A Dynamical Sensor for Robot Juggling

A chapter invited for inclusion in the
World Scientific Press Volume
Visual Servoing - Automatic Control of Mechanical Systems with Visual Sensors

A. A. Rizzi [1] and D. E. Koditschek [2]

Center for Systems Science
Yale University, Department of Electrical Engineering

November 4, 1992

[1]Supported in part by IBM through a Manufacturing Graduate Fellowship and in part by the National Science Foundation under grant IRI-9123266.

[2]Supported by the National Science Foundation in part through a Presidential Young Investigator Award and in part under grant IRI-9123266.

Abstract

We discuss the sensory management strategy that has evolved over the course of our efforts to build a three degree of freedom robot capable of juggling two balls at once. A field rate stereo camera system passes estimates of the balls' positions to a juggling algorithm that drives the robot's joint actuators. In order to meet the stringent real-time constraints imposed by such a visual servoing task, we have found it necessary to pay increasingly careful attention to our strategy for controlling the dynamics of the camera window management system. The paper concludes with an intitial attempt to formalize this control problem.

1 Introduction

This paper describes our recent efforts to enhance a real-time stereo vision system built as a "dynamical sensor" for our three degree of freedom Bühgler Arm [13]. The robot resulting from pairing our revised vision system with this arm has recently achieved a long-targeted milestone in our laboratory: the spatial two-juggle — the ability to bat two freely falling balls into independent stable periodic vertical orbits by repeated impacts with the arm [10]. There is no mechanically active component in this vision system. Nevertheless, we have found that there is already enough room for thought in getting its management right as to presage a wealth of important and novel problems that might be termed the "control of attention." Such problems are sure to arise in the planning and control of more general dynamically dexterous machines. The present paper is intended as an exploration of how to pose and think about such problems in a particularly simple case.

The term "dynamical sensor" is intended to convey our relatively narrow interest in programmable camera systems as a means of closing loops that result in the manipulation of the physical world. Such sensors must be dynamical in two different ways. First, the computational model of the environment to be sensed will stress the geometry of its dynamics rather than the geometry of its shape. Second, and perhaps more fundamentally, the computational strategy used in deployment will stress the connection between the sensor's dynamical role in a feedback loop and the quality of its measurements. Dynamical sensor management becomes a control problem. For example, since there are hard real-time constraints to meet, the management of resources must stress considerations of update and latency over mere throughput. Moreover, the process of extracting a little information from a lot of data is driven toward the minimum that will suffice for the task at hand rather than striving for the most that might logically be had. Finally, when previously sensed data mediates the collection of new information, a stability problem may result.

Seen from a different perspective, we present here a case history of how to coax greater performance out of a mechanical plant by improvements in its sensory processing capabilities. All machines are built in the face of conflicting design constraints that limit ultimate performance. It has seemed particularly fruitful to explore these limitations in robotics research since, almost by definition, a robot is a machine that should be capable of use in unforseen applications. Indeed, a great many researchers have sought to understand how improvements in control algorithms and sensor interpretation strategies can enhance the capabilities of existing robots. Conversely, a growing number of investigators has begun to examine how the design and implementation of mechanical components can result in sophisticated performance even in the absence of active controls and sensing [8, 9, 14]. We consider such explorations of the tradeoffs between design constraints and performance limitations to be fundamental in robotics research. We are further convinced that the algorithmic tinkering documented here is worthy of the reader's consideration since it arose in the course of our efforts to achieve a real task with a physical machine.

The robot resulting from pairing our original vision system with this arm had allowed a straightforward extension to the spatial setting of the ideas developed by Bühler *et al.* for a planar juggler [4, 7]. We found that a slight generalization of Bühler's "mirror law" resulted in a stable spatial one-juggle — batting a single ball into a specified periodic vertical trajectory — as capable of withstanding significant unexpected perturbations as its planar forebear [12]. However, in the effort to explore how well these ideas generalized to the two-juggle we found it necessary to devote a considerably greater amount of thought to managing the camera system than originally foreseen. None of the particular constraints that limit our specific apparatus can be considered fundamental in themselves. In comparison to the constituents of our robot there are commercially available at present more rigid structural members, actuators possessed of greater torque-to-mass ratios, higher capacity computation, and so on. A substitution of any of these would have significantly mitigated the difficulties described below, and may likely have directly enabled successful two-juggling without appeal to the more considered sensing strategies we will discuss. However, performance limits being inescapable, such substitutions merely postpone the underlying problems raised by the need to build systems whose capabilities exceed that of their parts. We seek in this paper to engage those underlying problems.

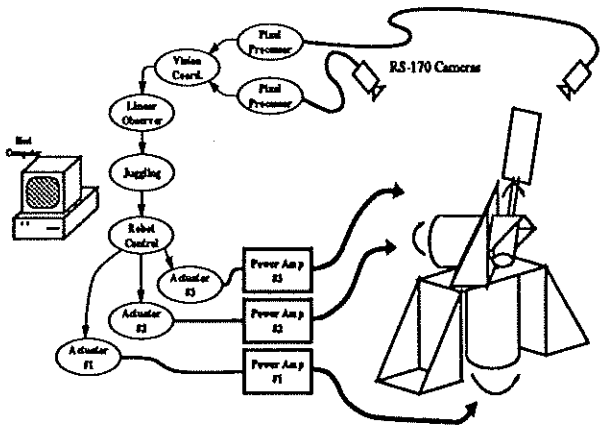

Figure 1: The Yale Spatial Juggler

The paper is organized as follows. We review the overall architecture and control strategy of this juggling system in Section 2. In Section 3 we trace the emergence of the dynamical sensing problem in the the recent history of our experiments with this apparatus and offer a summary of the intuitive solutions we have implemented relatively successfully to date. In Section 4 we sketch our present understanding of how to pose these problems more formally in a manner that might shed some theoretical light on why our intuitive schemes work and how they might fail. The paper concludes with a brief assessment of the gap between our current laboratory practice and theoretical understanding.

2 System Overview

This section presents an overview of our previous version of the juggling robot system [12]. We review how the juggling algorithms and control methodology Bühler originally introduced for the planar system [4] may be extended in a straightforward manner to the present apparatus. We spend a little more time comparing the distinctions in sensory environments presented by the two different settings and explain how the spatial case is a good deal more complicated.

2.1 The Geometry and Dynamics of Juggling

Let $\mathcal{B}$ and $\mathcal{Q}$ denote the configuration space of the body and robot respectively. Bühler *et al.* [4] proposed a novel strategy for implicitly commanding a robot to "juggle" by forcing it to track a reference trajectory generated by a distorted reflection of the ball's continuous trajectory. This policy, the recourse to "mirror laws," may be represented as a map $m : T\mathcal{B} \rightarrow \mathcal{Q}$, so that the robot's reference trajectory,

$$r(t) = m(b(t), \dot{b}(t)), \tag{1}$$

is determined by the geometry of the graph of m and the dynamics of ball, $b(t)$. We now review these two phenomena.

2.1.1 Trajectory Generation via Ball Dynamics

The two properties of the ball's motion relevant to juggling are its flight dynamics (behavior while away from the paddle), and its impact dynamics (how it interacts with the paddle/robot). Since this paper

concerns the problem of building a flight sensor, we will simply refer to earlier publications for a discussion of the impact model and its implications for juggling [11].

For simplicity we have chosen to model the ball's flight dynamics as a point mass under the influence of gravity. This gives rise to the flight model

$$\ddot{b} = \tilde{a} \tag{2}$$

where $b \in \mathcal{B} = \mathbb{R}^3$, and $\tilde{a} = (0, 0, -\gamma)^T$ is the acceleration vector experienced by the ball due to gravity. A position-time-sampled measurement of this dynamical system will be described by the discrete dynamics,

$$\begin{aligned} w_{j+1} &= F^s(w_j) \triangleq A_s w_j + a_s; \\ & \qquad A_s \triangleq \begin{bmatrix} I & sI \\ 0 & I \end{bmatrix}; \quad a_s \triangleq \begin{bmatrix} \frac{1}{2}s^2\tilde{a} \\ s\tilde{a} \end{bmatrix} \\ b_j &= Cw_j; \qquad C = [I, 0], \end{aligned} \tag{3}$$

where s denotes the sampling period and $w_j \in T\mathcal{B}$.

The ball's motion, $(b, \dot{b})(t)$, generates a robot reference trajectory, through the mirror law discussed above (1). This reference trajectory (along with the induced velocity and acceleration signals) can then be directly passed to a robot joint controller. [1] In following the prescribed joint space trajectory, the robot's paddle pursues a trajectory in space that periodically intersects that of the ball. The impacts induced at these intersections result in the desired juggling behavior.

2.1.2 Geometric Programming I: The Juggling Algorithm

It must be emphasized that the function, m (1) we will present here comprises at once a mathematical description of our algorithm and its actual implementation. Implementing "geometric programs" of this type amounts to merely placing the particular transformation law — in the present case, (6) or (7) — in the juggling block of the data flow path depicted in the left side of Figure 1. One immediate practical benefit of this arrangement is the availability of very powerful high level development environments in the form of commercial symbolic manipulation packages. We build these functions in Mathematica on a SPARCstation and use the automatically generated C code on the target controller.

