REAL-TIME VISUAL SERVOING

Peter K. Allen
Billibon Yoshimi
Aleksandar Timcenko

Department of Computer Science
Columbia University
New York, NY 10027

ABSTRACT

This paper describes a new real-time tracking algorithm in conjunction with a predictive filter to allow real-time visual servoing of a robotic arm that is following a moving object. The system consists of two calibrated (but unregistered) cameras that provide images to a real-time, pipelined-parallel optic-flow algorithm that can robustly compute optic-flow and calculate the 3-D position of a moving object at approximately 5 Hz rates. These 3-D positions of the moving object serve as input to a predictive kinematic control algorithm that uses an $\alpha-\beta-\gamma$ filter to update the position of a robotic arm tracking the moving object. Experimental results are presented for the tracking of a moving model train in a variety of different trajectories.

1. INTRODUCTION

Tracking the three-dimensional movement of objects by a vision system in real-time is an important problem. It has been addressed by researchers in a number of different fields including target tracking, surveillance, automated guidance systems, inspection, and monitoring. The focus of our work is to achieve a high level of interaction between a real-time vision system that is capable of tracking moving objects in 3-D and a robot arm that contains a dexterous hand that can be used to intercept, grasp and pick up a moving object. We are interested in exploring the interplay of hand-eye coordination for dynamic grasping tasks such as grasping of parts on a moving conveyor system, assembly of articulated parts or for grasping from a mobile robotic system. However, the algorithms we have developed are quite general, and applicable to a wider range of domains including the ones described above.

The real-time tracking system we have built encompasses many of the components that are necessary for the creation of intelligent robotic systems. Specifically, the system is able to operate in real-time, at approximately human arm movement rates, using visual feedback in an active sense for a dynamically changing (as opposed to static) environment.

Previous efforts in the areas of motion tracking and real-time control are too numerous to exhaustively list here. We instead list some notable efforts that have inspired us or use similar approaches. Burt et al. [9] has focused on high speed feature detection and hierarchical scaling of images in order to meet the real-time demands of surveillance and other robotic applications. Related work has been reported by Lee and Wohn [23] and Wiklund and Granlund [35] who use image differencing methods to track motion. Corke, Paul and Wohn [13] report a feature based tracking method that uses special purpose hardware to drive a servo-controller of an arm-mounted camera. Goldenberg et al [14] have developed a method that uses temporal filtering with similar hardware to our own. Luo, Mullen and Wessel [24] report a real-time implementation of motion tracking in 1-D based on Horn and Schunk's method. Verghese et al. [33] report real-time, short-range visual tracking of objects using a pipelined system similar to our own. Safadi [30] uses a tracking filter similar to our own and a pyramid based vision system, but few results are reported with this system. Rao and Durrant-Whyte [29] have implemented a Kalman

---

This work was supported in part by DARPA contract N00039-84-C-0165, NSF grants DMC-86-05065, DCI-86-08845, CCR-86-12709, IRI-86-57151, IRI-88-1319, North American Philips Laboratories, Siemens Corporation and Rockwell Inc.

filter based de-centralized tracking system that tracks moving objects with multiple cameras. Miller [26] has integrated a camera and arm for a tracking task where the emphasis is on learning kinematic and control parameters of the system. Weiss et al [34] also use visual feedback to develop control laws for manipulation. Brown [8] has implemented a gaze control system that links a robotic ‘‘head’’ containing binocular cameras with a servo controller that allows one to maintain a fixed gaze on a moving object. Clark and Ferrier [12] also have implemented a gaze control system for a mobile robot.

An earlier work of ours [2], explored hand-eye interaction with a simple vision algorithm and a SCARA robot that performed tracking in the X-Y plane using a single calibrated camera that was mounted on the arm. The performance of this system was inherently limited by the vision algorithm and the lack of a predictive component in the tracking filter to allow the arm control to compensate for vision processing delays. It also was unable to track in three dimensions.

