Real-Time Vision for Robot Swat Juggling

Sanjiv Singh

CMU-RI-TR-90-27

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

December 10, 1990
(revised August 15, 1991)

Table of Contents

List of Figures

Abstract

This paper describes the implementation of a real-time vision system developed to track moving objects in real-time. In the implementation described, a "puck" sliding on an inclined plane is tracked so that its position and velocity may be used to "swat-juggle" it to a constant height at a specified set point. Raw centroid calculations of the puck based on images from a CCD camera are filtered using an "augmented" linear observer and an interpolator to produce puck state estimates. Experimental data and results are presented and compared to the previous method of puck state sensing.

1. Introduction

Computer vision is a natural means of measuring positions and velocity of moving objects as in the case of a planar juggler developed at Center for System Science, Yale University that "swat-juggles"[1] two pucks falling freely on a frictionless plane inclined into the earth's gravitational field.

To be able to "swat-juggle," the robot must have access to the state (position and velocity) of the pucks. While puck position, can be measured directly from analysis of a video image, the velocity must be estimated. Generally, it is not a good idea to simply differentiate position estimates to obtain velocity since the position estimates can be noisy, and the resulting velocity estimates are noisier still. Hence, we would like to estimate the unmeasured states in a principled way as well as filter the measured states. Another reason to keep obtain state estimates at a high rate is that it helps in the image processing- if accurate estimates of the puck can be obtained, it is possible to process only part of the image around the expected image of the puck. This report will describe the methodology used to obtain position and velocity updates at a high rate using computer vision.

Other researchers have reported similar work- in two cases a ball was tracked in real-time using special purpose hardware [Andersson, Atkeson]. Atkeson's swat juggling robot used a pair of cameras positioned orthogonally, to measure the position of a ball by calculating the centroids of the image of the ball in a scene. There are two major differences between his work and that described in this report. Atkeson assumed an orthographic (as opposed to perspective) projection for each camera. This required that both cameras be positioned exactly orthogonal to the coordinate frame of the robot. Under this assumption, camera calibration is a relatively simple process, but suffers in accuracy and the inability to deal with arbitrarily positioned cameras. We have implemented a method that does not stipulate position of the camera(s). Secondly, Atkeson's juggler estimated position and velocity at frame rate. More precisely, every time a video image was obtained, a linear least squares fit was performed to estimate position of the ball at impact. Such an approach is purely geometric there is no consideration of dynamics like gravity and friction. In contrast, the work reported is motivated by the need to accurately determine the position of the puck at a high rate (1 Khz). This is done by filtering position measurements obtained at 60 hz from the camera, using an augmented linear observer that explicitly encodes gravitational and frictional forces. An interpolator that also encodes puck dynamics, is used to obtain position and velocity estimates between position measurements.

Andersson's ping-pong playing robot also tracked a moving ball in real time. His approach is similar to ours in that dynamics were used to correct position measurements. Andersson found it necessary to account for gravity and air-drag. There were compensated for with a local quadratic fit to position data at every position measurement. Velocity and acceleration estimated from this fit were then used to compute higher order

[1] Swat-Juggling refers to the action of "swatting" an object (usually a puck or a ball) to a specified height without grasping the object.

terms using a dynamic model which in turn were used to "pre-correct" the sums for the quadratic fit of the following position measurements.

Buehler demonstrated a variety of juggles on the Yale Juggle as examples of a class of tasks which involve repeated robot-environment interactions [Buehler89, Buehler90a, Buehler90c]. Position and velocity estimation was accomplished by the use of an oscillator inside each puck in conjunction with an inductive grid that is buried in the inclined plane over which the puck is moving. A single measurement of a puck using this scheme was not very accurate (+/- 1 cm) but was significantly improved through a high sampling rate and a linear observer that filtered the measurements. A problem with this scheme is that it is difficult to scale to the case where more than two objects must be tracked. Ideally, we would like a sensing mode that doesn't limit the number of objects being tracked. Further, for a system that tracks objects moving in three space, it will not be possible to use an inductive grid.

We were motivated by the need for a more general and extensible means of estimating puck states. The system implemented can be extended to track multiple objects as well as objects moving in three space, in a straightforward manner. This report describes the integration of a real-time vision system with the planar juggler using off the shelf CCD cameras. Once a snap shot of a puck is obtained, an observer and interpolator are used to filter position measurements, to estimate velocities and to predict the motion between measurements. Our work has essentially followed the same methodology used by Buehler except that we have replaced the inductive sensing scheme with passive vision.

In the following sections, we first discuss the apparatus used. Next, camera calibration scheme implemented is briefly described in Section 3, and the computing architecture for the entire juggler is detailed in Section 4. Section 5 discusses the design of the tracking system, and, Section 6 discusses experimental results.

2. The Planar Juggler

An existing apparatus was used in the experimentation[Buhler89]. The apparatus consists of one or two pucks sliding on an inclined plane that are batted successively by a bar with a billiard cushion rotating in the juggling plane. There are three parts to this system:

1. Puck State Sensing: to close a feedback loop at a high rate, it is necessary to be able to access positions and velocities of the pucks at a high rate (approximately 1Khz). Previously, puck sensing was accomplished by placing an oscillator inside each puck and burying a grid inside the juggling plane, thus imitating a digitizing tablet. State estimation was accomplished by measuring grid voltages induced in the grid by the pucks and filtering the raw data using puck dynamics. A sensing module (processor and multiplexing hardware) was dedicated to this task.

   A new sensing module has been designed that uses vision as the mode of measuring puck positions and velocities.

2. Juggling Algorithm Computation: a separate module is designated to compute the reference trajectory (angle and angular velocity) of the juggling bar given the robot state (from a shaft encoder on the juggling arm motor) and puck states.

3. Motor Servo Control: this module is dedicated to commanding a high torque DC servo actuator at a rate of approximately 1.5 Khz using a PD algorithm.

Fig. 1 shows the configuration of the juggler with the added vision system. Both puck state sensing modalities were retained so as to compare data from the grid sensor and the vision sensor, though juggling was accomplished only through the use of vision data.

