Automated Tracking and Grasping of a Moving Object with a Robotic Hand-Eye System

Peter K. Allen
Aleksandar Timcenko
Billibon Yoshimi
Paul Michelman

Department of Computer Science
Columbia University
New York, NY 10027

Technical Report CUCS-034-91

November 26, 1991

Abstract

Most robotic grasping tasks assume a stationary or fixed object. In this paper, we explore the requirements for tracking and grasping a moving object. The focus of our work is to achieve a high level of interaction between a real-time vision system capable of tracking moving objects in 3-D and a robot arm equipped with a dexterous hand that can be used 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. Coordination between an organism's sensing modalities and motor control system is a hallmark of intelligent behavior, and we are pursuing the goal of building an integrated sensing and actuation system that can operate in dynamic as opposed to static environments. The system we have built addresses three distinct problems in robotic hand-eye coordination for grasping moving objects: fast computation of 3-D motion parameters from vision, predictive control of a moving robotic arm to track a moving object, and grasp planning. The system is able to operate at approximately human arm movement rates, and we present experimental results in which a moving model train is tracked, stably grasped, and picked up by the system. The algorithms we have developed that relate sensing to actuation are quite general and applicable to a variety of complex robotic tasks that require visual feedback for arm and hand control.

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.

1 INTRODUCTION

The focus of our work is to achieve a high level of interaction between a real-time vision system capable of tracking moving objects in 3-D and a robot arm equipped with 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. Coordination between an organism's sensing modalities and motor control system is a hallmark of intelligent behavior, and we are pursuing the goal of building an integrated sensing and actuation system that can operate in dynamic as opposed to static environments.

There has been much research in robotics over the last few years that addresses either visual tracking of moving objects or generalized grasping problems. However, there have been few efforts that try to link the two problems. It is quite clear that complex robotic tasks such as automated assembly will need to have integrated systems that use visual feedback to plan, execute and monitor grasping.

The system we have built addresses three distinct problems in robotic hand-eye coordination for grasping moving objects: fast computation of 3-D motion parameters from vision, predictive control of a moving robotic arm to track a moving object, and grasp planning. The system is able to operate at approximately human arm movement rates, using visual feedback to track, stably grasp, and pickup a moving object. The algorithms we have developed that relate sensing to actuation are quite general and applicable to a variety of complex robotic tasks that require visual feedback for arm and hand control.

Our work also addresses a very fundamental and limiting problem that is inherent in building integrated sensing/actuation systems; integration of systems with different sampling and processing rates. Most complex robotic systems are actually amalgams of different processing devices, connected by a variety of methods. For example, our system consists of 3 separate computation systems: a parallel image processing computer, a host computer that filters, triangulates and predicts 3-D position from the raw vision data, and a separate arm control system computer that performs inverse kinematic transformations and joint-level servoing. Each of these systems has its own sampling rate, noise characteristics, and processing delays, which need to be integrated to achieve smooth and stable real-time performance. In our case, this involves overcoming visual processing noise and delays with a predictive filter based upon a probabilistic analysis of the system noise characteristics. In addition, real-time arm control needs to be able to operate at fast servo rates regardless of whether new predictions of object position are available.

The system consists of two fixed cameras that can image a scene containing a moving object (Figure 1). A PUMA-560 with a parallel jaw gripper attached is used to track and pick up the object as it moves (Figure 2). The system operates as follows:

1. The imaging system performs a stereoscopic optic-flow calculation at each pixel in the image. From these optic-flow fields, a motion energy profile is obtained that forms the basis for a triangulation that can recover the 3-D position of a moving object at video rates.

2. The 3-D position of the moving object computed by step 1 is initially smoothed to remove sensor noise, and a non-linear filter is used to recover the correct trajectory parameters which can be used for forward prediction, and the updated position is sent to the trajectory-planner/arm-control system.

3. The trajectory planner updates the joint level servos of the arm via kinematic transform equations. An additional fixed gain filter is used to provide servo-level control in case of missed or delayed communication from the vision and filtering system.