The system we have built addresses three distinct problems in real-time motion tracking: fast computation of 3-D motion parameters, predictive control of a moving robotic arm to track a moving object, and grasp planning which entails coordination with a dexterous robotic hand. The focus of this paper is on the first two problems.

This work is notable for the following reasons: First, the vision algorithm is robust since it is based upon detecting optic-flow in real-time as opposed to simple differencing or thresholding methods. Second, the system is capable of tracking objects in three dimensions, in real-time, using unregistered (but calibrated) cameras. Third, the robotic control system is able to accurately predict kinematic parameters for trajectory following of moving objects in real-time. We have tested the system on circular, oval and arbitrary trajectories with good results.

2. COMPUTING OPTIC-FLOW

In a visual tracking problem, motion in the imaging system has to be translated into 3-D scene motion. Our approach is to initially compute local optic-flow fields that measure image velocity at each pixel in the image. A variety of techniques for computing optic-flow fields have been used with varying results including matching based techniques [3, 10, 31], gradient based techniques [11, 18, 27] and spatio-temporal energy methods [1, 16]. Optic-flow was chosen as the primitive upon which to base the tracking algorithm for the following reasons:

- The ability to track an object in three dimensions implies that there will be motion across the retinas (image planes) that are imaging the scene. By identifying this motion in each camera, we can begin to find the actual 3-D motion.
- The principal constraint in the imaging process is high computational speed to satisfy the update process for the robotic arm parameters. Hence, we needed to be able to compute image motion quickly and robustly. The Horn-Schunck optic-flow algorithm (described below) is well suited for real-time computation on our PIPE image processing engine.
- We have developed a new framework for computing optic-flow robustly using an estimation-theoretic framework [32]. While this work does not specifically use these ideas, we have future plans to try to adapt this algorithm to such a framework.

Our method begins with an implementation of the Horn-Schunck method of computing optic-flow [17]. The underlying assumption of this method is the optic-flow constraint equation, which assumes image irradiance at time t and $t+\delta t$ will be the same:

$$I\,(\,x+\delta x,\, y+\delta y,\, t+\delta t\,) \;=\; I\,(x,\, y,\, t\,). \qquad (1)$$

If we expand this constraint via a Taylor series expansion, and drop second and higher order terms, we obtain the form of the constraint we need to compute normal velocity

$$I_x\, u + I_y\, v + I_t \;=\; 0 \qquad (2)$$

where u and v are the velocities in image-space, and I_x, I_y and I_t are the spatial and temporal derivatives in the image. This constraint limits the velocity field in an image to lie on a straight line in velocity space. The actual velocity cannot be determined directly from this constraint due to the aperture problem, but one can recover the component of velocity normal to this constraint line as:

$$V_n = -\frac{I_t}{\sqrt{I_x^2 + I_y^2}} \qquad (3)$$

While computationally appealing, this method of determining optic-flow has some inherent problems. First, the computation is done on a pixel by pixel basis, creating a large computational demand. Second, the information on optic flow is only available in areas where the gradients defined above exist. A second, iterative process is usually employed to propagate velocities in image neighborhoods, based upon a variety of smoothness and heuristic constraints.

We have overcome the first of these problems by using the PIPE image processor [4, 21]. The PIPE is a pipe-lined computer capable of processing 256x256x8 bit images at frame rate speeds, and it supports the operations necessary for optic-flow computation in a pixel-parallel method (a typical image operation such as convolution, warping, addition/subtraction of images can be done in one cycle - 1/60 second). The second problem is alleviated by our not needing to know the actual velocities in the image. What we need is the ability to locate and quantify gross image motion robustly. This rules out simple differencing methods which are too prone to noise and will make location of image movement difficult. Hence, a set of normal velocities at strong gradients is adequate for our task, precluding the smoothing step of the algorithm.