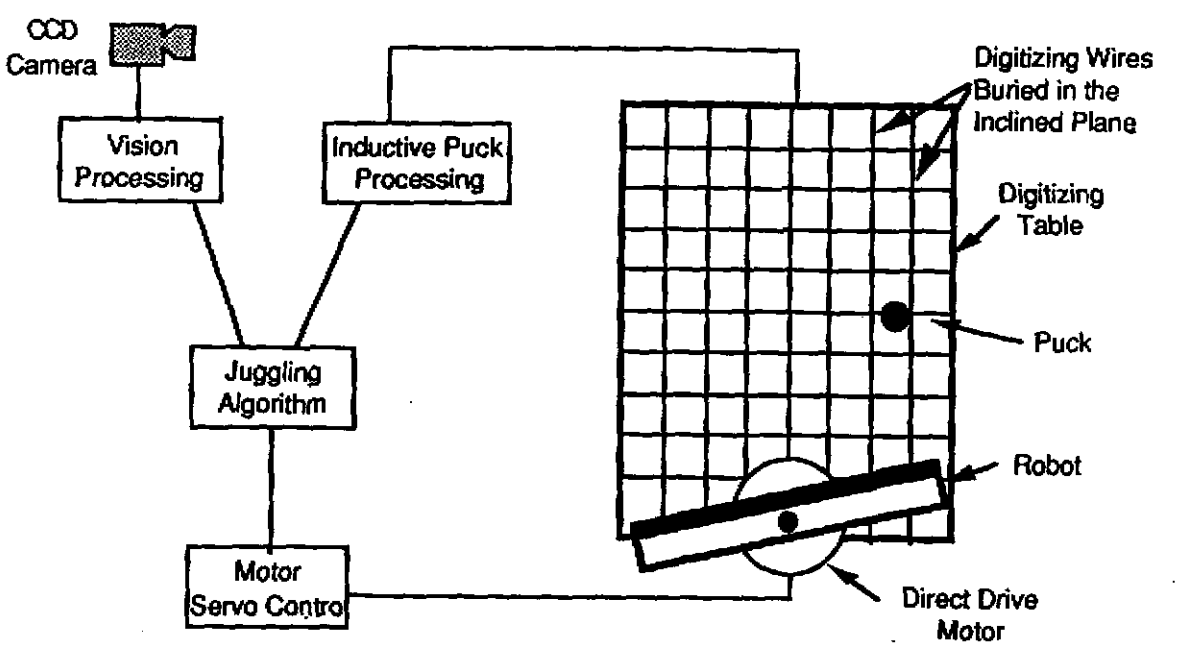

Figure 1: Configuration of the Planar Juggler

3. Camera Calibration

Figure 2 shows the physical setup of a CCD camera mounted on the ceiling looking at the juggler. Note that the camera has been rotated about the optical axis by 90 degrees so as to provide greater sensing resolution in the vertical direction (the video image has more columns than rows).

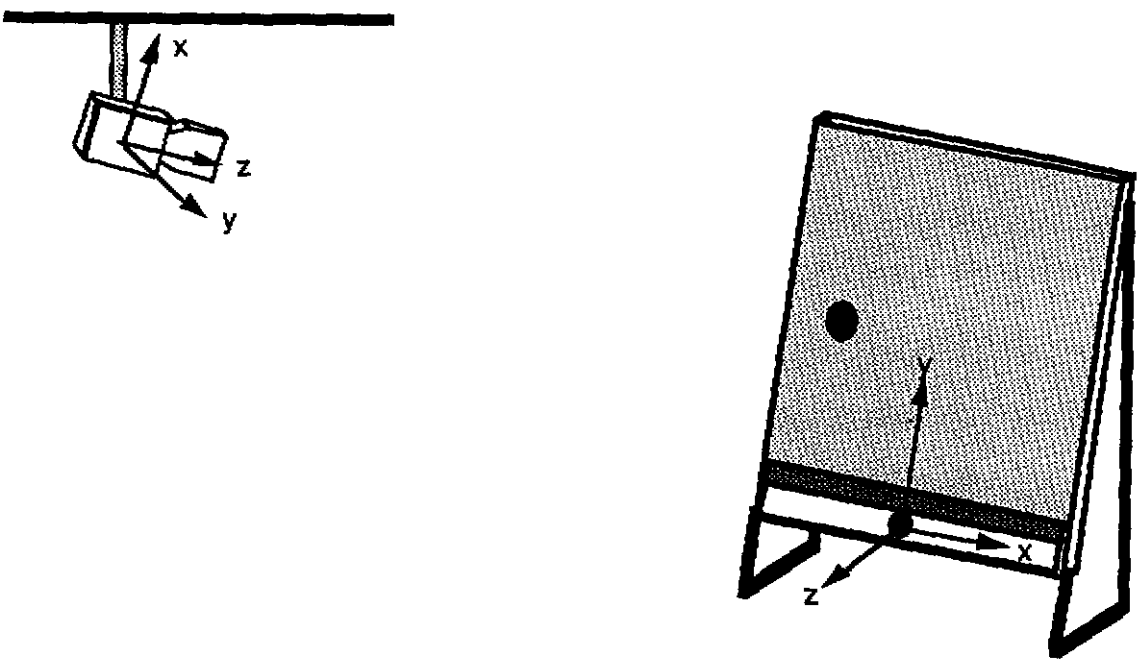

Figure 2: Camera setup for position sensing

To be able to sense puck position, it is necessary to go through a calibration phase which relates pixel locations in the image to physical locations on the juggling plane. The calibration scheme [Tsai], computes the following parameters:

f: focal length of the camera
s: a fudge factor to compensate for disparate digitization timings between the camera and the frame grabber.
k1,k2: lens distortion parameters
R: A 3X3 Rotation matrix that describes the transformation from the camera frame to the world frame
T: A 1X3 translation matrix that denotes the translation vector between the above two matrices.

It is necessary to provide input to this algorithm in the form of a set of training points that lie in several planes, for which both spatial coordinates in world frame and image coordinates in the image space are known. For objects that lie in a plane, it is only necessary to provide training points that all lie in a plane. However, for the general case, in which objects may lie in three space, it is necessary to provide training points that lie in several planes. This is accomplished by using an image of a calibration grid located on the juggler

as shown in figure 3.

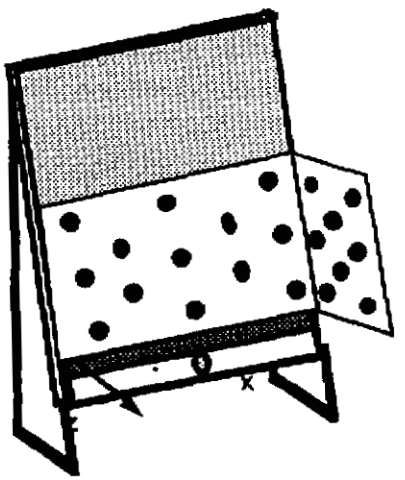

Figure 3: Calibration grid located on the inclined plane

Figure 4 shows an actual image of the calibration grid obtained from the CCD camera.