A "vertical one-juggle" is achieved when the robot arms bats a single ball into a periodic vertical trajectory whose horizontal position and vertical height have been specified in advance. The one-juggle algorithm used in the present work is a reasonably straightforward extension of Bühler's planar "mirror law" [4] to the spatial case. The kinematics of the machine, depicted in Figure 2, lead to a forward map of the form

$$p(\tilde{q}) = \begin{bmatrix} -(S_1 d_2) + (C_1 C_2 C_3 - S_1 S_3) d_g + C_1 S_2 s_2 \\ C_1 d_2 + (C_2 C_3 S_1 + C_1 S_3) d_g + S_1 S_2 s_2 \\ -(C_3 S_2 d_g) + C_2 s_2 \\ 1 \end{bmatrix}.$$

[1] In the case of a one degree of freedom arm we found that a simple PD controller worked quite effectively [3]. In the present setting, we have found it necessary to introduce a nonlinear inverse dynamics based controller [15]. The high performance properties of this controller notwithstanding, our present success in achieving a spatial two-juggle has required some additional "smoothing" of the output of the mirror law described in a companion article [10].

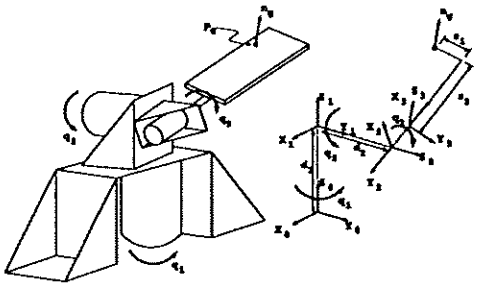

Figure 2: The Bühgler arm (left) and it's kinematics (right).

We begin by using the inverse kinematics of the robot arm, [2]

$$\begin{bmatrix} \phi_b \\ \theta_b \\ \psi_b \\ 0 \\ s_b \end{bmatrix} \triangleq p^{-1}(b) = \begin{bmatrix} \operatorname{atan}(\frac{-b_2}{-b_1}) + \operatorname{asin}(\frac{\sin(q_3)d_2}{\sqrt{b_1^2+b_2^2}}) \\ -\frac{\pi}{2} \quad + \quad \operatorname{atan}(\frac{b_3}{\sqrt{b_1^2+b_2^2-\sin(q_3)d_2^2}}) - \\ \operatorname{asin}(\frac{\cos(q_3)d_2}{\sqrt{b^Tb-\sin^2(q_3)d_2^2}}) \\ q_3 \\ 0 \\ \sqrt{b^Tb - \sin^2(q_3)d_2^2 - \cos^2(q_3)d_2^2} \end{bmatrix}, \tag{4}$$

to define what is effectively an expression of the "ball in joint space coordinates." We now seek to express formulaically a robot strategy that causes the paddle to respond to the motions of the ball in four ways:

(i) $q_{d1} = \phi_b$ causes the paddle to track under the ball at all times.

(ii) The paddle "mirrors" the vertical motion of the ball through the action of θ_b on q_{d2} as expressed by the original planar mirror law [4].

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

To this end, define the ball's *vertical energy* and *radial distance* as

$$\eta \triangleq \gamma b_z + \frac{1}{2}\dot{b}_z^2 \quad \text{and,} \quad \rho_b \triangleq \sin(\theta_b)s_b \tag{5}$$

respectively. The complete mirror law combines these two measures with a set point description ($\bar{\eta}$, $\bar{\rho}$,

[2] The paddle's surface provides what is effectively an additional two degrees of unactuated freedom. Since we will only require a total of four degrees of freedom to adjust the paddle's normal at the point of impact with a ball, its width is left unused by setting $s_1 = 0$.

and $\bar{\phi}$) to form the function

$$\mathrm{m(w)} \triangleq \left[\begin{array}{c} \overbrace{\dot{\phi}_b}^{\text{(i)}} \\ \underbrace{-\frac{\pi}{2} - (\kappa_0 + \kappa_1(\eta - \bar{\eta}))\left(\theta_b + \frac{\pi}{2}\right)}_{\text{(ii)}} + \\ \underbrace{\kappa_{00}(\rho_b - \bar{\rho}_b) + \kappa_{01}\dot{\rho}_b}_{\text{(iii)}} \\ \underbrace{\kappa_{10}(\phi_b - \bar{\phi}_b) + \kappa_{11}\dot{\phi}_b}_{\text{(iv)}} \end{array} \right]. \tag{6}$$

2.1.3 Geometric Programming II: The Two-Juggle

A two-juggle task requires that the robot perform two simultaneous one-juggle tasks with two independent balls separated in both space and time. Separation in space avoids ball-ball collisions (not currently part of the environmental model) and separation in time (the two balls should not fall simultaneously) is necessary to ensure that the machine is capable of striking one ball and moving into position under the second, prior to the first falling to the floor. Apparently there is an obstacle present in the phase space of the system which we are attempting to "avoid". Thus the juggling algorithm must be able to control the phase relationship between the two balls in addition to the three new variables associated with the position and energy of the additional ball.

Individual mirror laws for the two balls, constructed as in [3], are combined to form the overall two-juggle law by the use of a scalar valued "analytic switch" $s \in [0, 1]$,

$$m_{II}(w_0, w_1) = s(w_0, w_1)m_0(w_0, w_1) + (1 - s(w_0, w_1))\, m_1(w_1, w_0). \tag{7}$$

The function s weights the relative importance of tracking the mirror due to one or the other ball and is designed to achieve the value zero or unity depending upon whether, respectively, ball 1 or ball 0 requires the most urgent attention as follows.

The *phase* of a ball $\epsilon(w) \triangleq -\frac{\dot{b}_3}{\sqrt{2\eta}}$ is a function varying over the interval $[-1, 1]$ that passes through zero at the apex, reaches 1 at the "bottom", and flips sign upon beginning the ascent portion of its orbit. An *urgency* function, σ, is defined for each ball using a set of splines which map ball phase into the closed interval $[0, 1]$. Figure 3(a) shows the urgency function currently used: it is set at one for a short period before impact, returns to zero quickly after impact and rises to one again shortly after the ball passes its apex.

Finally, s is given by

$$s = \frac{1 - \sigma(w_1)}{2 - \sigma(w_0) - \sigma(w_1)}, \tag{8}$$

and is designed to map appropriately the urgencies of both balls (the unit box) into the unit interval as depicted in Figure 3(b).

In summary, the phase, ϵ, varies smoothly from -1 immediately after impact, to 0 at the balls apex, to 1 immediately prior to impact. The urgency σ reaches unity when the ball is near impact as measured by ϵ, and 0 as it rises to its apex. Finally s combines two *urgencies* by smoothly mapping the unit box onto $[0, 1]$ so that $s = 0$ when $\sigma_1 = 1$, and $s = 1$ when $\sigma_0 = 1$.

2.2 The Geometry and Dynamics of a Motion Sensor

Central to this juggling strategy is a sensory system capable of "keeping it's eyes on the ball." We require that the vision system produce a 1 KHz signal containing estimates of the ball's spatial position and

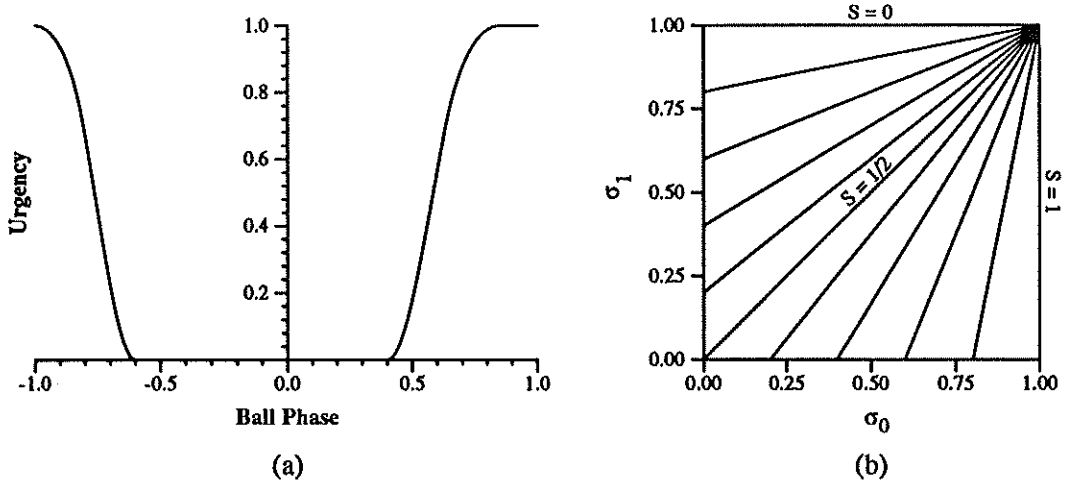

Figure 3: (a) *Urgency* of a ball as a function of ball phase. (b) Contour plot of the mapping from ball urgencies to s

velocity (six measurements). Denote this "robot reference rate" by the symbol $\tau_r = 10^{-3} sec.$ Two RS-170 CCD television cameras constitute the "eyes" of the juggling system and deliver a frame consisting of a pair of fields at 60 Hz, so that a new field of data is available every $\tau_f = 16.\overline{6} \cdot 10^{-3} sec.$ The CYCLOPS vision system, described in [11, 5], provides the hardware platform upon which the data in these fields are used to form the input signal to the mirror law, (6). The remainder of this section describes how this is done.