3. A REAL-TIME OPTIC-FLOW ALGORITHM

Our goal is to track a single moving object in real-time. We are using 2 static cameras that image the scene and need to report motion in 3-D to a robotic arm control program. Each camera is calibrated with the 3-D scene, but there is no explicit need to use registered (i.e scan-line coherence) cameras. Our method computes optic-flow fields in each camera and then use a triangulation to intersect the flow fields in areas of image motion in each camera. To implement our algorithm in the PIPE, we used 4 processors on the PIPE. The processors are assigned as 2 per camera - one each for the calculation of X and Y motion energy centroids in each image. We also use a special processor board (ISMAP) to perform real-time histogramming. The steps below correspond to the numbers in Figure 1 (single camera):

Figure 1: PIPE Motion Tracking Algorithm.

1. The camera images the scene and the image is sent to processing stages in the PIPE.

2. The image is smoothed by convolution with a Gaussian mask. The convolution operator is a built in operation in the PIPE and it can be performed in one frame cycle.

3-4. In the next 2 cycles, two more images are read in, smoothed and buffered, yielding smoothed images I_0 and I_1 and I_2. The ability to buffer and pipeline images allows temporal operations on images, albeit at the cost of processing delays (lags) on output. There are now 3 smoothed images in the PIPE, with the oldest image lagging by 3/60 second.

5. Images I_0 and I_2 are subtracted yielding the temporal derivative I_t.

6. In parallel with step 5, Image I_1 is convolved with a 3x3 horizontal spatial gradient operator, returning the discrete form of I_x. In parallel, the vertical spatial gradient is calculated yielding I_y (not shown).

7-8. The results from steps 5 and 6 are held in buffers and then are input to a look-up table that divides the temporal gradient at each pixel by the absolute value of the summed horizontal and vertical spatial gradients. This yields the normal velocity in the image at each pixel.

9-10. In order to get the centroid of the motion information, we need the X and Y coordinates of the motion energy. For simplicity sake we show only the situation for the X coordinate. The gray-value ramp in Figure 1 encodes the horizontal coordinate value (0-255) for each point in the image. If we threshold the computed normal velocities, and then AND the above threshold velocities with the positional ramp, we have an image which encodes high velocity with its positional coordinates in the image. In our experiments, we thresholded all velocities below 10 pixels per 60 msec. to zero velocity.

11. By taking this result and histogramming it, via a special stage of the PIPE which performs histograms at frame rate speeds, we can find the centroid of the moving object by finding the mean of the resulting histogram. Histogramming the high velocity position encoded images yields 256 16-bit values (a result for each intensity in the image). These 256 values can be read off the PIPE via a parallel interface in about 10 ms. This operation is performed in parallel to find the moving objects Y centroid (and in parallel for X and Y centroids for camera 2). The total associated delay time for finding the centroid of a moving object becomes 15 cycles or 0.25 seconds.

The same algorithm is run in parallel on the PIPE for the second camera. Once the motion centroids are known for each camera, they are back-projected into the scene using the camera calibration matrices and triangulated to find the actual 3-D location of the movement. Because of the pipelined nature of the PIPE, a new X or Y coordinate is produced every 1/60 second with this delay.

The system exhibits an interesting mix of local and global computations, separated by processors and update rates. The PIPE is able to perform the local optic-flow computation at video rates (but with delay), the ISMAP board gathers global histogram statistics, and these are then shipped via a high-speed interface to a host where the stereo triangulation and kinematic control algorithms reside, updating the arm parameters every 30 msec.

4. ROBOTIC ARM CONTROL

The second part of the system is the arm control. The robotic arm has to be controlled in real-time to follow the motion of the object, using the output of the vision system. The vision system output is not sufficient as a control parameter since its output is both noisy as well as delayed in time. The control system needs to do the following:

- Filter out the noise with a digital filter
- Predict the position to cope with delays introduced by both vision subsystem and the digital filter
- Perform the kinematic transformations which will map the desired manipulator's tip position from a Cartesian coordinate frame into joint coordinates, and actually perform the movement

We have adopted a simple, modular solution for the control subsystem where each task performs its job independently from the others.

This approach, although suboptimal, exhibits a high degree of modularity and, nonetheless, simplicity; characteristics which make a complex robotic system more robust and extensible.