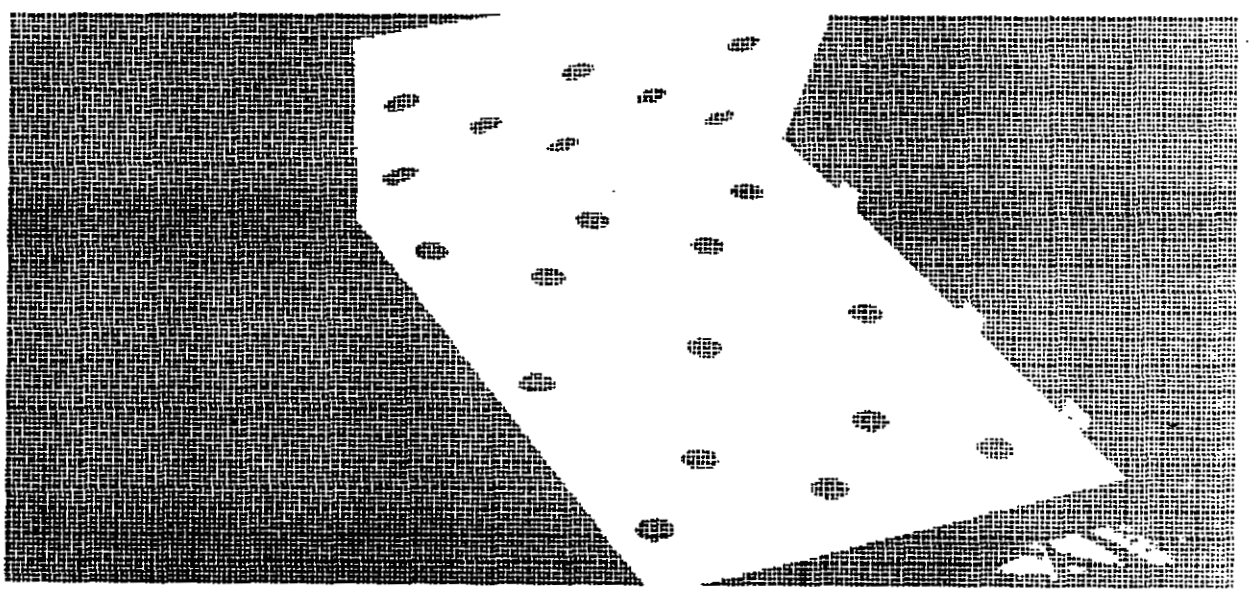

Figure 4: A video image of the calibration grid

The location of the centroid of each of the circles on the grid is known a priori through careful measurement, while the corresponding centroids of the circles in image space must be computed. At present, the correspondence problem of deciding which circle in three space corresponds to which circle in the image space is solved by hand. The result of this process is a set of image coordinates for every circle on the calibration grid:

row, col: the centroid in image space
x, y, z: the centroid in world space

The calibration scheme uses these training points as inputs to a non-linear minimization scheme to estimate the parameters (f, s, $k1$, $k2$, R, T). Once the camera parameters are known, given an (i, j) pair in image space, the corresponding point in world space can be solved by equations 8(a) and 8(b) in Tsai's report. Essentially, two relationships-$f(x, z)$, $f(y, z)$ are obtained.

If the object(s) being tracked move in a plane then the z parameter is known and it suffices to simultaneously solve for x and y in the two equations above. However, if the object moves in 3-space, two cameras are needed. For each camera there are two such functions, giving rise to an over constrained set of equations: $f_1(x, z)$, $f_1(y, z)$, $f_2(x, z)$, $f_2(y, z)$. These four equations can be solved using a variety of minimization methods. In our case the pucks move in a plane so a single camera suffices.

4. System Architecture

The Cyclops vision system [Cyclops] was used to track the pucks. The Cyclops system consists of three components:

- digitizer: the digitizer digitizes the RS-170 signal from the CCD camera and outputs the image on a video bus. The RS-170 signal is interlaced, so each of the half-frames is broadcast on the video bus, alternately.
- memory modules: An arbitrary number of memory modules can listen to the video bus. These are configured to listen load one of the two half frames broadcast by the digitizer.
- frame-processors: Each memory module, has attached to it, a frame-processor that operates on the data loaded into the memory module. Each frame processor is able to communicate with other processors in the system via messages.

The configuration is shown in figure 5.

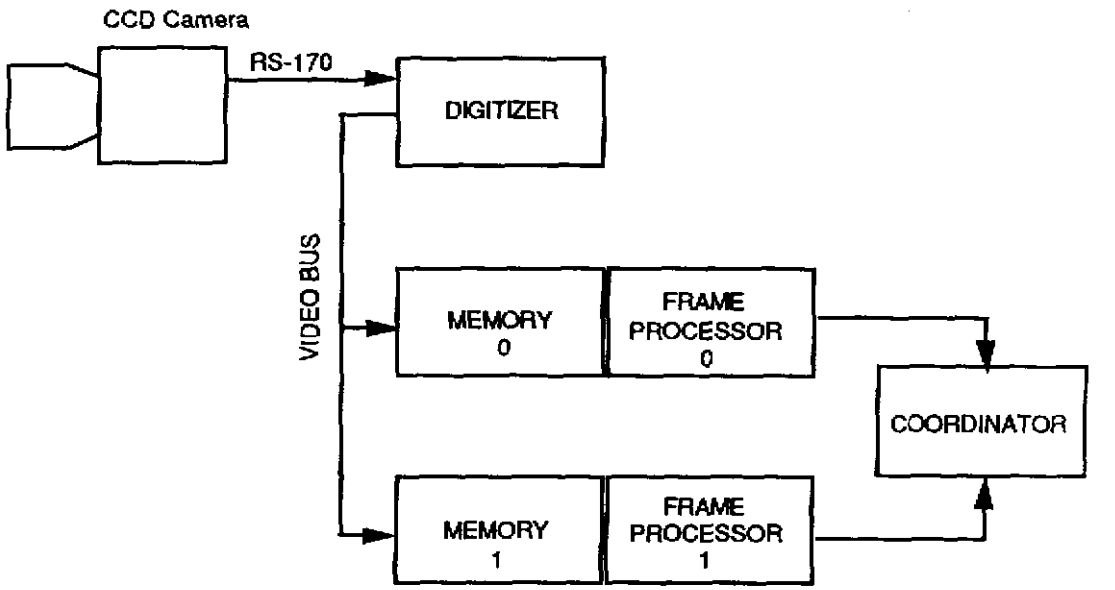

Figure 5: The Cyclops vision system

In this case, each frame-processor is dedicated to a window on each half frame for the sake of efficiency. This window is moved as the object being tracked moves. We used a window of 30X 60 pixels and were able to do all the processing within 7ms. The window size was large enough that once tracking was initiated, the puck never escaped the window under normal experimentation.

Within this window, a binary thresholding operation is done and the centroid (first order moment) of the bright pixels is found. Since the puck is a bright object against a dark background, the operations of thresholding and finding centroids are straight forward- all pixels within the window that have pixel values above the threshold are averaged in their

x and y values to obtain the centroid. Each frame-processor then converts the centroid information to world coordinates and these are sent to a "Coordinator" module.

Since each frame-processor gets a frame at 30Hz, it is possible to combine the data from both processors to obtain state estimates at 60 Hz. The Coordinator does exactly this. Its function is two fold:

- Updating the window: As each frame-processor sends centroid information to the coordinator, the new position of the puck in the image is sent to the other frame-processor so that it can use an updated estimate of the window in which to find the puck in the next sensing cycle.

- Filtering raw position data: Data from vision system is noisy on a per sample basis but it can be filtered using a linear observer that uses puck dynamics to provide to smooth raw position data obtained from the vision system. Figure 6 shows (x) raw position data obtained from the vision sensor. The observer is also used to estimate puck velocities.