2.2.1 Previous Experience with a Planar Juggler

In the planar apparatus that originated the juggling algorithms described above [4] the sensory system was almost an afterthought. The robot's reliable nature was almost entirely due to the robustness of the juggling algorithm itself. Sensor readings — voltages induced by active puck signals upon a 64 x 34 "pixel" wire grid — had very little cost. In the case of one falling body, a centroid could be taken over the entire (byte valued) image in less than 3.5 msec. In the case of two falling bodies, a distinct signature was put on the active signals, and two separate full image first order moments could be computed in less than 7 msec. Although the wire grid had only 1 inch resolution, estimates from the linear flight dynamics observer achieved .1 inch accuracy in steady state. Thus, no amount of bumping or jogging short of knocking the puck physically off the juggling plane could confuse the sensing system. The robot's actions were governed entirely by the state of the environment and its own state, the properties of its sensory system being essentially transparent. In contrast, conveying an accurate and timely measurement of the state of the spatial robot's juggling environment using a camera based system requires careful design of sensory strategies that we now review.

2.2.2 Camera Model: A Triangulation Function

We work with the simple projective stereo camera model,

$$\mathbf{p} : \mathbb{R}^3 \rightarrow \mathbb{R}^4$$

that maps positions in affine 3-space to a pair of image plane projections in the standard manner.

Let $\mathcal{F}_c$ be a frame of reference whose origin is at the focal point and whose z-axis is directed toward the image plane of this camera. Let $\mathcal{P} = [p_x, p_y, p_z, 1]^T$ denote the homogeneous representation with respect

to this frame of some spatial point. Then the camera, with focal length f, transforms this quantity as

$$ {}^{c}u = f \begin{bmatrix} p_x/p_z \\ p_y/p_z \\ 1 \\ 0 \end{bmatrix} \triangleq P_f({}^{c}p). \tag{9} $$

Here, ${}^{c}u \in \mathbb{R}^4$ is the homogeneous vector representing, with respect to $\mathcal{F}_c$, the line joining the origin of $\mathcal{F}_c$ to the image plane coordinates of ${}^{c}p$. Thus, for a camera whose position and orientation relative to the base frame, $\mathcal{F}_0$ are described by the homogeneous matrix ${}^{0}H_c$, the projection of a point, ${}^{0}p$ is

$$ u = P_f({}^{0}H_c{}^{-1}{}^{0}p). $$

Given two such cameras separated in space, whose frames of reference with respect to $\mathcal{F}_0$ are represented by ${}^{0}H_l$ and ${}^{0}H_r$, it is straightforward to derive a *triangulation function*, $\mathbf{p}^{\dagger}$, capable of reconstructing the spatial location of a point, given its projection in both images. In particular if projection onto the *right* and *left* image planes is given by

$$ \mathbf{p}({}^{0}p) \triangleq \begin{bmatrix} {}^{r}u_r \\ {}^{l}u_l \end{bmatrix} = \begin{bmatrix} P_{f_r}({}^{0}H_r{}^{-1}{}^{0}p) \\ P_{f_l}({}^{0}H_l{}^{-1}{}^{0}p) \end{bmatrix}, $$

then a (not unique) triangulation function is given by

$$ \mathbf{p}^{\dagger}({}^{r}u_r, {}^{l}u_l) \triangleq \frac{1}{2}\left({}^{0}H_r(o + t_r{}^{r}u_r) + {}^{0}H_l(o + t_l{}^{l}u_l)\right), \tag{10} $$

where

$$ \begin{bmatrix} t_r \\ t_l \end{bmatrix} \triangleq (R^T R)^{-1} R^T (o - {}^{0}H_r{}^{-1}{}^{0}H_l\, o) $$

and

$$ o \triangleq [0 \quad 0 \quad 0 \quad 1]^T;\ R \triangleq [-{}^{r}u_r \mid {}^{r}u_l];\ {}^{r}u_l = {}^{0}H_r{}^{-1}{}^{0}H_l\, {}^{l}u_l. $$

This amounts to finding the midpoint of the biperpendicular line segment joining the two lines defined by ${}^{r}u_r$ and ${}^{r}u_l$. Note that there is considerable freedom in the definition of $\mathbf{p}^{\dagger}$ since it maps a four dimensional space (the two image plane vectors) onto a space of dimension three (ball position).

Finally it is worth noting that although the implementation of a triangulation system of this type is simple, the measurement of the parameters required for its construction is quite difficult. This calibration problem is discussed in the [11].

2.2.3 Sensory Management: Real-Time Moments Generation

Following Andersson's experience in real-time visual servoing [1] we employ a first order moment computation applied to a small window of a threshold-sampled (thus, binary valued) image of each field. Thresholding, of course, necessitates a visually structured environment, and we presently illuminate white ping-pong balls with halogen lamps while putting black matte cloth cowling on the robot, floor, and curtaining off any background scene. Thus, the "world" as seen by the cameras contains only one or more white balls against a black background. In the case that only one white ball is presented, the result of this simple "early vision" strategy is a pair of pixel addresses, $c \in \mathbb{R}^4$, containing the centroid of the single illuminated region seen by each camera.

Figure 4 depicts the sensor management scheme we had employed to obtain c in support of the previously reported spatial one juggle [12]. Each camera is serviced by a pair of processors. A field from a camera is acquired in time τ_f by one of the pair while the other is busy computing its centroid. There is

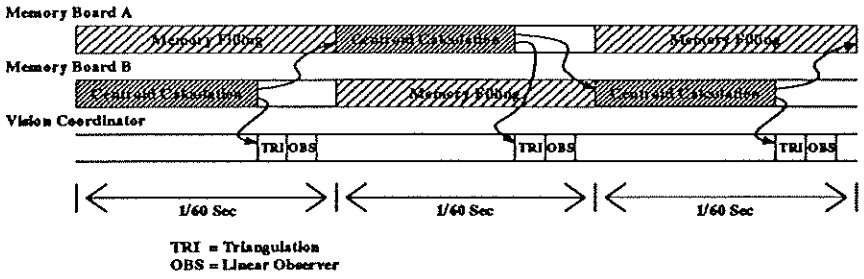

Figure 4: Timing Diagram for the Deployment of a Two Node Cyclops System in Support of Single Ball Sensing

now an obvious but still very interesting tradeoff to be made between real-time sampling considerations and accuracy of position estimates. Assume that the camera sampling interval — hence the maximal time allowed to process a particular sample — is set by the camera field period, τ_f, introduced above. [3] Denote by r the "processing rate" of the elementary image plane computation. For example on our XP/DCS node [6], we estimate that first order moments over reasonably complicated regions of the image plane may be computed at a rate of roughly $r = 200$ pixel/msec. It follows that no window may include more than $p = r \cdot \tau_f$ pixels, and, given l distinct windows, the sum of the areas allotted each one must not be greater than p. This constraint may be enforced by limiting the magnitude of a region at full resolution or limiting the resolution of a larger region through sub-sampling: the latter is the only feasible choice if recovery from unexpected perturbations in flight is desired. For our present hardware setup, $p \approx 3200$ pixels represents less than 2.5 % of the full 512×256 pixel image. Thus, unless the l windows have a combined area of less than 2.5 % of a field, a smaller window area within which to average implies a more accurate result.

In our successful one-juggle experiments, the moments were taken over a small subwindow of 30 by 40 pixels centered at the pixel location corresponding to the centroid address of the previously examined field. The pair of image plane centroids, $c \in \mathbb{R}^4$, was delivered to the vision coordinator at field rate, and was between one and two fields old, depending upon how much time it took to form the centroid. Centroid data form one processor was passed over to the second whose window coordinates were adjusted accordingly. The data was passed forward as well to the triangulation/observer processor. The two nodes then reversed roles, and the process repeated.

Our modifications to this procedure will be carefully discussed in the next section. Anticipating Section 4, note that such sensory management procedures represent the active component in the sensing control strategy.