4.1. DIGITAL FILTER

The 3-D values of the moving object's position determined by the vision system are both noisy and out of date due to the processing delays. To alleviate this, the system has been modeled as a standard second order system, and the input to the system is a series of filtered predictions. The need for signal smoothing emerged after our first experiments were conducted. Since the sampling period of the control subsystem is 30*ms* and the noise components were about 100*Hz*, we have adopted simple second-order filter with a transfer function in the s-domain [6]:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{(1+as)^2} \tag{4}$$

where $Y(s)$ is the Laplace picture of filter's output, $U(s)$ Laplace transform of filter's input with s being Laplace operator and a time constant selected experimentally. Since we have second-order filter, according to Bode's theory we have a 40 dB/decade decay of the filter's amplitude characteristics with a breaking frequency of $1/a$ Hz.

If we write (4) in the time domain, keeping in mind that s is related to $\frac{d}{dt}$ (i.e. time derivative) we get

$$y(t) + 2a\frac{d}{dt}y(t) + a^2\frac{d^2}{dt^2}y(t) = u(t) \tag{5}$$

By choosing state-space vector components x_1 and x_2 to be $x_1 = y$ and $x_2 = \dot{x}_1$ we get the canonical representation of our filter in the continuous-time domain:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{a^2} & -\frac{2}{a} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{a^2} \end{bmatrix} u \tag{6}$$

Now we need to get a discrete form of the system

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = E \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + Fu(k) \tag{7}$$

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

These yield the discrete-time matrices:

$$E = e^{AT} = \begin{bmatrix} (1+\frac{T}{a})e^{-T/a} & T\,e^{-T/a} \\ -\frac{T}{a^2}e^{-T/a} & (1-\frac{T}{a})e^{-T/a} \end{bmatrix} \tag{8}$$

$$F = \int_0^T e^{A\tau} B d\tau = \begin{bmatrix} 1-(1+\frac{T}{a})e^{-T/a} \\ Te^{-T/a} \end{bmatrix} \tag{9}$$

where

$$A = \begin{bmatrix} 0 & 1 \\ \frac{-1}{a^2} & \frac{-2}{a} \end{bmatrix} , \quad B = \begin{bmatrix} 0 \\ \frac{1}{a^2} \end{bmatrix} \tag{10}$$

are canonical model matrices, and T is the sampling period of the control subsystem. After substituting (8) and (9) into (7) we get the final form of our digital filter:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} (1+\frac{T}{a})e^{-\frac{T}{a}} & Te^{-\frac{T}{a}} \\ -\frac{T}{a^2}e^{-\frac{T}{a}} & (1-\frac{T}{a})e^{-\frac{T}{a}} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1-(1+\frac{T}{a})e^{-\frac{T}{a}} \\ Te^{-\frac{T}{a}} \end{bmatrix} u(k) \tag{11}$$

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \tag{12}$$

where $x_1(k)$ and $x_2(k)$ are state variables, and $y(k) = x_1(k)$ is the output variable. The elements of system matrices E and F are pre-computed off-line so the real-time computation burden is only 6 multiplications and 4 additions. The time constant a is experimentally set to 16*ms*.

4.2. PREDICTIVE FILTERING

In choosing a predictive filter for tracking, the major constraints are knowledge of the nature of the motion to be tracked, knowledge of the noise characteristics of the sensor, and computational cost of the filter. Two common approaches are auto-regressive models (ARM) [19] and Kalman filters [22]. Kalman filtering generates time-variable tracking coefficients that are determined by *a priori* models for the statistics of measurement noise and target dynamics. Because of its dynamic nature, it imposes a computational cost which can be difficult to incur in certain real-time applications. Fixed-coefficient filters have the advantage of simple implementation using fixed parameters for the filter gains [7]. In a few particularly simple target tracking problems, it is possible to derive closed-form steady-state solutions for the associated Kalman filter covariance matrix and the corresponding filter gains. Such solutions can be used to avoid real-time computation of the complete filter equations [20], and can approach the Kalman filter in steady-state performance.