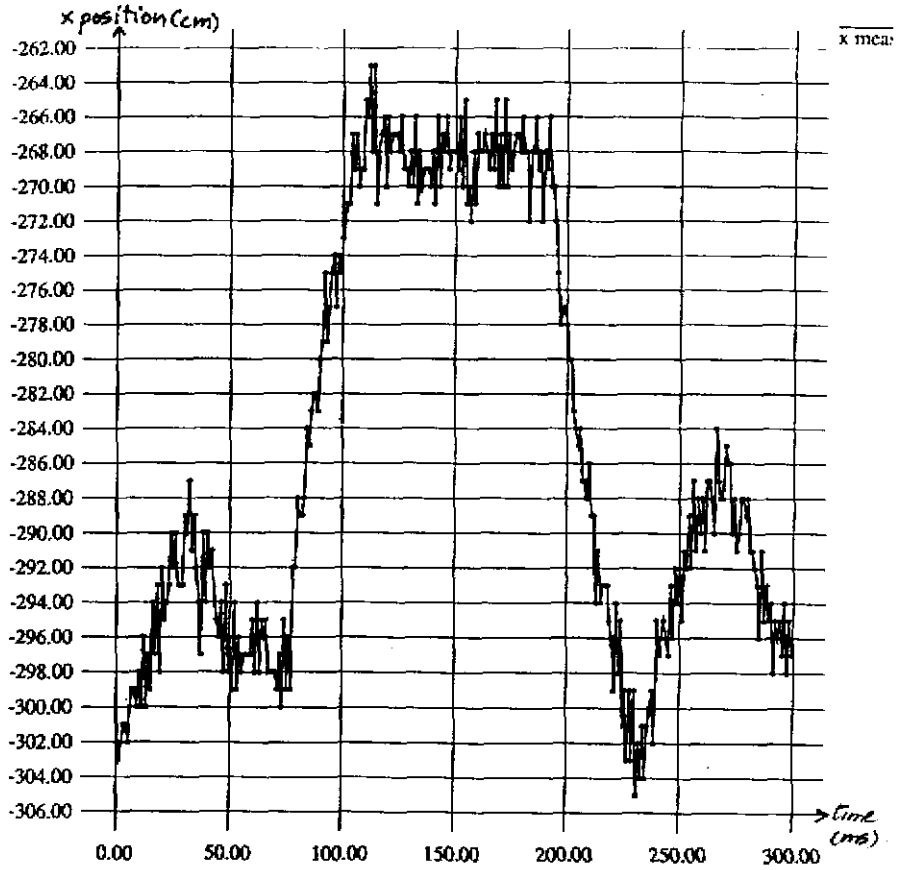

Figure 6:Raw (x) position data from the vision sensor

The previous incarnation of the juggler used 3 T-800 transputers to accomplish juggling. One transputer was dedicated to each of juggling algorithm computation, puck sensing and motor servo control. Additionally a T-400 transputer was used as a host interface to a PC AT, primarily for the purposes of compilation and data logging. The current work

has introduced 4 additional processors. Two T-800 transputers are used as frame-processors for low level image calculations. Their output is sent to another T-800 transputer that runs the "coordinator" tasks. A fourth transputer (B00-7) is used as a graphics processor to display the located centroid of the puck in the image. The whole system is configured as in figure 7.

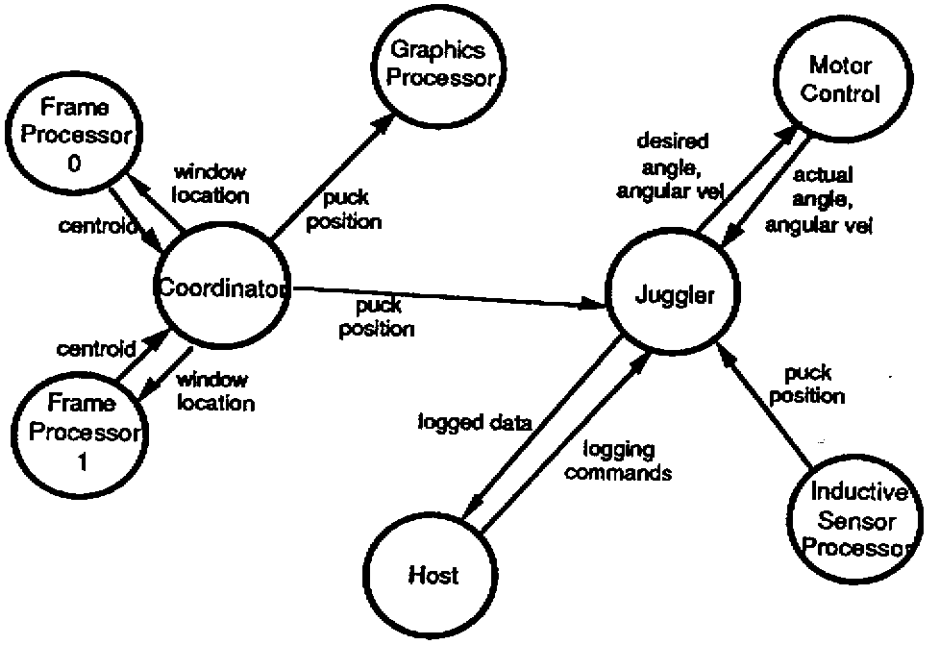

Figure 7: The MIMD architecture for the juggler

5. Tracking the Puck

5.1. Puck Dynamics

A complete model of puck dynamics incorporates gravitational and friction forces experienced by the puck. Friction forces can be characterized by two different effects- dry friction and viscous friction. The former is the force experienced during constant sliding, whereas viscous friction is the force required to move a stopped object. since the puck is in motion constantly, it is sufficient to only model dry friction. A simplified model of the puck motion (apart from the impact) can be described by:

$$\ddot{y} = -g - (\dot{y} \cdot f_y) \tag{1}$$

$$\ddot{x} = -(\dot{x} \cdot f_x) \tag{2}$$

Here x, y denote the puck position, g is the acceleration due to gravity, f_x and f_y are the dry friction terms (resulting from contact between the puck and the juggling plane) in the x and y directions, respectively. Dry friction has been experimentally determined to be 0.16 [Buhler90b]. However, this estimate was found to be good only at high velocities, which in our task are in the y direction. Velocities in the x direction are almost negligible and hence we have ignored the friction in this direction.

Puck dynamics are much harder to encode at the impact. A restitution model is used in the control of the juggling arm [Buhler90b]. For the purposes of tracking the puck, we have used an ad hoc scheme that resets some of the observer states when impact is detected.

5.2. System Timing

Figure 8 shows the time separation of events. The CCD camera shutter opens at S_1, S_2,.., S_n (two shutter events are separated by 1/60th sec). At time S_{n+1}, the first half-frame is completely loaded into frame-processor 0. Immediately, frame-processor 1 starts loading the next half frame of the RS-170 video signal. At time C_n, frame-processor completes the centroid computation and conversion of the centroid to from i, j to x, y coordinates. This information is sent to the Coordinator which uses an observer to filter the position data. The centroid data is converted back into image coordinates and sent to frame-processor 1 such that as soon as the second half frame has been loaded, the new position of the puck can be used in the new centroid operation.

Notice that if the observer is written in standard form (new estimates of the state at time n are based on state estimates from time n-1 and measurements from time step n), then the output of the observer at time O_n is an estimate of the states at time S_n. Since we can

only reliably complete the state estimation by time S_{n+2}, we would like the estimation scheme to output state estimates for the puck at time S_{n+2}. Further, the next measurement (shutter exposure) happens at S_{n+1}. We solve this problem in two steps. First, we write the observer in such a way that its output S_{n+2} is an estimate of the states at S_{n+1}. Next we use a simple predictor to carry the motion of the puck forward in time by the time (S_{n+1} - S_n).