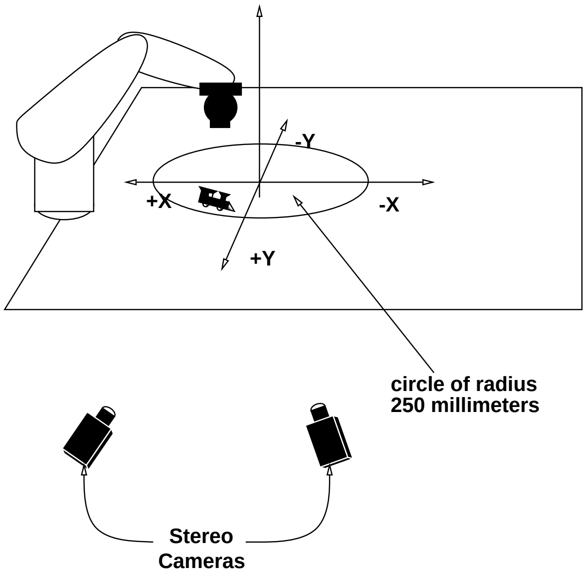


Figure 1: Tracking Grasping System

4. Once tracking is stable, the system commands the arm to intercept the moving object and the hand is used to grasp the object stably and pick it up.

The following sections of the paper describe each of these subsystems in detail along with experimental results.

2 PREVIOUS WORK

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. [10] 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 [30] and Wiklund and Granlund [46] who use image differencing methods to track motion. Corke, Paul and Wohn [14] report a feature-based tracking method that uses special purpose hardware to drive a servo controller of an arm-mounted camera. Goldenberg et al.[18] have developed a method that uses temporal filtering with vision hardware similar to our own. Luo, Mullen and Wessel [31] report a real-time implementation of motion tracking in 1-D based on Horn and Schunk's method. Verghese et al. [42] report real-time, short-range visual tracking of objects using a pipelined system similar to our own. Safadi [38] 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 [37] have implemented a Kalman filter-based de-centralized tracking system that tracks moving objects with multiple cameras. Miller [32] 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. [45] also use visual feedback to develop control laws for manipulation. Brown [9] 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

Figure 2: Experimental Hardware.

[13] also have implemented a gaze control system for a mobile robot. A variation of the tracking problems is the case of moving cameras. Some of the papers addressing this interesting problem are [15, 17, 47].

The majority of literature on the control problems encountered in motion tracking experiments is concerned with the problem of generating smooth, up-to-date trajectories from noisy and delayed outputs from different vision algorithms. Our previous work [4] coped with that problem in a similar way as in [39], using an $\alpha - \beta - \gamma$ filter, which is a form of a steady-state Kalman filter. A similar approach can be found in papers by [34, 29, 6]. In [34] a sophisticated control scheme is described which combines a Kalman filter's estimation and filtering power with an optimal (LQG) controller which computes the robot's motion. The authors have presented good tracking results, as well as stated that the controller is robust enough so the use of more complex (time-varying LQG) methods is not justified. The choice of gain matrices in the cost function and the best set of noise variances is done empirically. The work of Lee and Kay [29] addresses the problem of uncertainty of cameras in the robot's coordinate frame. The fact that cameras have to be strictly fixed in robot's frame might be quite annoying since each time they are (most often incidentally) displaced, one has to undertake a tedious job of their recalibration. Again, the estimation of moving object's position and orientation is done in the Cartesian space and a simple error model is assumed. Andersen et al. [6] adopts a 3rd-order Kalman filter in order to allow a robotic system (consisting of two degrees of freedom) to play the labyrinth game.

A somewhat different approach has been explored in the work of Papanikolopoulos et al. [35], Houshangi [24] and Koivo et al. [27]. The auto-regressive (AR) and auto-regressive moving-average with exogenous input (ARMAX) models are investigated. It is noteworthy to point out, as stated in [35], that this is more of an implementation than a conceptual difference from the classical Kalman-filter approach since the coefficients of polynomials in ARMAX model depend on the Kalman gains.

3 VISION SYSTEM

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 [5, 11, 40], gradient based techniques [23, 33, 12] and spatio-temporal energy methods [21, 2]. 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 [41]. 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 [22]. 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:

$$I(x + \delta x, y + \delta y, z + \delta z) = I(x, y, z) \tag{1}$$

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

$$I_x u + I_y v + I_t = 0 \tag{2}$$

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:

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

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 [26, 8]. The PIPE is a pipelined parallel image processing 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 need to iteratively propagate velocities in the image.

3.1 A REAL-TIME OPTIC-FLOW ALGORITHM

Our goal is to track a single moving object in real-time. We are using 2 fixed 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. Four processors are used in parallel 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 3.

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. These velocities are then thresholded and any isolated (i.e. single pixel motion energy) blobs are morphologically eroded.