We have implemented an $\alpha - \beta - \gamma$ filter [5, 30] which includes an acceleration estimate as well as a position and velocity estimate. Below are the prediction equations for the system (13-15) and the correction equations (16-18) using the filter parameters $\alpha - \beta - \gamma$:

$$x_{k+1|k} = x_{k|k} + T\, v_{k|k} + \frac{1}{2} T^2 a_{k|k} \quad (13)$$

$$v_{k+1|k} = v_{k|k} + T a_{k|k} \quad (14)$$

$$a_{k+1|k} = a_{k|k} \quad (15)$$

$$x_{k+1|k+1} = x_{k+1|k} + \alpha_{k+1} \left[z_{k+1} - x_{k+1|k} \right] \quad (16)$$

$$v_{k+1|k+1} = v_{k+1|k} + \frac{1}{T} \beta_{k+1} \left[z_{k+1} - x_{k+1|k} \right] \quad (17)$$

$$a_{k+1|k+1} = a_{k|k} + \frac{1}{2T^2} \gamma_{k+1} \left[z_{k+1} - x_{k+1|k} \right] \quad (18)$$

$x_{k|k}$ is the position estimate at time interval k

$x_{k+1|k}$ is the position estimate at time interval $k+1$ given k samples

$v_{k|k}$ is the velocity estimate at time interval k

$a_{k|k}$ is the acceleration estimate at time interval k

α_k is the position tracking parameter at k

β_k is the velocity tracking parameter at k

γ_k is the acceleration tracking parameter at k

W is world-coordinate frame

S is robot shoulder coordinate frame

M is 6th joint coordinate frame

T is tool (gripper) coordinate frame

G is grasping position coordinate frame

O is moving object coordinate frame

Base is constant transform between *W* and *S*

*T*6 is variable transform computed by RCCL in each sampling interval

Tool is variable transform defined by Utah-MIT hand kinematics

Drive is the transform introduced internally by RCCL to obtain straight-line motion in Cartesian coordinates

Grasp is constant transform which defines grasping point relative to the moving object

Obj is variable transform defined by vision subsystem outputs - it defines the position of the moving object in the world coordinate frame

Figure 2: Transform Equation. Graph nodes represent coordinate frames and graph edges represent coordinate transforms.

T is the sampling period

z_k is measured position at time k

The gains for the fixed coefficient filters represent a compromise between noise reduction and maneuver-following capability in the steady-state. Kalata [20] has shown that the optimal steady-state gains for a $\alpha - \beta - \gamma$ filter are given by:

$$\frac{\gamma^2}{4(1-\alpha)} = \lambda^2 \tag{19}$$

$$\beta = 2(2-\alpha) - 4\sqrt{1-\alpha} \equiv \alpha = \sqrt{2\beta} - \frac{1}{2}\beta \tag{20}$$

$$\gamma = \frac{\beta^2}{\alpha} \tag{21}$$

where λ is the *target maneuvering index* given as

$$\lambda = \frac{T^2\sigma_a}{\sigma_n} \tag{22}$$

where T is the sample period, σ_a is the position uncertainty due to variance of the acceleration, and σ_n is measurement noise variance. In practice, it is very difficult to accurately estimate the variances above. We treat this ratio of variances as a free parameter in calculating the filter gains, and have found a suitable value empirically.

To calculate α and β [5] we define $\eta = \sqrt{\beta}$ and from (20) we have:

$$\alpha = \sqrt{2}\eta - \frac{1}{2}\eta^2 \tag{23}$$

From (21) and (23) we have

$$\gamma = \frac{\eta^4}{\sqrt{2}\eta - \frac{1}{2}\eta^2} \tag{24}$$

Finally, we can insert (23) and (24) into (19) to get:

$$\lambda^2 = \frac{\left(\frac{\eta^3}{\sqrt{2} - \frac{1}{2}\eta}\right)^2}{4(1 - \sqrt{2}\eta + \frac{1}{2}\eta^2)} \tag{25}$$

which after a simple algebraic transformations resolves into a 6-degree polynomial equation in η with parameter λ:

$$\eta^6 - \lambda^2\left(\frac{1}{2}\eta^4 - 3\sqrt{2}\eta^3 + 13\eta^2 - 12\sqrt{2}\eta + 8\right) = 0 \tag{26}$$