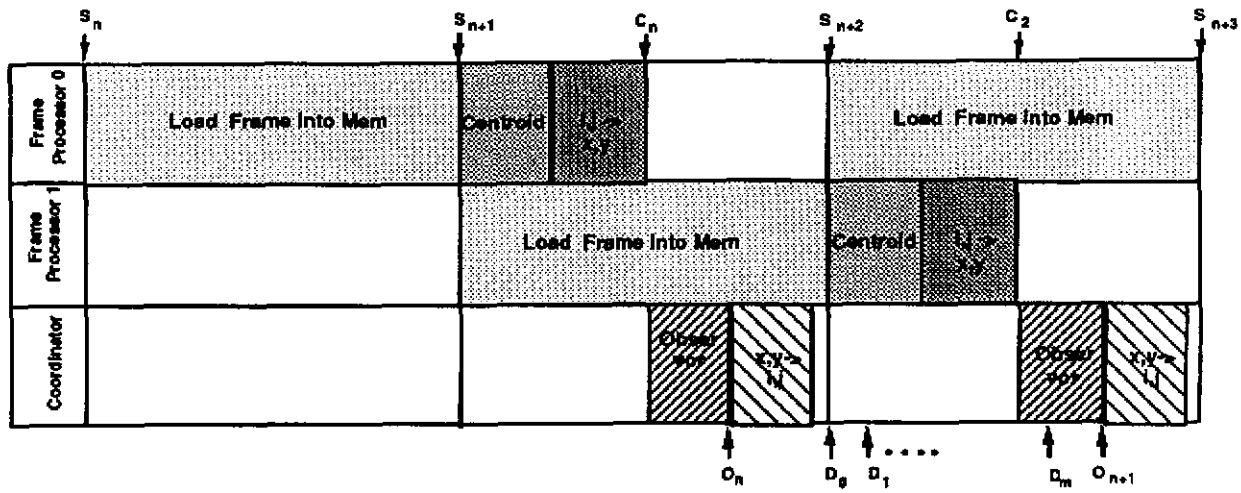

Figure 8: System timing

There is one further issue. We would like the observer to output at high rate (1 Khz), but the output of the observer is only once every 1/60th of a sec. To achieve this high rate between observer outputs, we use a simple interpolator that carries the equations of motion in time forward by a delta (= 0.001s) at each output time, D_m. (At time D_0, the first full state estimate (x and y positions and velocities) is available.) To actually achieve this, the processor that the coordinator runs on, must multitask between running the observer and running the interpolator at fixed time intervals. The interpolator thus is encoded as an interrupt task, that is, it is executed with a high priority at every 1ms.

The data flow in the entire process is shown in figure 9.

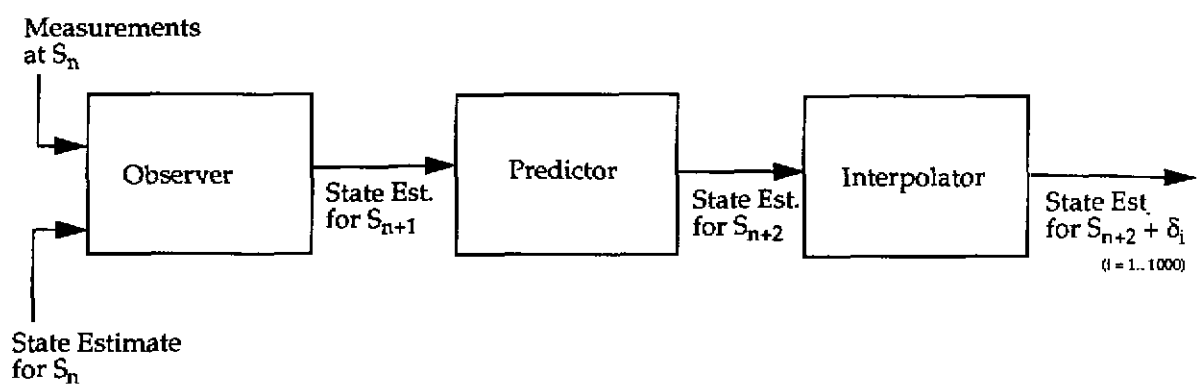

Figure 9: Data flow

5.3. Observer design

Writing the dynamics of the puck in state space form:

$$\dot{F} = A \cdot X + B \cdot u \tag{3}$$

$$G = C \cdot X \tag{4}$$

where F is the state vector, B is the scaling factor for the input, G is the output vector, C is the matrix that specifies the combination of the states for measurement, and u is the external input. More explicitly:

$$\dot{F} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -f_x & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -f_y \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} \cdot g \tag{5}$$

$$G = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} \tag{6}$$

where g_1 and g_2 are the actual measurements.

So far the system has been described in terms of continuous time dynamics. Since we use discrete measurements, the above system is best expressed as an equivalent difference equation where the A and B matrices have been appropriately transformed to Φ and Γ, given the sampling interval. The discretized system is written as:

$$F_{n+1} = \Phi \cdot F_n + \Gamma \cdot u_n \tag{7}$$

The input to our system will be simply the gravitational force, g, which is invariant. For a sampling rate of 60 hz, and for $f_x = 0.0$, $f_y = 0.16$, the appropriate difference equation is:

$$F_{n+1} = \begin{bmatrix} 0 & 0.016 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.016 \\ 0 & 0 & 1 & 0.997 \end{bmatrix} \cdot F_n + \begin{bmatrix} 0 \\ 0 \\ -0.001 \\ -0.016 \end{bmatrix} \cdot g \tag{8}$$

We write the observer in a form such that the output of the observer $\hat{F}_n$ is a prediction of the states at time step n+1 given measurements at time step n:

$$\hat{F}_{n+1} = \Phi \cdot \hat{F}_n + \Gamma \cdot g + L \cdot (G_n - (C \cdot \hat{F}_n)) \tag{9}$$

where L is the observer gain matrix that is computed by pole placement (poles of the observer are at 0.95, 0.955 for the x system and 0.4, 0.399 for the y system). The resulting L matrix is:

$$L = \begin{bmatrix} 0.095 & 0.0 \\ 0.135 & 0.0 \\ 0.0 & 1.198 \\ 0.0 & 21.473 \end{bmatrix} \tag{10}$$

The observer gain can be thought of a set of weights that are placed of the residual, or the error between the predicted and measured states. Essentially, pole placement is a way of encoding how much the measurements can be trusted in comparison to the dynamic model. If the measurements are noisy a relatively smaller weight should be placed on the residual. This is specified by choosing "slow" observer poles (close to, but less than unity). Alternatively, if the measurements are less noisy, faster poles (closer to zero) can be specified. In our system, the position sensing in the x direction exhibits more noise than position measurements in the y direction. Correspondingly, the poles chosen for the x system are slower than those chosen for the y system. It should be noted that observer design is an iterative process that is based on trail and error. Observer poles are chosen by an initial insight into the system and the resulting state estimates are evaluated. Sensitivity to noise and rates of convergence are traded off to arrive at the final observer gains.

5.4. Predictor

The task of the predictor is to compensate for the latency between the observer output and the actual puck states. This effect is achieved simply by integrating forward in time using (8) after every estimate of the observer. Thus, at O_n we have an estimate of puck states at S_{n+2}.

