3. Consider a scalar valued function,

$$\varphi : \mathcal{P} \times \mathcal{W} \to \mathbb{R},$$

on the phase space of the two degree of freedom puck, $\mathcal{W}$, and a one degree of freedom robot, whose angular position and velocity we denote by $p = (\theta, \dot{\theta})$. In particular, define

$$\varphi(p,w) \triangleq \theta + \kappa(w) \cdot atan\left(\frac{z}{x}\right) - \kappa_2(x - x^*),$$

where $\kappa(w)$ was defined in (6). Notice that $\dot{\theta}$ is free in φ, hence, denoting the configuration space of the robot as $\mathcal{C}$, so that $\mathcal{P} \triangleq T\mathcal{C} = \mathcal{C} \times \mathbb{R}$, we may think of it as a scalar valued map on $\mathcal{C} \times \mathcal{W}$. Since $\partial\varphi/\partial\theta \neq 0$, the zero level,

$$\mathcal{S} \triangleq \varphi^{-1}[0] = \{(\theta, w) \in \mathcal{C} \times \mathcal{W} : \varphi(\theta, w) = 0\},$$

defines a smooth co-dimension one surface in $\mathcal{C} \times \mathcal{W}$ according to the implicit function theorem. Similarly, the "impact surface", the zero level of

$$\psi(p,w) \triangleq \theta - atan(z/x),$$

defines another smooth co-dimension one surface in $\mathcal{I} \triangleq \psi^{-1}[0] \subset \mathcal{C} \times \mathcal{W}$. Note that $\mathcal{I}$ intersects $\mathcal{S}$ transversely, since $grad\,\varphi$ is linearly independent of $grad\,\psi$, thus, $\mathcal{S} \cap \mathcal{I}$ is a smooth three-dimensional manifold in $\mathcal{C} \times \mathcal{W}$, again using the implicit function theorem applied to the $\mathbb{R}^2$ valued map, $f \triangleq (\varphi, \psi)$. In particular, this manifold may be parametrized by $(x, \dot{x}, \dot{z})$, as is seen by solving $f = 0$. This reasoning shows that the discrete dynamical system on $(x, \dot{x}, z, \dot{z})$ of Section 3 induces a lower dimensional discrete dynamical system on $(x, \dot{x}, \dot{z})$ as long as any robot trajectory is assumed to lie in $\mathcal{S}$. We defer to a paper presently in preparation the proof that this family of dynamical systems may be stabilized around point in the task plane $w^* \in \mathcal{T} - \{0\}$ by a suitable adjustment of the gains $\kappa_0, \kappa_1, \kappa_2$.

It is evident any trajectories in $\mathcal{W}$ generate trajectories in $\mathcal{S}$, and, therefore, in $\mathcal{P}$ according to the rule

$$\theta = -\kappa(w) \cdot atan\left(\frac{z}{x}\right) + \kappa_2(x - x^*).$$

Thus, having chosen the feedback gains, κ_0, to fix a desired task point, w^*, and κ_1, κ_2 to make it an attractor, according to the reasoning sketched above, the actual robot implementation is immediate. We simply force our robot to track the desired trajectory using standard PD techniques. This procedure results in empirical success as illustrated by the sample plots of actual runs presented in the figures below.

In all of the figures, the units are in inches (rescaled — made skinnier to fit on the conference mats) from the motor axis which is perpendicular to the plane. In each run, the desired apex point was set at $x = 11$ and $z = 31$ inches, the puck was dropped from an arbitrary initial position (marked on the plot), and roughly 10 impacts were recorded. The juggler functions reliably for many more impacts (as the video demonstration accompanying this paper shows) but there is a problem displaying the continuous trajectories clearly on the plots. Figure 3 shows a run from an initial condition to the right and considerably below the apex point. Figure 4 shows a run from an initial condition to the left and considerably below the apex point. Figures 5 and 6 show runs from initial conditions higher from both left and right than the apex point.