Since we have $0 < \alpha < 1$, $\beta > 0$ and $\gamma > 0$, we may use a simple Newton iterative procedure with initial estimate η=0 to solve for polynomial

The updated information from the cameras is used to predict the next set point for the robotic arm that is tracking the moving object. We are using RCCL [15] to control the robotic arm (a PUMA 560). RCCL (Robot Control C Language) allows the use of C programming constructs to control the robot as well as defining transformation equations (as described in [28]). The transformation equations permit dynamic updating of arm position by generating the 4×4 transform of the moving object's position from the vision system and sending this information to the arm control algorithm (see Figure 2).

5. EXPERIMENTAL RESULTS

Figure 3 shows the experimental hardware that was used for the initial experiments of the system. It consists of 2 CCD cameras with 25mm lenses mounted approximately 1.5 meters apart, with the moving object's full trajectory in the field of view. In the experiments described below, the lenses were increased to 50mm and the baseline increased, with the cameras placed approximately orthogonal to each other. The cameras were calibrated using the two-plane method of Martins et al [25] in which a planar test pattern is imaged twice in each camera. The calibration was able to determine the three-dimensional position of the test patterns to within 3-4 millimeters in X, Y and Z coordinates. The experimental results were obtained with the cameras mounted approximately 1 meter above the plane of the table pointing down at an angle of about 10 degrees from horizontal.

In the experiments, a model train was tracked by vision as it moved around a circular track and an oval track and the arm was commanded to follow each trajectory. The train was moving at a velocity of approximately 25 cm/sec (the algorithm was tested at faster speeds and works with increasing overshoot in the control algorithm). The results shown below were obtained without the robotic hand mounted on the system. Figure 4 shows the left and right camera images of the moving train, and Figure 5 shows the thresholded motion-energy for each camera found by the algorithm. Figure 6 is the actual arm trajectory obtained by following the train as it moves on an oval track. The trajectory is quite close to the actual trajectory except as the camera distance to the moving object increases. The

Figure 3: Experimental Hardware

Figure 4: Left and right camera images

algorithm is less accurate here for two reasons: first, the inherent stereo digitization error increases with distance from the cameras, and secondly, the profile of the moving object is quite different at this position, yielding two views whose motion centroids may not correspond to the same exact point in space, thus inducing triangulation error. The tracking begins with oscillations until the filter kicks in fairly rapidly as shown. The path is for 5 revolutions and is very repeatable due to the robustness of the vision algorithm. At the end of the path the arm is commanded to smoothly end its motion. Figures 7, 8 and 9 are plots of the camera triangulated and predicted X, Y and Z trajectories of the train, showing the fidelity of the predictive component. The Z component of the train track was roughly planar with a variation in Z of about 5-7 mm. Figure 10 shows the trajectory of the arm when a the train moved in a circular path, again showing the initial oscillations and repeatable tracking once the filter has received enough data to fulfill its predictive capabilities. and

In addition to the trajectories above, we had the arm follow a semi-random trajectory created by a moving toy robot that changes its path direction arbitrarily when it hits an obstacle. In this experiment, the toy robot is placed inside the oval train track and it moves in a path until it hits the tracks, changing direction as it bounces off the tracks. While no ground truth is available for this trajectory, the arm follows the motion quite well, even though the robot moves in a way that is hard to model or predict. The results of these experiments are also available on video tape.

6. FUTURE WORK AND CONCLUSIONS

The algorithm described here has worked quite well in providing real-time visual servoing of a robotic arm, a fairly difficult task. We hope to continue to improve this algorithm, specifically by reducing the stereo error problems discussed above. One approach is to perform a better calibration of the camera system, and the other is try to better isolate centroids in each separate camera to be the same physical point. Various heuristics are being investigated on this front.