5.5. Interpolator

Since we would like to get state estimates at a high rate, we can repeatedly use the same method that the predictor uses, only this time integrating over a much smaller time interval. (11) gives the system dynamics as a difference equation discretized at a time interval of 0.001 s.

$$F_{n+1} = \begin{bmatrix} 0 & 0.001 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.0009 \\ 0 & 0 & 1 & 0.9998 \end{bmatrix} \cdot F_n + \begin{bmatrix} 0 \\ 0 \\ -0.0000005 \\ -0.0009999 \end{bmatrix} \cdot g$$

This allows for successive state estimates at D_i (i = 0.. 16)

5.6. Dealing with Impacts

Since the observer assumes a linear system, the impact which is highly non-linear, is treated as a disturbance. Thus, left to itself, the observer produces erroneous output for a short period after the impact, till the error between its estimates and measured positions

becomes large enough to significantly effect the state estimates. This effect can be seen in figure 10 in which the output of the y position as estimated is compared to the output of the same state by the grid sensor. In this example, it has taken roughly 6 sensing cycles for the observer to correct for the effect of the impact. For 3 updates after the position estimates are in the right direction, the sign of velocity is still negative. On its own, the linear observer treats the impact as a disturbance until repeated measurements (and the corresponding residuals) force the observer to the correct states.

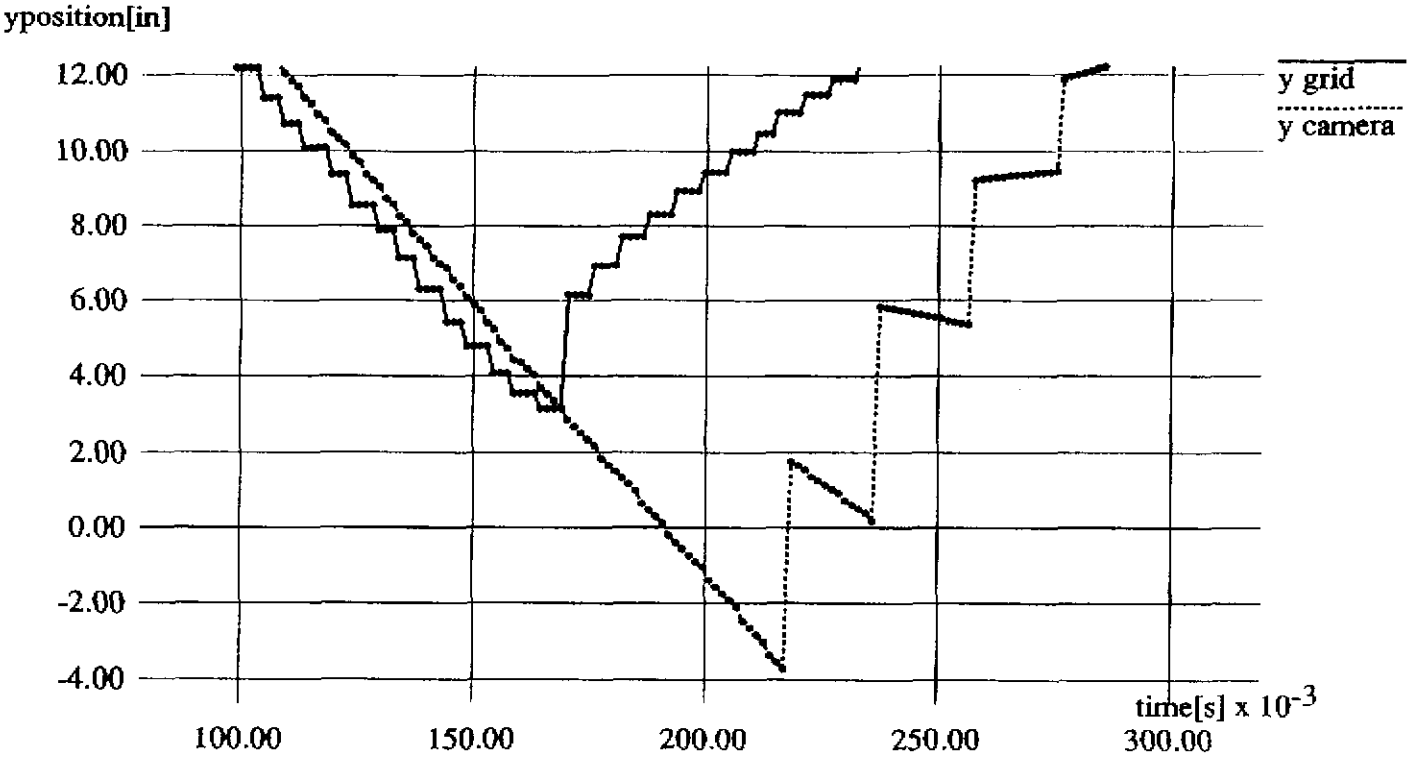

Figure 10: Output (y position) of the linear observer

In figure 10, the dashed line shows the output of the observer while the solid line denotes the output of the same state by the grid sensor. It turns out that for juggling it is most important to have good state information just before the impact whereas the error due to the linear observer does not start showing till just after the impact. Still, we would like to augment the linear observer with a method that would improve the state estimates just after the impact. A simple heuristic suffices:

if

$$(y < \varepsilon)\ and\ (\dot{y} < 0)$$

then

$$\dot{y} = \ddot{y}$$

Figure 11 shows the improvement in the observed states right after the impact.