2.2.4 Signal Processing: A Luenberger Observer

Given a report of the ball's position from the triangulator, we employ a linear observer to recover its full state — positions and velocities. Let w_k denote the position and velocity vectors of the ball at the time

[3] An imaging system is capable of acquiring a limited number of camera fields, n, before the available memory is filled, and further data must be discarded. For the Cyclops [5] system we have designed, this number depends upon the number of XP/DCS [6] nodes "ganged" together on the pixel bus of a single camera together with the size of the video RAM. In principle, this number might be significantly increased by allowing subsampling in hardware. In any event, assuming that it is *never* good to discard new data, this presents an upper bound on the the permissible sample period, the time spent computing with one field, $n \cdot s_f$. Longer sampling periods cannot be sustained without discarding data, so there is no point in allowing a lengthier computation unless uniform sampling is to be violated. Moreover, since the latency is strictly greater than the sample time, dwelling over an image will lead to stale information. In our present configuration $n = 1$. We will abide by the natural uniform sampling rate, s_f until experience suggest otherwise.

of the k^{th} field. As described above, the window operates on pixel data that is at least one field old,

$$p_k = F^{-\tau_f}(w_k),$$

to produce a centroid. Denote by W_k the function that takes a white ball against a black background into a pair of thresholded image plane regions and then into a pair of first order moments at the k^{th} field. The data from the triangulator may now be written as

$$\bar{b}_k = \mathrm{p}^\dagger \circ W_k \circ \mathrm{p}(Cp_k). \tag{11}$$

Thus, the observer operates on the delayed data,

$$\hat{p}_{n+1} = F^{\tau_f}(\hat{p_n}) - G(C\hat{p}_n - \bar{b}_n), \tag{12}$$

where the gain matrix, $G \in \mathbb{R}^{6\times 3}$, is chosen so that $A_{\tau_f} + GC$ is asymptotically stable — that is, if the true delayed data, Cp_n, were available then it would be guaranteed that $\hat{p}_k \rightarrow p_k$. [4]

We provide the mirror law an appropriately extrapolated and interpolated version of these estimates as follows. The known latency is corrected by the prediction,

$$\hat{w}_k = F^{\tau_f + \iota_k}(\hat{p}_k),$$

where ι_k denotes the time required by the centroid computation at the k^{th} field. Subsequently, the mirror law is passed the next entry in the sequence,

$$F^{i\tau_r}(\hat{w}_k), (i = 1, ..., \tau_f - \iota_{k+1})$$

until the next estimate, $\hat{p}_{k+1}$ is ready.

3 Sensing Issues Arising from Actuator Constraints

We now take up the central theme of the paper articulated in the introduction. We describe a set of enhancements to the sensory system software that enlarge the robot's behavioral repertory. Specifically, the robot's original task specification avoided most of the sensing difficulties associated with "multi-target tracking" by having two separate balls which did not cross in the images and remained continuously within the field of view. This section reviews the software changes that now permit juggling in the face of occlusions caused by balls going out of frame or crossing in the image plane.

We begin our discussion with a quick overview of the design constraints we are faced with in the robot mechanism itself. We then describe the software modifications. Finally, we present data that documents their effect on the machine's behavior.

3.1 The Constraints

The two-juggle task, as described in Section 2.1.2, was the original point of departure in designing our apparatus. Specifically, we anticipated a machine capable of juggling two balls separated by 90° to a height of approximately one meter. Realities of implementation, however, forced a number of modifications to

[4] In principle, one might choose an optimal set of gains, G^*, resulting from an infinite horizon quadratic cost functional, or an optimal sequence of gains, $\left\{G_i^*\right\}_{i=0}^{k}$, resulting from a k-stage horizon quadratic cost functional (probably a better choice in the present context), according to the standard Kalman filtering methodology. Of course, this presumes rather strong assumptions and a significant amount of à priori statistical information about the nature of disturbances in both the free flight model (3) as well as in the production of $\bar{b}$ from $\hat{d}$ via the moment generation process. To date we have obtained sufficiently good results with a common sense choice of gains G that recourse to optimal filtering seems more artificial than helpful.

this model task. To begin with, given the particular arrangement of our laboratory space, fixing our cameras to achieve a desirable spatial resolution of ball position resulted in a field of view smaller than a full meter above the robot's base. Moreover, having belatedly added mechanical brakes to our actuators (following a number of exciting mishaps with this large horsepower direct drive machine), we were left with much more massive links than originally anticipated, considerably reducing the accelerations our motors can deliver. Finally, we have found it difficult to raise the resonant frequency of the structural members supporting the robot's paddle much beyond 40 - 50 Hz without adding unreasonable mass (the present design is a formica-sheathed foam construction stiffened by piano wires).

Given such a machine, our initial strategy had been to adjust the task specification to fit its performance limitations. We lowered the height of the constituent one-juggles in order to stay within the camera's field of view. However, servicing two of these "faster-paced" juggles at a separation of 90° proved to require much higher torques than our base actuator could deliver. Bringing the two one-juggles closer together resulted in achievable base torques, but the still faster paced duty cycle required input torque profiles with sufficiently high-frequency energy content to seriously excite the resonance of the paddle structure. Thus, there being no obvious way of further adjusting the task to fit the machine, it became clear that we would have to re-fit the machine to the task.

We have recently successfully raised the component one-juggle tasks to the previously anticipated one meter height. This has required an ability to sense and recover from out of frame events (a ball passing out of the field of view due to the height of the juggle). Even so, servicing two of these slower one-juggles simultaneously cannot be accomplished at the intended 90° separation — we are presently experimenting with a two-juggle at 20° separation instead. But this mode of operation is not without difficulties and requires the sensing system to handle regularly occurring ball occlusions (two balls appearing at the arbitrarily close together in an image).

3.2 Modifications to the Sensing System

We now offer a brief sketch of the intuitively generated modifications we have made to the simple sensing system described above in Section 2.2.3. Briefly, these changes enable us to juggle balls that pass out of the cameras' field of view, and to continue juggling balls which may briefly occlude each other from the perspective of either camera. This has afforded us a successful implementation of a long-targeted milestone in our laboratory: the spatial two-juggle.

Individually, each of the "obvious hacks" that yield this new sensory capability represents a minor enhancement to the original system with little independent intellectual or engineering interest. However, getting them all to work in concert requires a greater amount of thought. Moreover, when combined, they significantly increase the capabilities of the robot. Finally, their addition to the original sensor management system introduces the first hint in our work that controlling the machine's "state of attention" may be an important and fundamental problem in robotics.

3.2.1 Occlusion Detection

Bringing the two one-juggle tasks closer together in space greatly increases the potential for the balls to pass arbitrarily close together in a particular image resulting in an *occlusion* event. Handling such situations requires either the ability to detect and reject images containing occlusions, or to locate the balls reliably in spite of the occlusion. Our disinclination to pursue the second option relates to our interest in exploring robust and extensible algorithms suited to our computational resources. While a two-ball occlusion can be relatively easily disambiguated, more balls or more complicated shapes give rise to increasingly difficult and computationally intensive geometric problems. Instead, we prefer to make a very coarse (and presumably, more robust) decision concerning when an occlusion has occurred, and entrust to a dynamical model (the observer of Section 2.2.4) the precise localization of where either ball

may be at any moment. As will be seen directly, this decision has consequences that set us out on the path of building a "dynamical sensor."

Since we are already committed to measuring the first order moments of a binary image as the primary method of localization, it is natural to extend this notion and use the second order moments as a simple and robust *occlusion detector.* Under well-structured lighting conditions, the "ballness" of an image is easily determined by putting thresholds around the the ratio of the eigenvalues of the matrix of the second order moments in conjunction with a test on the planar orientation its eigenvectors. When multiple balls appear in a single window — as determined by a data array that fails this second order moment test — the entire window of data is discarded and the observer simply integrates forward its present estimates. We presume that the results of such pure prediction will be more accurate than a computation based upon spurious centroid data.

An analogous line of reasoning supports our use of the zeroth order moment to characterize occlusions resulting from an out-of-frame or out-of-window event. A window of binary thresholded pixels with insufficient density is discarded as empty and the observer again updates its estimates on the basis of pure prediction. In the out-of-window event, the alternative strategy of re-examining the entire frame for the missing object is much too costly. In the out-of-frame event where a ball leaves the camera's field of view there is obviously no alternative to this strategy.

3.2.2 Observer Based Window Placement

In a situation where there are guaranteed to be regular occlusion events (because the balls are to be juggled high and close together), the policy outlined above of ignoring data from occluded windows severely compromises the effectiveness of the simple previously acceptable window placement manager. Recall from Section 2.2.3 that the original scheme simply used the centroid from the previous field as the window center in the next field. A spatial volume of roughly .1 meter diameter whose centroid is one field (.016 sec) old will not be likely to capture balls moving at speeds well in excess of 7 meters per second.

Instead, an obvious improvement results from using the estimates of the observer itself to place the windows. Namely, in the enhanced vision system, the windows in the next image to be processed are centered at a point formed by projecting the present state of the observer onto the camera image planes. Thus, the window locator has now become the output of a dynamical system internal to the robot whose inputs from the physical world we manage according to the decision process described above.