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 3 encodes the horizontal coordinate value (0-255) for each point in the image. If we logically 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

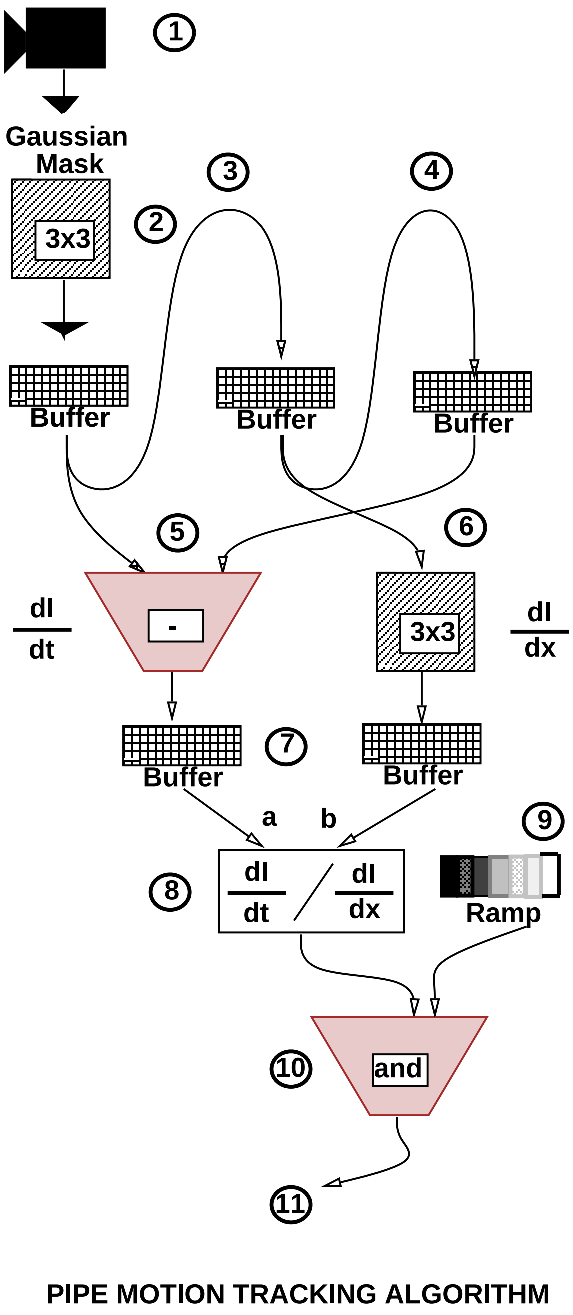


Figure 3: PIPE Motion Tracking Algorithm.

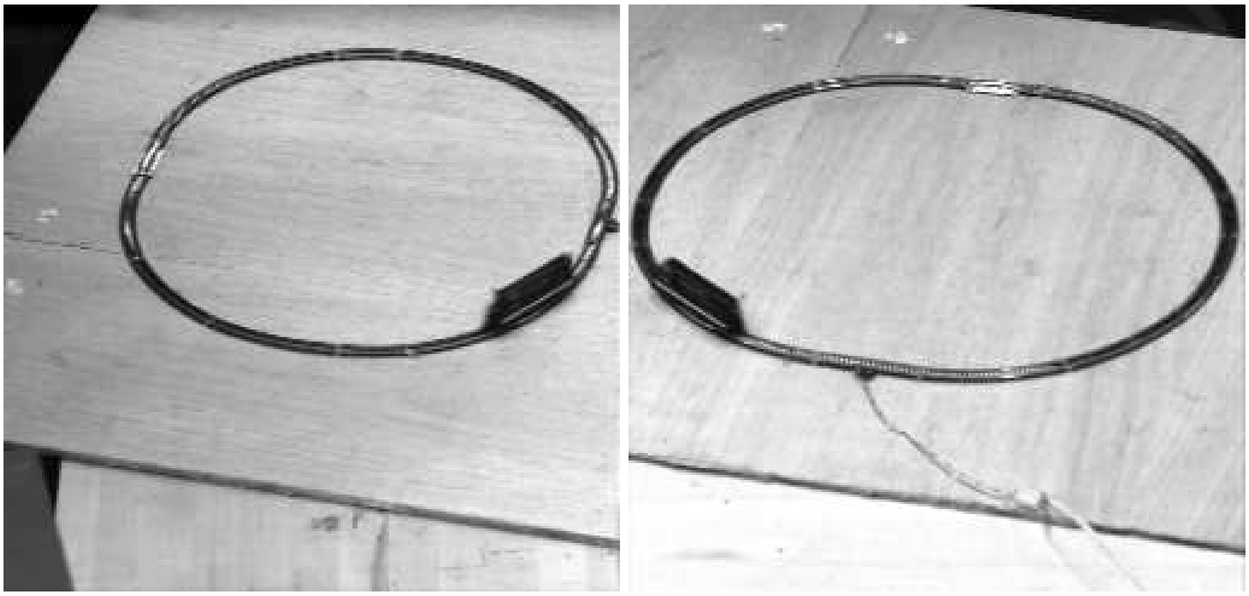


Figure 4: Left and right camera images.