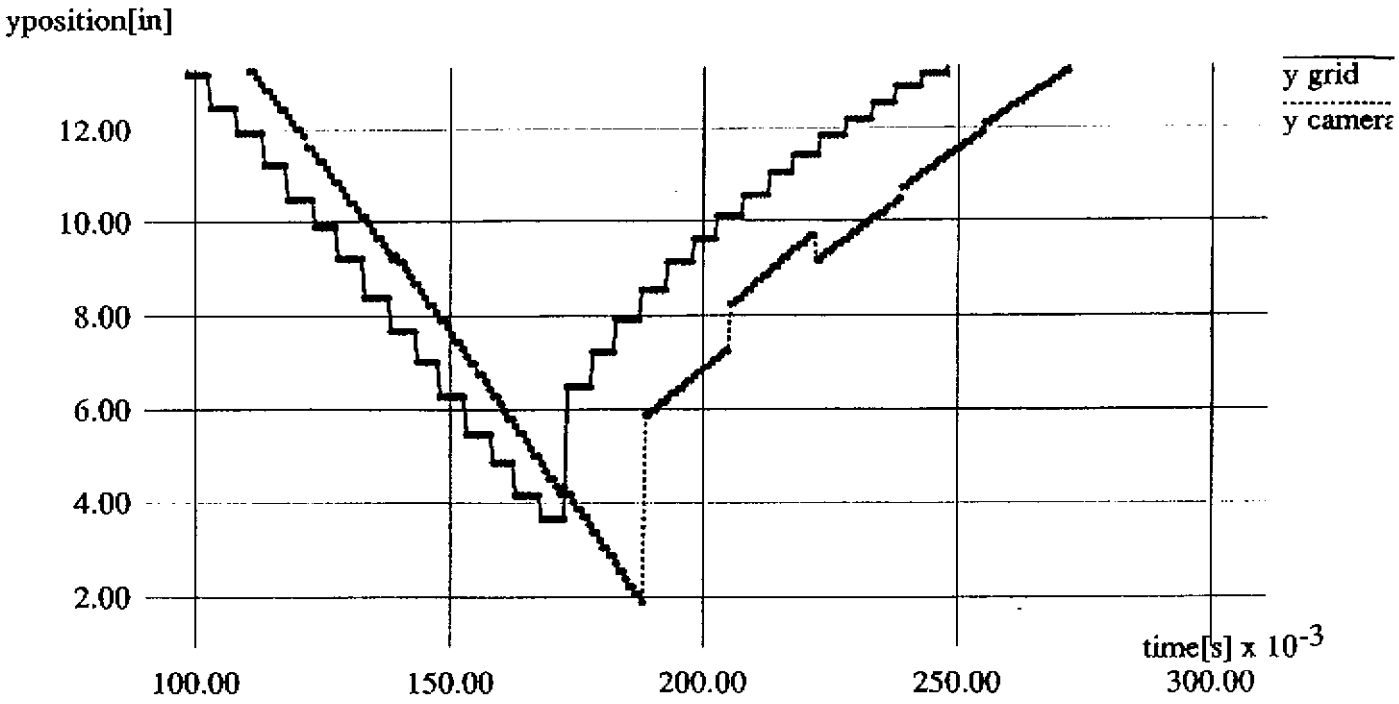

Figure 11: Output of linear observer (y position) with added heuristic

6. Results

Even though there is a large disparity in sampling rates and accuracy between the grid sensor and the vision sensor it is possible to qualitatively match the performance of the grid sensor with the vision sensor. Since it is difficult to compare each sensing mode to an absolute reference frame- i.e. to determine the errors of each sensing mode, in an absolute sense, the discussion of results is restricted to a comparison between sensing modes. In all the experiments discussed, the vision sensor was used to obtain position as well as to juggle. The grid sensor was run concurrently to collect data for purposes of comparison.

The results shown in figure 11 are typical. The best results (as evidenced by the smallest difference between the two sensing modes) are obtained exactly when the state estimates are the most important- just before impact. In this case, the steady state difference is around 1 inch (2.5 cm). After impact, it takes a while for the observer (discussed above) to produce estimates that match the grid sensor. This is mainly due to the large difference in sampling rates (60 hz for the vision sensor as opposed to 1000 hz for the grid sensor). Hence after impact, the difference in the two sensing modes can be as much as 6 inches (15 cm). This difference is made up quickly and the typical difference as the puck moves towards the peak, is around 2 inches (4.5 cm).

In some cases, as at the peaks of the y motion, the vision sensor provides a qualitatively better state estimate. By this, we mean that the vision sensor shows a more "natural" peak than does the grid sensor (figure 12(a)). Notice that the vision sensor produces discontinuities every so often. These correspond to the output of the observer when measurements are taken. The following points until the next discontinuity come from integration of the puck dynamics. Since the grid sensor gets data at a very high rate, the discontinuities seen in its output are smaller.

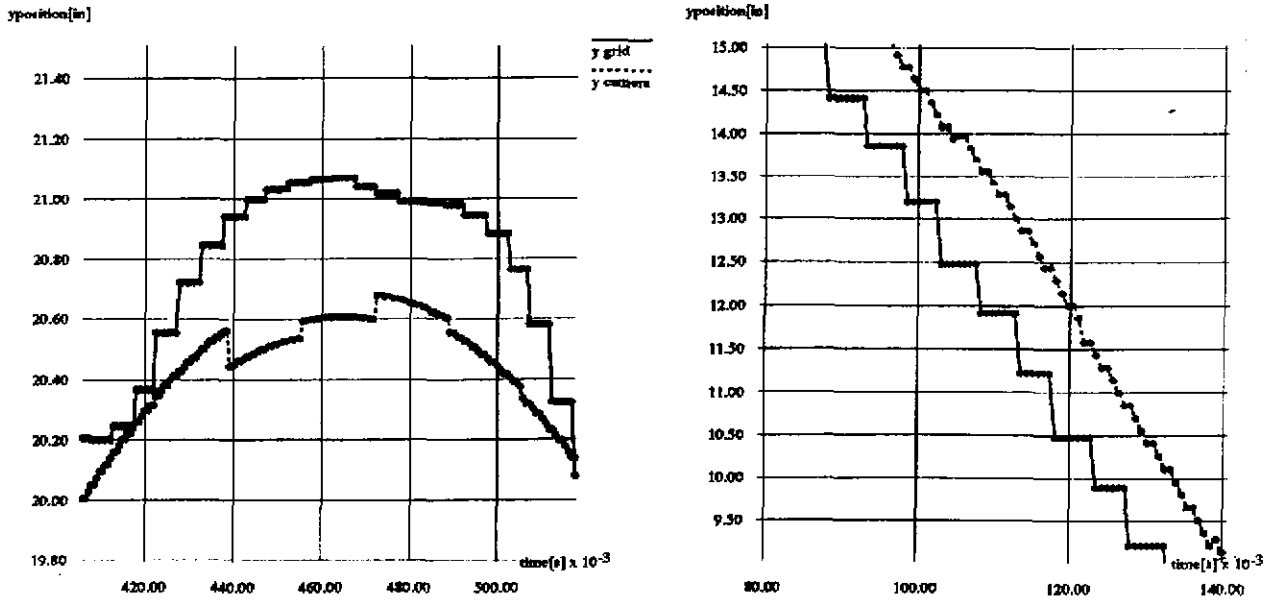

Figure 12: Comparison of position data (a) near the peak of y motion (b) during *downward flight*

Figure 13 compares the y velocity estimates by the two sensing modes. It can be seen that right after impact, there is a difference in the velocity estimates. This is once again due to the difference in sampling rate- the grid sensor detects the impact sooner than the vision sensor.