3.2.3 Impact Detection and Estimation

The two modifications described above have traded computational difficulty (simple geometric interpretation) for detailed dynamical knowledge (trusting the observer to correctly place the windows). However, the observer described in Section 2.2.4 is missing a model of a key dynamical feature in the life of the ball — the effect of the robot's impacts (u in (3)). If we drive the window manager with the output of the purely Newtonian observer then after the first impact the window center will continue to "fall" while the ball bounces up (with the relatively high velocity) and will almost certainly fail to lie within the next window — the ball is lost and the juggling stops.

In order to implement the observer with an enriched representation of the ball's dynamics we require both a model of impact and rather precise knowledge of the time the impact takes place. The former we have presented in [12]. The latter could be determined analytically in principle: starting with the assumption that the robot tracks its mirror law exactly (6); computing a position-velocity phase at contact; computing the induced effective impact. For reasons we have discussed at length in [2], our present mirror law constructions do not admit a closed form computation of the robot phase at contact. While numerical computation is a potentially feasible alternative, a predicted quantity will always be inferior to a sensed datum. Were the actual time of impact available, then a direct reading of the robot's

joint space measurements could provide the sensory alternative. Thus, we have chosen to augment the sensing system with a physical *impact detector.*

This device consists of a single microphone attached directly to the robot paddle whose output is passed through a narrow band filter tuned to the fundamental frequency produced by the impact, then rectified and threshold detected. The appropriate input, effectively a state change in the dynamical system (3) is calculated from the state of the ball and the robot at the time of the impact, and this is passed to the observer.

3.2.4 Window Size Adjustment

Although a central theme in this work concerns the advantages of trading a computationally intensive and brittle geometric model of the environment for a more robust dynamical model, there is no escaping the likelihood of error accumulation in either case. Our inability to compute with more than a small percentage of the available pixels during the 16 msec interval between successive camera fields forces a tradeoff between the accuracy of the centroid data input to the observer and the possibility of an unnecessary but unrecoverable out-of-window event. This tradeoff is governed by the choice of sampling resolution, or, equivalently, image plane window area. Intuitively, it seems clear that we ought to be able to develop some rational scheme for adjusting the sampling resolution in accord with an evolving set of error estimates. But what model of decision making offers an appropriate basis for such decisions, and where might one find a reasonable model by which to form the requisite estimates of error?

There are three principal sources of error in the sensor. First, noise inevitably corrupts the image frame processing (e.g., distortions introduced by thresholding an imperfectly illuminated scene, or by insufficient spatial resolution). Second, the observer is itself compromised by parametric errors (e.g., the gravitational force, $\tilde{a}$ in (3) is obtained through our calibration procedure) and omissions (e.g., there is no model of spin during flight). Finally, these are exacerbated by the intermittent loss of input data that attends occlusion events (e.g., out-of-frame events may easily last in excess of .25 seconds).

In the absence of a more principled approach to window area management, we have adopted the following strategy. Window area grows following any image plane measurement failure (i.e., an occlusion event). Window area shrinks following a valid measurement. The intuition is that we are capable of growing the windows large enough to compensate for the inevitable modeling error and reliably reacquire the ball either when it returns to the field of view or the occlusion ends. Conversely, after the observer has had a number of position inputs to process, we presume that the risk of losing the ball is outweighed by the potential advantage of gaining accuracy in the estimate from higher spatial resolution and minimizing the risk of further occlusions with the other ball/window.

3.2.5 Window Overlap/Prioritization

Of course, the larger the windows, the greater the likelihood of their overlapping and multiple balls being visible in individual windows. We have introduced an excision rule for removing intersecting regions from one window and assigning them exclusively to the other. Our rule weighs the cost of losing entirely a poorly tracked ball more heavily than that of corrupting the estimates of a relatively well tracked ball. This amounts to first looking for the things we know about in the image, blocking them out, and continuing to search for the remainder of the objects. Thus, we assign the windows a level of priority inversely corresponding to their size. The higher priority (smaller) window's pixels are excised from the moments computation of the lower priority (larger) window, but all of its pixels are used in the computation of its own moments.

In practice, this strategy seems to have the desired effect of not confusing a ball we are tracking well with one we have temporarily lost. That is, it avoids the spurious occlusion event caused by a well tracked ball (one we have seen in the recent past) entering a large window associated with a poorly tracked ball

(one whose observer error has not yet grown small). More significantly, we have not yet introduced a means of discriminating between occlusions generated by out-of-frame versus window overlap conditions. For example it is not uncommon that a window overlap near the edge of the field of view is followed followed by one of the balls moving out of the field of view. Suppose the out-of-frame ball is assigned a higher priority than the ball still in view while the window overlap persists (that is, the in-view ball remains within the now enlarged window owned by the out-of-frame ball). The excision rule gives the pixels generated by the in-view ball to the out-of-frame ball's window, the window manager now starts to track the in-view ball, and the out-of-frame ball is lost. This sort of failure happens frequently enough that still more sophisticated window excision and overlap handling strategies than presently in place seem to be desirable.

3.3 Effect of the Modifications

We have recently achieved a functional two-juggle but have not yet logged more than a few dozen hits of both balls [10]. We are convinced that the sensing enhancements discussed above have significantly contributed to our recent success, and that their refinement will afford two-juggle performance comparable to our current one-juggle performance. Some documentation of this recent progress now follows. It should be stressed that the figures in this section display real data taken from our runs with the physical apparatus.

3.3.1 Recovery from Out-of-Frame

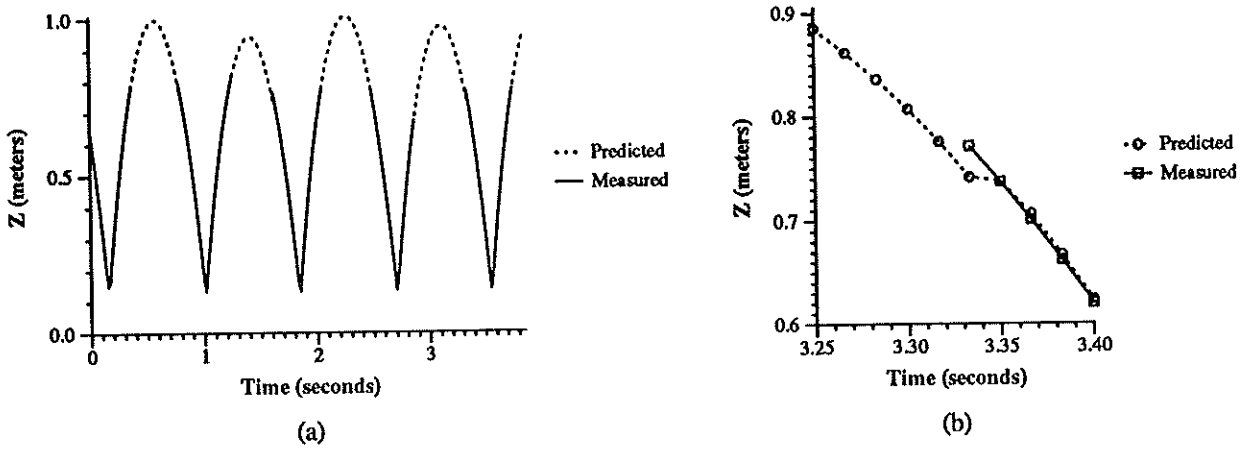

Figure 5: Measured (from physical data during a real juggling run) and predicted (by the observer operating on that physical data during this run) ball heights for an out of frame juggling sequence (a), and an expanded view of a single recovery event (b).

As mentioned above, this set of modifications has allowed the juggling height to be raised to the point that every juggle passes out of the field of view of our vision system. Figure 5 (a) and (b) depict exactly such a sequence. The top 0.25 to 0.4 seconds of each flight are outside the field of view, as is evident by the lack of position measurements during this period. Nevertheless the observer continues to predict the ball's location, and the ball is recovered as it passes back into the system's field of view. Figure 5(b) shows a detail of a single recovery. Evidently there is indeed a slight build up of prediction error (approximately 5 cm vertical error) over the near 0.5 second that the ball was out of view. However since the measurement window has grown, this magnitude of error is readily accommodated.

3.3.2 Recovery from Ball-Ball Occlussions

Having recently succeed in presenting the vision system with two objects for a prolonged period of time, we have been been able to observe the occlusion events discussed above. Figure 6 and 7 depict the image

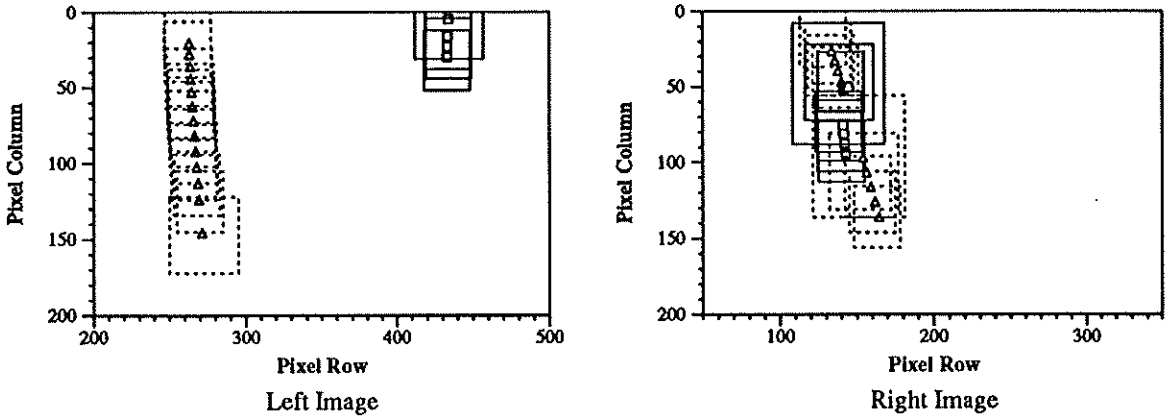

Figure 6: Left and Right image-plane tracks of a ball-ball occlusion event. Data are taken from an actual two-juggle run on the physical apparatus.