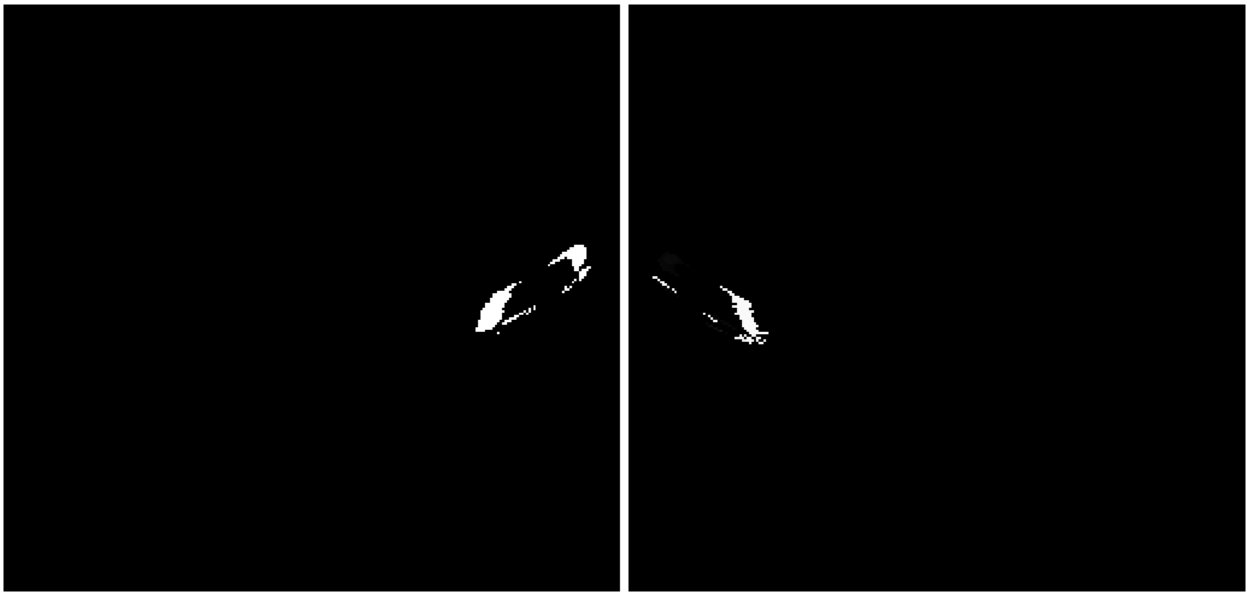


Figure 5: Motion energy derived from optic flow (left and right cameras).

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. Figure 4 shows 2 camera images of the moving train and Figure 5 shows the motion energy derived from the real-time optic-flow algorithm.

While we are able to derive 3D position from motion-stereo at real-time rates, there are a number of sources of noise and error inherent in the vision system. These include stereo-triangulation error, moving shadows which are interpreted as object motion (we use no special lighting in the scene) , and small shifts in centroid alignments due to the different viewing angles of the cameras, which have a large baseline. The net effect of this is to create a 3-D position signal that is accurate enough for gross level object tracking, but is not sufficient for the smooth and highly accurate tracking required for grasping the object. We describe in the next section how a probabilistic model of the motion that includes noise can be used to extract a more stable and accurate 3D position signal.

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 raw vision system output is not sufficient as a control parameter since its output is both noisy and 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

Our vision algorithm provides in each sampling instant a position in space as a triplet of Cartesian coordinates (x, y, z). The task of the control algorithm is to smooth and predict ahead the trajectory, thus positioning the robot where the object is during its motion.

A well known and useful solution is the Kalman filter approach, because it successfully performs both smoothing and prediction. However, the assumption the Kalman filter makes is that the noise applied to the system is white. That fact directly depends on the parametrization of the trajectory and, unfortunately in our case, the simplest possible parametrization - Cartesian- does not support this noise model. Our previous work [4] used a variant of this approach and obtained tracking that was smooth but not accurate enough to allow actual grasping of the moving object. Our solution to this problem was to appeal to a local coordinate system that was able to model the motion and system noise characteristics more accurately, thus producing a more accurate control algorithm.