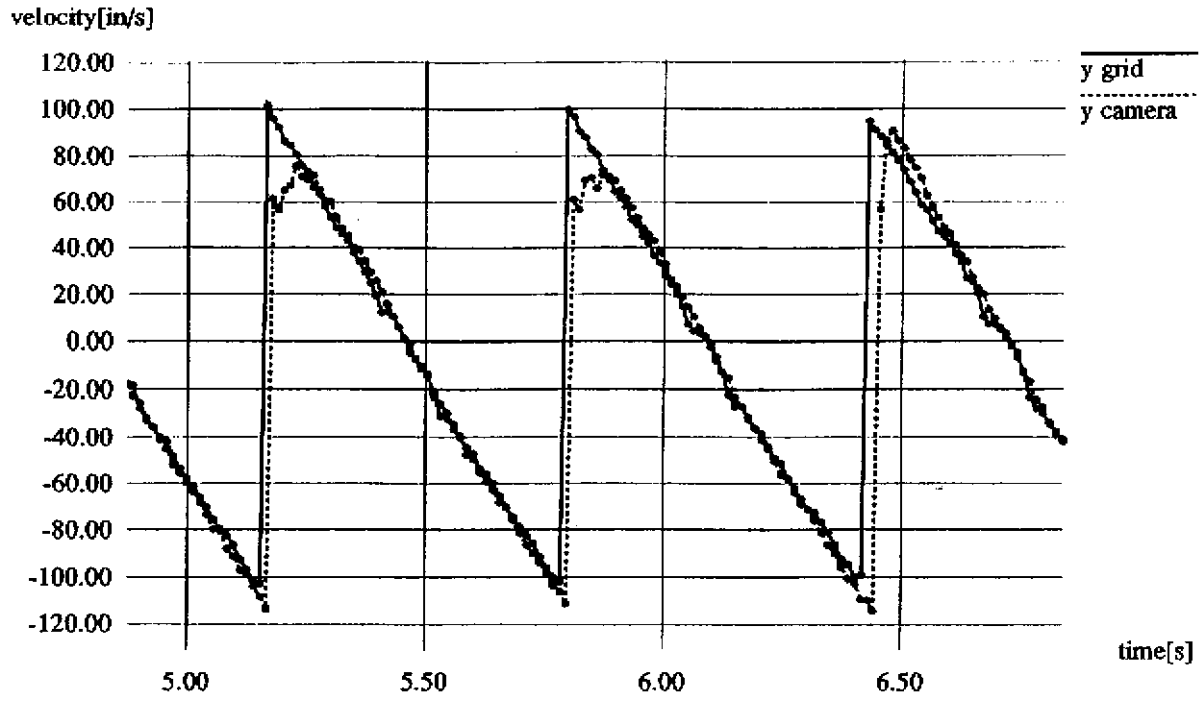

Figure 13: (y) Velocity estimates

Figure 14 compares the y position estimates for repeated juggles by the two sensing modes.

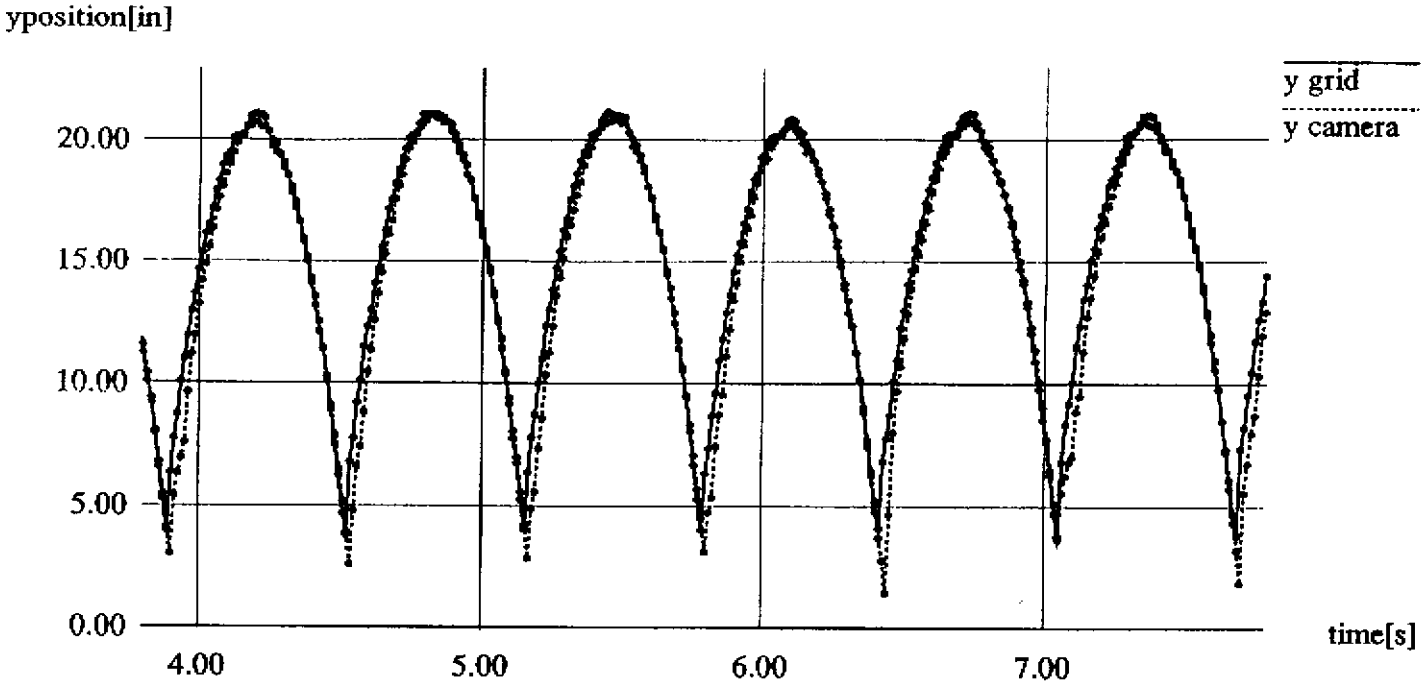

Figure 14: Comparison of steady state performance (y position) between vision and inductive sensing

Finally, the proof of the pudding is in how well the robot is able to juggle under each sensing mode. Our experiments with both sensing modes have show that the robot can juggle ad infinitum- our longest experiment with the vision sensor lasted 5 minutes without any degradation of performance.

7. Conclusions

A sensing scheme using machine vision has been demonstrated that tracks a moving object in real time. This scheme was tailored to produce full state estimates (positions and velocities) of a puck falling on an inclined plane such that it was possible to replicate previously demonstrated swat juggling. The advantage of this scheme is that it is not as contrived as its predecessor in which it was necessary to put active electronic oscillators inside each puck. Most importantly, the scheme scales nicely for tracking multiple objects- it is only necessary to replicate part of the hardware- each additional module is dedicated to tracking a single object. The scheme described can also be extended in a straight forward manner to tracking objects moving in 3 space.

Acknowledgments

Acknowledgment is due to Martin Buehler for his pioneering work in robot juggling. A big thank you to Al Rizzi and Louis Whitcomb for their untiring support with debugging the system. The author would like to acknowledge Dan Koditschek for the opportunity to do this work and the stimulating discussions that put the work in perspective.

References

[Andersson] R. Andersson, *A Robot Ping-Pong Player: An Experiment in Real-Time Control*, MIT Press, 1988.

[Atkeson] E. Aboaf, S. M. Drucker, C. G. Atkeson, *Task Level Robot Learning: Juggling a Tennis Ball More Accurately*, In *Proceedings IEEE International Conference on Robotics and Automation*, Cincinnati, OH, May, 1989.

[Buehler89] M. Buehler, D. E Koditschek, P. J. Kindlmann, A. Ganz, In *Proc. First International Symposium on Experimental Robotics*, Montreal, Canada, June, 1989.

[Buehler90a] M. Buehler, D. E. Koditschek, *From Stable to Chaotic Juggling: Theory, Simulation and Experiments*, In *Proc. IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, May 1989.

[Buehler90b] M. Buehler, *Robotic Tasks with Intermittent Dynamics*, Ph.D. Thesis, Yale University, 1990.

[Buehler90c] M. Buehler, D. E. Koditschek, P. J. Kindlmann, *A Family of Robot Control Strategies for Intermittent Dynamical Environments*, IEEE Control Systems Magazine 10(2): 16-22, Feb 1990.

[Cyclops] M. Buehler, N. Vlamis, C. J. Taylor, A. Ganz, *The Cyclops Vision System*, In *Proc. North American Transputer Users Group Meeting*, Salt Lake City, UT, Apr 1989.

[Tsai] R. Tsai, A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology Using Off the Shelf TV Cameras and Lenses, IBM Research Report, RC 11413, 1985.