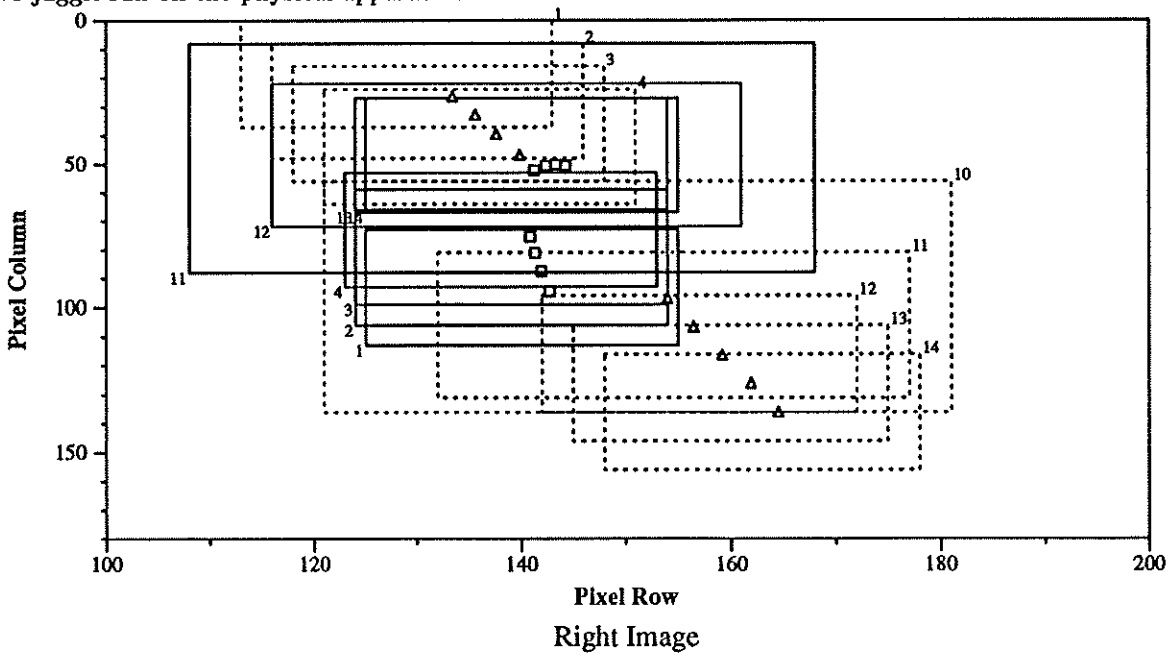

Figure 7: Expanded view of the left image-plane tracks showing the occlusion event.

plane tracks generated during an occlusion event. The small squares represent measurements assigned to ball 0, while the triangles are those associated with ball 1. The solid and dotted boxes are the windows used for moment calculations for ball 0 and 1 respectively. These are numbered corresponding to the temporal sequence of fields read. Figure 7 is a blow-up of a subregion of the right image plane shown in the previous figure, and is included so that the occlusion event (which occurs in the left camera) can be more clearly seen. In this particular sequence ball 0 (the squares) is rising towards its apex as ball 1 falls

"behind" it causing an occlusion in the 5^{th} frame. [5] The balls remain occluded (lying within the overlap region between the two large windows) until the 10^{th} frame at which point ball 1 reappears from behind the search window for ball 0, and frame 11 when ball 0 becomes visible due to the search window for ball 1 shrinking and exposing it.

Although we have just begun to analyze data of this sort from our a working two-juggle we feel that a careful analysis of these events will allow for improved tuning of the window sizes and their rates of growth and shrinkage. Currently reliable recovery from these occlusion events remains the major obstacle to achieving sustained two-juggle performance we would consider comparable to that which we have been able to achieve with the one-juggle task.

4 Toward the Control of Attention

We have no doubt that further intuitive tinkering with the enhanced sensor management scheme described in the previous section will result in satisfactory behavior. Yet, as more and more "enhancement modules" are added in the rather ad hoc fashion we have described, predicting and controlling their interactions becomes an increasingly difficult design problem. With the hope of developing a more principled approach to such design problems, we offer here a slightly more formal version of how to model and control the relevant sensor dyanamics. It should be stressed that this formalism neither incorporates all nor cleaves faithfully in detail to any of the of the "enhancements" we presently employ. In contrast to those purely pragmatic measures adopted to "get on with the work," this re-examination is heavily weighted by considerations of analytical tractability. We are convinced that this interactive process of pragmatic building followed by theoretical reflection leading to further refined building, and so on, is the best way to advance the infant field of robotics.

Image plane windows that are too large will introduce unecessary noise through subsampling and time taken to compute the centroid. Larger windows will also have a higher probability of occluding when there are multiple targets to track. On the other hand, windows that are too small will be likely to loose their target with potentially catastrophic results. In this preliminary exploration, we focus on the matter of how to place and size the windows in a rational manner.

4.1 The Window Management Variables as a "State of Attention"

The window manager controls the locus and extent of the image plane windows. Thus, we tentatively define a window's *state of attention* at some field interval, k, as the pair

$$a_k = \left(\hat{b}_k, \rho_k\right) \in \mathbb{R}^3 \times \mathbb{R}^+ \tag{13}$$

where $\hat{b}_k$ denotes an estimate of the spatial position of a falling ball, and where the positive scalar ρ_k is a measure of "certainty." With respect to a norm, $\|\cdot\|_M$, that will be defined below, a_k induces two windows on the two camera image planes including all stereo image pixel pairs, c having the property that

$$\left\{c \in \mathrm{p}(\mathbb{R}^3) : \|\hat{b}_k - \mathrm{p}^\dagger(c)\|_M \leq \rho_k\right\}.$$

If enough of the pixels corresponding to the image of the ball pass through the imaging threshold to produce a sufficiently large $zero^{th}$ order moment in the windows just defined, the first order moments will be passed to the triangulator to be interpreted as a spatial position. Otherwise, an "empty window" will

[5] To enhance visual clarity we have chosen to not show the windows that failed one of the "valid data" (i.e., zeroth or second order moment computation) tests and thus result in no input to the observer. Consequently, the windows "jump" from 4 to 11 and 4 to 10 for ball 0 and 1 respectively.

be logged. For the sake of notational simplicity, we will denote the situation that first order moments are successfully formed inside the windows of the k^{th} camera field as

$$b_k \in \mathcal{N}(a_{k-1}).$$

This notation immediately points up the *dynamics* intrinsic to the window management problem that appears at present as mere delay. Regardless of how it is computed, the state of attention, a_k must be assembled from information derived from existing sensory observations. Thus, the acquisition of new data is necessarily mediated by old knowledge.

For a suitable norm, we look back to the stabilized observer equations (12). Because the poles of the closed loop observer have been placed within the unit circle there exists a positive definite symmetric matrix, M, such that

$$[A_{\tau_f} + GC]^{\mathrm{T}} M [A_{\tau_f} + GC] < M,$$

and we will denote the Euclidean norms induced by this matrix as

$$\|x\|_{\mathrm{M}} \triangleq \left(x^{\mathrm{T}} M x\right)^{1/2}; \qquad \|A\|_{\mathrm{M}} \triangleq \sup_{\|x\|_{\mathrm{M}}=1} \|Ax\|_{\mathrm{M}}$$

For ease of exposition we introduce the notational conventions,

$$\alpha \triangleq \|A_{\tau_f}\|_{\mathrm{M}}; \qquad \bar{\alpha} \triangleq \|A_{\tau_f} + GC\|_{\mathrm{M}}$$

and assume, purely for further notational convenience, that the poles of the closed loop observer equation (12) have been placed on the real line with multiplicity two with the consequence that

$$\| [A_{\tau_f} + GC]^{-1} \|_{\mathrm{M}} = 1/\bar{\alpha}.$$

4.2 Observer Errors from a Noisy Model

Clearly, the task at hand is to develop a control scheme for updating the state of attention, a_k as a function of its previous value and presently available data. To do so we must append to our previous state estimation procedure some notion of its changing degree of certainty. Thus, reconsider the Newtonian flight model (3), with the addition of both a process and a sensor noise model. We wish to model the inaccuracies in the Newtonian flight law as well as the salient features of the inaccuracies in ball position measurement introduced through the use of the camera. The latter include two central phenomena: the absence of data when the ball lies outside of its assigned window; and the imprecision of spatial localization as the size of the window grows (and either delay grows or resolution shrinks correspondingly). For present exploratory purposes, we will be content with a crude deterministic representation of the imprecision inherent in these process and sensor models. What seems more critical to emphasize is an incorporation in the noisy model of the particular effect of image plane geometry. For it is exactly the window size and consequent spatial resolution that is under control.