Acknowledgements

We would like to acknowledge the INMOS Corporation, GMF Robotics Corporation, and PMI Corporation whose support, in conjunction with an NSF Presidential Young Investigator Award held by the second author, has made this research possible.

References

[1] M. Bühler and D. E. Koditschek. A prelude to juggling. In *Conference Presentation: 26th IEEE Conference on Decision and Control*, pages (paper available from authors — not in proceedings), Los Angeles, CA., Dec. 1987.

[2] F. Levin, M. Bühler, and D. E. Koditschek. The Yale Real-Time Distributed Control Node. In *Oregon State University Conference on Parallel Processing*, page , Oregon State University, Portland, Ore., Apr 1988.

[3] Marc H. Raibert. *Legged Robots That Balance*. MIT Press, Cambridge, MA, 1986.

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

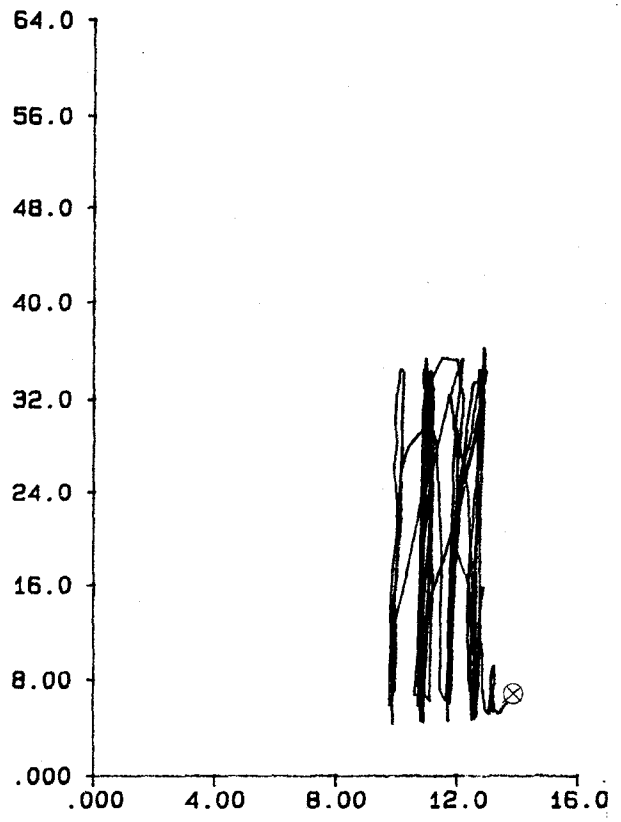

Figure 3: Initial Condition Low and to the Right

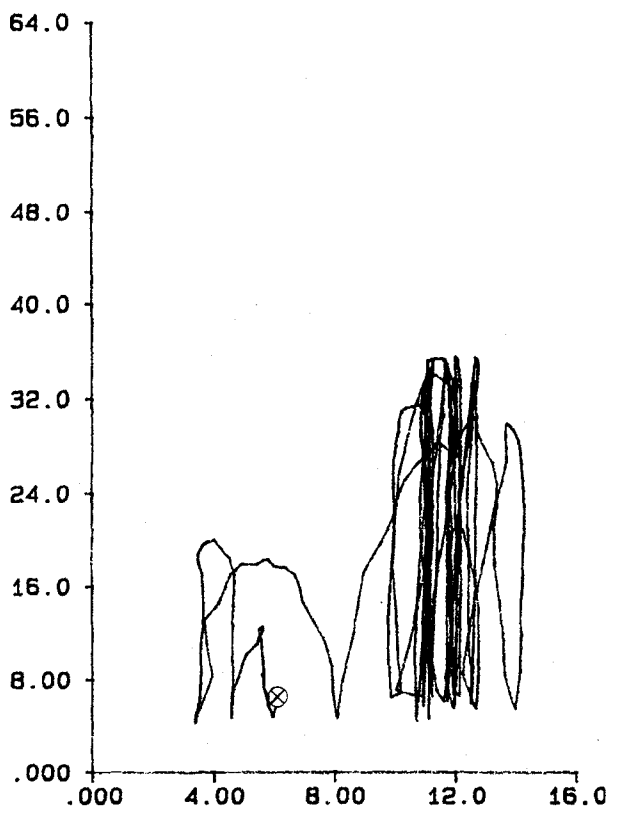

Figure 4: Initial Condition Low and to the Left

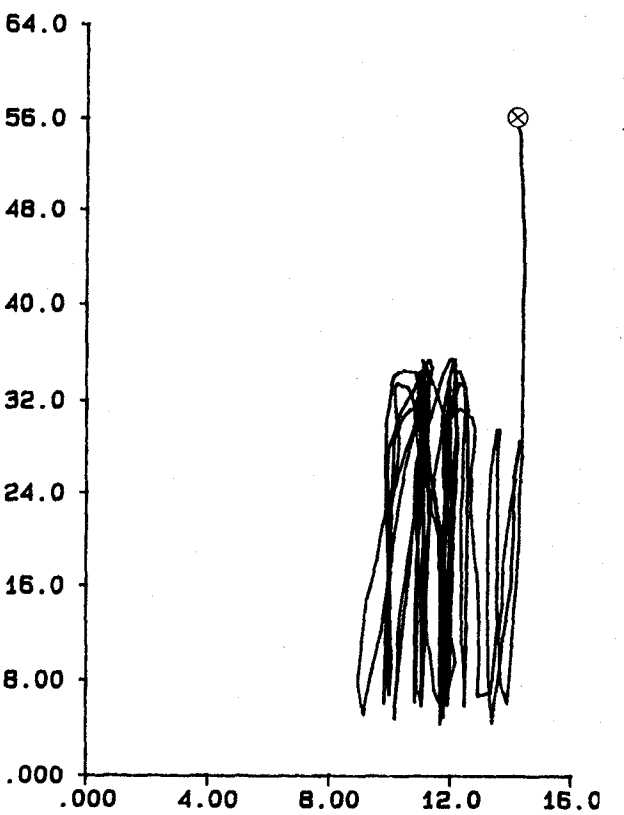

Figure 5: Initial Condition High and to the Right

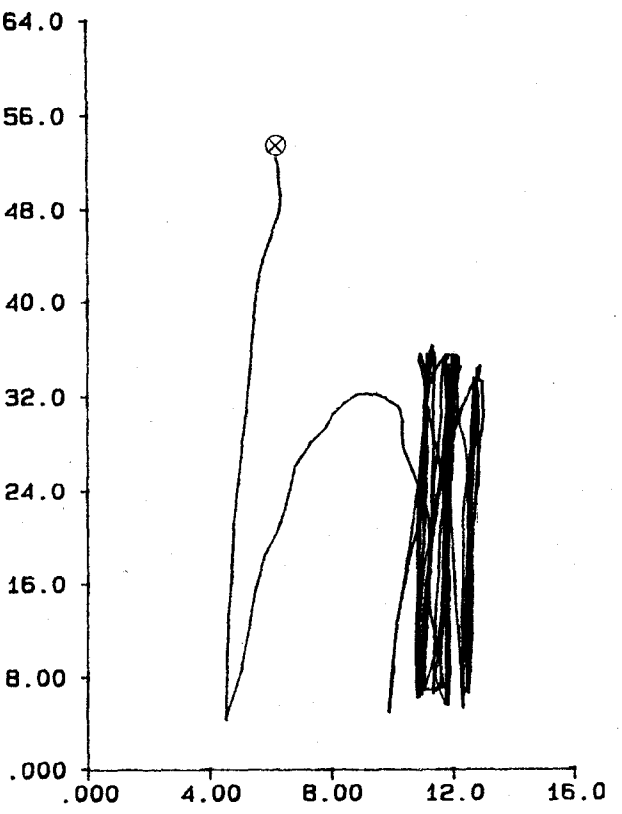

Figure 6: Initial Condition High and to the Left