We also are interested in using a Kalman filter approach with dynamic gains for prediction. There is an added computational cost this approach, but it may yield more accurate trajectories, particularly with more complex trajectories. In addition, we are exploring tracking multiple objects using the PIPE's ability to work on a region by region basis.

Figure 5: Motion energy derived from optic flow (left and right cameras).

Figure 6: Tracked arm path, oval trajectory.

Finally, we are working on using the attached robotic hand to intercept and pick up the moving objects. An interesting approach we are currently exploring is to use vision to servo the hand and object together. The parallel processing capabilities of the PIPE may allow us to servo on both moving objects (target and robotic hand) to allow precise contact and grasping.

References

1. Adelson, E. H. and J. R. Bergen, ‘‘Spatio-temporal enrgy models for the perception of motion,’’ *Journal of the Optical Society of America*, vol. 2, no. 2, pp. 284-299, 1985.
2. Allen, Peter, ‘‘Real-time motion tracking using spatio-temporal filters,’’ *Proceedings of DARPA Image Understanding Workshop*, Morgan-Kauffman Publishers, Palo Alto, May 1989.
3. Anandan, P., *Measuring visual motion from image sequences,* Technical Report COINS-TR-87-21, COINS Department, University of Massachusetts, Amherst, 1987.
4. Aspex,, *PIPE User’s Manual*, Aspex Incorporated, 530 Spring St., New York, NY.
5. Bar-Shalom, Y. and T. Fortmann, *Tracking and data association,* Academic Press, 1988.
6. Bellman, Richard, *Introduction to the mathematical theory of control processes,* Academic Press, New York, 1967.
7. Blackman, Samuel, *Multiple-target tracking with radar applications,* Artech House, Dedham, MA , 1986.
8. Brown, Christopher, ‘‘Gaze controls with interaction delays,’’ *Proc. DARPA Image Understanding Workshop*, pp. 200-218, Morgan-Kauffman, Palo Alto, May 23-26, 1989.
9. Burt, P. J., J. R. Bergen, R. Hingorani, R. Kolczynski, W. A. Lee, A. Leung, J. Lubin, and H. Shvayster, ‘‘Object tracking with a moving camera,’’ *IEEE Workshop on Visual Motion*, pp. 2-12, Irvine, CA, March 20-22, 1988.
10. Burt, P. J., C. Yen, and X. Xu, ‘‘Multi-resolution flow-through motion analysis,’’ *Proceedings of the IEEE CVPR Conference*, pp. 246-252, 1983.
11. Buxton, B. F. and H. Buxton, ‘‘Computation of optic flow from the motion of edge features in image sequences,’’ *Image and*

Vision Computing, no. 2, 1984.

12. Clark, James J. and Nicola J. Ferrier, ‘‘Control of visual attention in mobile robots,’’ *IEEE Conference on Robotics and Automation*, pp. 826-831, Scottsdale, AZ, May 15-19, 1989.

13. Corke, Peter, Richard Paul, and K. Wohn, *Video-rate visual servoing for sensory-based robotics*, Technical Report, Grasp Laboratory, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

14. Goldenberg, Richard, Wan Chi Lau, Alfred She, and Alan Waxman, ‘‘Progress on the prototype PIPE,’’ *IEEE Conference on Robotics and Automation*, Raleigh, N. C., March 31 - April 3, 1987.

15. Hayward, Vincent and Richard Paul, ‘‘Robot manipulator control under UNIX,’’ *Proc. of the 13th ISIR*, pp. 20:32-20:44, Chicago, April 17-21, 1983.

16. Heeger, David, ‘‘A model for extraction of image flow,’’ *First International Conference on Computer Vision*, London, 1987.