We substitute for (3) and (11) the system

$$\begin{aligned} w[(j+1)\tau_r] &= F^{\tau_r}(w(j\tau_r)) + n_N(j\tau_r) \\ p_{k+1} &= w[k\tau_f] \\ \tilde{b}_k &= \hat{C}_k [p_k + n_S(\rho_{k-1})] \end{aligned} \tag{14}$$

It seems reasonable to take as a first crude model of the failings of the putative Newtonian free-flight model (3), n_N, a bounded deterministic sequence of uncontrolled inputs (perhaps generated via a map on the state space). The sensor noise introduced by thresholding a finite resolution image before computing moments is modeled by the function n_S. Because the resolution must decrease as the window magnitude

increases in consequence of subsampling, n_S is non-decreasing in its argument. Because no subsampling is required for sufficiently small windows, n_S is a positive constant for small values of its argument. For present purposes it seems adequate to take n_S to be affine in ρ,

$$\|n_S(\rho_k)\|_{\mathrm{M}} \leq \nu_0 + \nu_1 \rho_k \tag{15}$$

The deterministic output map, $\hat{C}_k$ returns the value $C = [I, 0]$ as in (3) when the body's image is in its assigned window, and vanishes otherwise:

$$\hat{C}_k \triangleq \begin{cases} C & : b_k \in \mathcal{N}(a_{k-1}) \\ 0 & : b_k \notin \mathcal{N}(a_{k-1}) \end{cases} . \tag{16}$$

We have determined in the face of an "empty window" to use simple extrapolation of the present estimate. Thus, the resulting observer takes the same form as (12) only with $\hat{C}_k$ (16) incorporated,

$$\begin{array}{rl} \hat{p}_{k+1} & = F^{\tau_f}(\hat{p}_k) + G(\bar{b}_k - \hat{C}_k \hat{p}_k) \\ \hat{w}(k\tau_f + j\tau_r) & = F^{\tau_f + \iota_k + j\tau_r}(\hat{p}_k); \qquad j = 0, 1, ..., \tau_f + \iota_{k+1} - \iota_k \\ \hat{b}_k & = CF^{\tau_f}(\hat{p}_k) . \end{array} \tag{17}$$

Here, we distinguish between the state estimate, $\hat{w}(\cdot)$, that is sent forward to the juggling algorithm, and the attention variable, $\hat{b}$, that will be sent back to the window manager. The robot gets $\hat{w}(k\tau_f)$ as soon as it is formed: future predictions are made at the faster physical rate, τ_r. The window manager will make use of $\hat{p}_k$ in the form of $\hat{b}_k$ to handle the $(k+1)^{st}$ image.

There are now three distinct kinds of error, each with its own causes and effects. The first is the standard error due to the observer,

$$\tilde{p}_k \triangleq p_k - \hat{p}_k,$$

and is governed by the dynamics

$$\begin{array}{rl} \tilde{p}_{k+1} & = \left(A_{\tau_r} + G\hat{C}_k\right)\tilde{p}_k + n_k \\ & \qquad n_k \triangleq Gn_S(\rho_{k-1}) + n_N[(k-1)\tau_f]. \end{array} \tag{18}$$

Denoting the present error magnitude by $\vartheta_k \triangleq \|\tilde{p}_k\|_{\mathrm{M}}$, we have

$$\begin{array}{rl} \vartheta_{k+1} & \leq \lambda_k \vartheta_k + \|n_k\|_{\mathrm{M}} \\ & \qquad \lambda_k \triangleq \begin{cases} \bar{\alpha} < 1 & : b_k \in \mathcal{N}(a_{k-1}) \\ \alpha > 1 & : b_k \notin \mathcal{N}(a_{k-1}) \end{cases} \end{array} \tag{19}$$

and the condition on $\hat{C}_k$ and λ_k may now be expressed explicitly as

$$b_k \in \mathcal{N}(a_{k-1}) \Longleftrightarrow \|C^{\mathrm{T}}(Cw[(k-1)\tau_f] - \hat{b}_{k-1})\|_{\mathrm{M}} < \rho_{k-1}. \tag{20}$$

Thus, there is a second sort of error associated with this event. It is due to the conjunction of process noise with time delay in the formation of the extrapolated state estimate. For, assuming $\|n_N\|_{\mathrm{M}}$ is bounded above by the scalar ν_N, we have

$$\begin{array}{rl} \|C^{\mathrm{T}}(Cw[(k-1)\tau_f] - \hat{b}_{k-1})\|_{\mathrm{M}} & \leq \|w[(k-1)\tau_f] - F^{\tau_f}(\hat{p}_{k-1})\|_{\mathrm{M}} \\ & \leq \alpha\|\tilde{p}_{k-1}\|_{\mathrm{M}} + \sum_{j=1}^{\tau_f}(\alpha^{k-j}\|n_N[(k-2)\tau_f + j\tau_r]\|_{\mathrm{M}} \\ & \leq \alpha(\vartheta_{k-1} + \tau_f \nu_N) . \end{array} \tag{21}$$

It follows that if ρ_{k-1} is at least as large as the last expression, we are guaranteed (within the limits of our noise model) that the k^{th} window will not be empty — that condition (20) will hold.

The third sort of error concerns the quality of the estimate passed forward to the robot. If $\tilde{w}_k \triangleq w(k\tau_f + \iota_k) - \hat{w}(k\tau_f)$ we have, using arguments similar to those above,

$$\|\tilde{w}_k\|_M \leq \alpha^{\iota_k} (\vartheta_k + (\tau_f + \iota_k)\nu_N) \tag{22}$$

Since we prohibit the window manager from addressing more pixels than can be processed within the $16msec$ frame period, ι_k, the centroid computation time, increases by units of τ_r and saturates at the value τ_f:

$$\tau_r \leq \iota_k \leq \tau_f.$$

Thus, $\|\tilde{w}_k\|_M$ is a non-decreasing function of both ϑ and ρ.

4.3 Certainty Estimates from a Parallel Observer

Computations (21) imply that ρ_k should be set in relation to ϑ_k in order to insure data to the observer. But, unfortunately, we are not in possession of the error magnitude, ϑ, for the very reason that we were led to build an observer in the first place. Since $\hat{p}$ represents our only knowledge of p, the best estimate of ϑ is 0 as matters stand presently. To address this deficit, we will build a second state estimator and attempt to get additional information concerning ϑ by comparing the two.

Using the invertibility of the observability matrix,

$$\Theta \triangleq \begin{bmatrix} C \\ CA_{\tau_f} \end{bmatrix},$$

we may define a very different estimate of p of the form

$$d_k = F^{\tau_f}\left(\Theta^{-1}\left(\begin{bmatrix} \bar{b}_{k-1} \\ \bar{b}_k \end{bmatrix} - \begin{bmatrix} 0 \\ Ca_{\tau_f} \end{bmatrix}\right)\right).$$

This is a dead-beat observer for p, in the sense that if $\tilde{d}_k \triangleq p_k - d_k$, then converges to zero in two steps from all initial estimates, d_0 in the absence of noise, $n_S = n_N = 0$. In the present setting we have

$$\tilde{d}_k = \sum_{j=1}^{\tau_f} A_{\tau_r}^{k-j} n_N(\tau_r j) - A_{\tau_f}\Theta^{-1}\begin{bmatrix} \hat{C}_{k-1} n_S(k-1) \\ \hat{C}_k\left(n_S(k) + n_N(k-1)\right) \end{bmatrix}$$

and, noticing that

$$\begin{aligned} \|\hat{p}_k - d_k\|_M &= \|\tilde{d}_k - \tilde{p}_k\|_M \\ &= \| \left(A_{\tau_r} + G\hat{C}_k\right)\tilde{p}_{k-1} + n_{k-1} + \textstyle\sum_{j=1}^{\tau_f} A_{\tau_r}^{k-j} n_N(\tau_r j) - A_{\tau_f}\Theta^{-1}\begin{bmatrix} \hat{C}_{k-1} n_S(k-1) \\ \hat{C}_k\left(n_S(k) + n_N(k-1)\right) \end{bmatrix} \|_M \\ &\geq \tfrac{1}{\|(A_{\tau_r}+GC)^{-1}\|_M}\vartheta_{k-1} - \nu_\Delta(\rho_{k-1}, \rho_{k-2}), \end{aligned}$$

where

$$\nu_\Delta(\rho_{k-1}, \rho_{k-2}) \triangleq \|n_{k-1}\|_M + \alpha\tau_f\nu_N + \frac{\alpha}{\|\Theta\|_M}\left(\nu_N + \|n_S(\rho_{k-1})\|_M + \|n_S(\rho_{k-2})\|_M\right),$$

we are led to define a worst case estimate for ϑ as

$$\hat{\vartheta}_{k-1} \triangleq \left[\|\hat{p}_k - d_k\|_M + \nu_\Delta(\rho_{k-1}, \rho_{k-2})\right]/\bar{\alpha}, \tag{23}$$

guaranteeing that $\hat{\vartheta}_{k-1} \geq \vartheta_{k-1}$. For purposes of later analysis, note that

$$\hat{\vartheta}_{k-1} \leq \frac{\lambda_{k-1}}{\bar{\alpha}}\left[\vartheta_{k-1} + 2\nu_\Delta(\rho_{k-1}, \rho_k)\right]. \tag{24}$$

4.4 Window Radius Dynamics for Bounded Estimator Errors

Equipped with a worst case estimate for ϑ, we are now in a position to adjust ρ. According to the previous calculations (21), a window radius management strategy that achieves the relation

$$\rho_k \geq \alpha\left(\vartheta_k + \tau_f \nu_N\right)$$

guarantees data to the observer at step $k+1$. Noting that ϑ_k is causally determined by ρ_k, and thus cannot be estimated directly by the procedure (23) at stage k, we appeal to (19) and note that the desired relation is implied by

$$\rho_k \geq \alpha\left(\lambda_{k-1}\vartheta_{k-1} + \|n_{k-1}\|_{\mathrm{M}} + \tau_f \nu_N\right).$$

This demonstrates that the radius adjustment procedure

$$\rho_k = \alpha\left(\lambda_{k-1}\hat{\vartheta}_{k-1} + \|n_{k-1}\|_{\mathrm{M}} + \tau_f \nu_N\right) \tag{25}$$

will always yield a window large enough to capture the next centroid, up to the limits of the error models employed.

But there is now a question of observer convergence. For, recall that as ρ increases, the quality of the robot estimates deteriorates. Eventually, the recourse to subsampling might begin to have a net destabilizing effect through the injection of noise represented by n_k in (19)). We must show that the coupled dynamical system (19), (25) remains bounded.

Approximating the appearance of ρ in n_k, ν_Δ to first order (15), we have

$$\begin{aligned}
\|n_k\|_{\mathrm{M}} &\leq \gamma(\nu_0 + \nu_1\rho_{k-1}) + \nu_N \\
\nu_\Delta(\rho_k, \rho_{k-1}) &\leq (1+\alpha\tau_f)\nu_N + \gamma(\nu_0 + \nu_1\rho_k) + \tfrac{\alpha}{\|\Theta\|_{\mathrm{M}}}(\nu_N + 2\nu_0 + \nu_1\rho_k + \nu_1\rho_{k-1}) \\
&= (1+\alpha(\tau_f + 1/\|\Theta\|_{\mathrm{M}})\nu_N + (\gamma + 2\tfrac{\alpha}{\|\Theta\|_{\mathrm{M}}})\nu_0 + \nu_1\left(\gamma + \tfrac{\alpha}{\|\Theta\|_{\mathrm{M}}}\right)\rho_k + \nu_1\tfrac{\alpha}{\|\Theta\|_{\mathrm{M}}}\rho_{k-1}.
\end{aligned}$$

The coupled dynamical inequalities in question now may be written

$$\begin{aligned}
\vartheta_{k+1} &\leq \lambda_k\vartheta_k + \nu_1\rho_{k-1} + \gamma\nu_0 + \nu_N \\
\rho_{k+1} &\leq \alpha\left(\tau_f\nu_N + \gamma(\nu_0 + \nu_1\rho_{k-1}) + \nu_N + \tfrac{\lambda_k^2}{\alpha}\left[\vartheta_k + 2\nu_\Delta(\rho_k, \rho_{k-1})\right].\right)
\end{aligned}$$

4.5 Boundness of the State of Attention

Now in the coordinate system, $x \triangleq [\chi_1, \chi_2, \chi_3]^{\mathrm{T}}$, where $\chi_1(k) \geq \vartheta_k$ bounds the actual Lyapunov magnitude of (17) and $\chi_2(k) \geq \rho_k$, $\chi_3(k) \geq \rho_{k-1}$ represent bounds on the most recent window radius values, we obtain the dynamics

$$\begin{aligned}
x(k+1) &= Q_k x(k) + r \\
Q_k &\triangleq \begin{bmatrix} \lambda_k & 0 & \nu_1 \\ \frac{\alpha\lambda_k^2}{\bar{\alpha}} & \alpha\nu_1 g_1 & \alpha\nu_1 g_2 \\ 0 & 1 & 0 \end{bmatrix} \\
r &\triangleq \begin{bmatrix} r_1 \\ r_2 \\ 0 \end{bmatrix},
\end{aligned} \tag{26}$$

where the symbols $g_i, r_i, i = 1, 2$ denote constants derived from the computations developed above.

By construction of the radius adjustment procedure (25), the state of this system enters a region where $\lambda_k = \bar{\alpha} < 1$ after an initial transient. Now, elementary root locus analysis of the characteristic polynomial of this system,

$$s^2(-\bar{\alpha} + s) + \alpha\nu_1\left[(g_2 - 1)\bar{\alpha} + (\bar{\alpha}g_1 - g_2)s + g_1 s^2\right]$$

shows that the matrix Q has roots in the unit circle of the complex plane for small enough values of ν_1: they originate at $\{\bar{\alpha}, 0, 0\}$. This implies that if the noise coefficient, ν_1 is sufficiently small relative to the other parameters then the window management system succeeds in keeping the windows large enough to retain the required image, but not so large as to destabilize the estimation procedure.

5 Conclusion

The foregoing scheme is a reasonably faithful formalization of how the windows are placed in our present juggling system — the assignment of $\hat{b}_k$ (17). It is considerably less faithful to the way in which window radii, ρ_k, are determined. It is entirely silent concerning the manner in which ball-ball occlusions on one or the other image plane should be handled and their effects analyzed. Nevertheless, this initial attempt at formalization has clarified a number heretofore fuzzy issues.

The chief advance in our thinking is represented by the proposed formal radius management scheme (25) — the manner in which measurement uncertainty might be incorporated into our sensory manager. Specifically, the idea of running a second state estimator in parallel with the traditional Luenberger observer in order to compute a plausible bound on the estimate's error magnitude appears to be new. Whether it is also effective will depend upon the relative magnitude of sensor noise as compared to that of the drift term in the unstabilized Newtonian flight model (3). Further experimentation will be needed to ascertain whether or not this is so.

Acknowledgement

We would like to thank Prof. Koichi Hashimoto for his patient encouragement and helpful comments during our work on this paper.

References

[1] R. L. Andersson. *A Robot Ping-Pong Player: Experiment in Real-Time Intelligent Control.* MIT, Cambridge,MA, 1988.

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

[3] M. Bühler, D. E. Koditschek, and P. J. Kindlmann. Planning and control of a juggling robot. *International Journal of Robotics Research*, (to appear), 1992 .

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

[5] M. Bühler, N. Vlamis, C. J. Taylor, and A. Ganz. The cyclops vision system. In *Proc. North American Transputer Users Group Meeting*, Salt Lake City, UT, APR 1989.

[6] M. Bühler, L. Whitcomb, F. Levin, and D. E. Koditschek. A new distributed real-time controller for robotics applications. In *Proc. 34th IEEE Computer Society International Conference — COMPCON*, pages 63–68, San Francisco, CA, Feb 1989. IEEE Computer Society Press.

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

[8] Michael Erdmann and Matthew T. Mason. An exploration of sensorless manipulation. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1569–1574, San Francisco, CA, Apr 1986. IEEE.

[9] T. McGeer. Passive dynamic walking. In *The International Journal of Robotics Research*, page (to appear), Jun 1989.

[10] A. A. Rizzi and D. E. Koditschek. Further progress in robot juggling: The spatial two-juggle. submitted for presentation at the 1993 IEEE Conf. on Rob. and Aut., May 1993.

[11] Alfred Rizzi and Daniel E. Koditschek. Preliminary experiments in robot juggling. In *Proc. Int. Symp. on Experimental Robotics*, Toulouse, France, June 1991. MIT Press.

[12] Alfred A. Rizzi and D. E. Koditschek. Progress in spatial robot juggling. In *IEEE Int. Conf. Robt. Aut.*, pages 775–780, Nice, France, May 1992.

[13] 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.

[14] Stefan Schaal, Christopher G. Atkeson, and Sherif Botros. What should be learned? In *Proceedings of the 7'th Yale Workshop on Adaptive and Learning Systems*, pages 199–204, New Haven, Connecticut, USA, 1992. Yale University.

[15] 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*, (to appear), 1992.

Kollab