17. Horn, B. K. P., *Robot vision*, M.I.T. Press, 1986.

18. Horn, B. K. P. and B. Schunck, ‘‘Determining optical flow,’’ *Artificial Intelligence*, vol. 17, pp. 185-203, 1983.

19. Houshangi, N., ‘‘Control of a robot manipulator to grasp a moving target using vision,’’ *IEEE International Conference on Robotics and Automation*, pp. 604-609, Cincinnati, May 13-18, 1990.

20. Kalata, Paul, ‘‘The tracking index: A generalized parameter for $\alpha-\beta$ and $\alpha-\beta-\gamma$ trackers,’’ *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-20, no. 2, pp. 174-182, March 1984.

21. Kent, E. W., M. O. Shneier, and R. Lumia, ‘‘PIPE: Pipelined image processing engine,’’ *Journal of Parallel and Distributed Computing*, no. 2, pp. 50-78, 1985.

22. Lee, S. and Y. Kay, ‘‘An accurate estimation of 3D position and orientation of a moving object for robot stereo vision: Kalman filter approach,’’ *IEEE International Conference on Robotics and Automation*, pp. 414-419, Cincinnati, May 13-18, 1990.

23. Lee, Sang Wook and K. Wohn, *Tracking moving objects by a mobile camera*, Techn-

Figure 7: X coordinates: Triangulated from vision (bullets) and predicted (straight lines), oval trajectory.

Figure 8: Y coordinates: Triangulated from vision (bullets) and predicted (straight lines), oval trajectory.

ical Report MS-CIS-88-97, University of Pennsylvania, Department of Computer and Information Science, Philadelphia, November 1988.

24. Luo, Ren C., Robert E. Mullen Jr., and Daniel E. Wessell, ‘‘An adaptive robotic tracking system using optical flow,’’ *IEEE Conference on Robotics and Automation*, pp. 568-573, Philadelphia, 1988.

25. Martins, H. A., J. R. birk, and R. B. Kelley, ‘‘Camera models based on data from two calibration planes,’’ *Computer Graphics and Image Processing*, vol. 17, pp. 173-180, 1981.

26. Miller, W. Thomas, ‘‘Real-time application of neural networks for sensor-based control of robots with vision,’’ *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 4, pp. 825-831, July/Aug 1989.

27. Nagel, H. H., ‘‘On the estimation of dense displacement vector fields from image sequences,’’ *Workshop on motion: Representation and Perception*, pp. 59-65, Toronto, 1983.

28. Paul, Richard, *Robot Manipulators*, MIT Press, Cambridge, MA, 1981.

29. Rao, B. S. Y. and H. F. Durrant-Whyte, ‘‘A fully decentralized algoprithm for multi-sensor kalman filtering,’’ Report OUEL 1787/89, Dept. of Engineering Science, University of Oxford.

30. Safadi, Reem Bassam, ‘‘An adaptive algorithm for robotics and computer vision application,’’ Tech. Report MS-CIS-88-05, Department of Computer and Information Science, University of Pennsylvania, January 1988.

31. Scott, G. L., ‘‘Four-line method of locally estimating optic flow,’’ *Image and Vision Computing, 5(2)*, 1986.

32. Singh, Ajit, ‘‘An estimation-theoretic framework for image-flow computation,’’ *Proc. International Conference on Computer Vision (ICCV-90)*, Kyoto, Japan, December 4-7, 1990.

33. Verghese, Gilbert, Karey Gale Lynch, and Charles R. Dyer, ‘‘Real-time motion tracking of three-dimensional objects,’’ *IEEE International Conference on Robotics and Automation*, Cincinnati, May 13-18, 1990.

34. Weiss, Lee E., Arthur Sanderson, and Charles P. Neuman, ‘‘Dynamic sensor-based control of robots with visual feedback,’’ *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 5, pp. 404-417, October 1987.

35. Wiklund, Johan and Gosta Granlund, ‘‘Tracking of multiple moving objects,’’ in *Time Varying Image Processing and Moving Object Recognition*, ed. V. Cappelini, pp. 241-249, Elsevier Science Publishers, 1987.

Figure 9: Z coordinates: Triangulated from vision (bullets) and predicted (straight lines), oval trajectory.