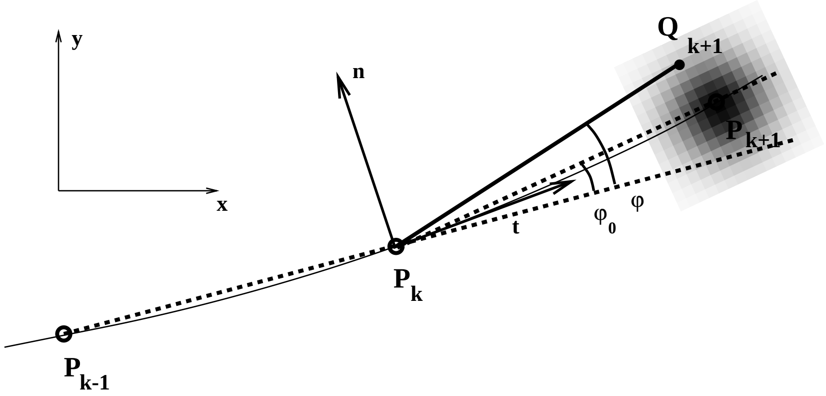


Figure 6: Trajectory: the moving object is in P_{k+1} while the vision computes Q_{k+1}

4.1 The Model of the 3D Motion

The main idea in the trajectory parametrization used in this paper is to describe a point in a *local* coordinate frame, relative to the point from the previous sampling instant, by the triplet of coordinates (s, ϕ, z) where

- s is the length of an arc between two points
- ϕ is the "bending" of the trajectory (see figure 6)
- z is the altitude difference in two consecutive points

Due to the existence of noise, all three coordinates are random variables with certain distributions. We have made the following assumptions, as a result of both reasoning about the vision algorithm and certain necessary simplifications:

- In sampling instant k our object is in point P_k
- In the next sampling instant $k+1$ the object is in P_{k+1} and the point returned by the vision algorithm is Q_{k+1}
- Q_{k+1} is normally distributed around P_{k+1}. The noise can be expressed by its two components, tangential n_t and normal n_n
- n_t and n_n are both zero-mean, with the same dispersion and mutually not correlated. Experimentally, it has been determined that their coefficient of correlation is between 0.1 and 0.2.

Under these assumptions it can be shown that (see Appendix A) the velocity v and curvature κ are:

$$v = \lim_{T \to 0} s/T \tag{4}$$

$$\kappa = \lim_{T \to 0} \tan \varphi_0 / s_0 \tag{5}$$

where $s_0 = ||P_{k+1} - P_k||$, $\varphi_0 = \pi - \angle P_{k-1} P_k P_{k+1}$ and T is the sampling interval.

The initial experiments with this model separates 3-D space into an XY plane and the Z axis, and addresses these two components of motion separately. However, the method for the XY plane can be extended to include another parameter which will create a full Frenet Frame at each instant

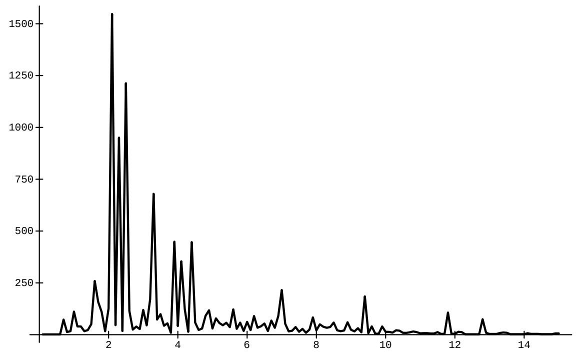


Figure 7: The density of s_1

of time in the trajectory. Our initial experiments (described below) tracked a planar curve, allowing us to use this simplification. Motion in the Z direction is tracked with a Cartesian displacement as outlined in [4].

Our model assumes the following coordinate transformation that relates the moving object's coordinate frame at one instant with the next instant in time:

$$\mathsf{Rot}(z, \phi_0) \circ \mathsf{Trans}(x, s) \circ \mathsf{Trans}(z, \triangle z) \tag{6}$$

where Rot and Trans are rotation about and translation along a given axis. Presented as a 4×4 matrix, transformation (6) is

$$T_{\mathsf{delta}} = \begin{bmatrix} \cos\phi_0 & -\sin\phi_0 & 0 & s\cos\phi_0 \\ \sin\phi_0 & \cos\phi_0 & 0 & s\sin\phi_0 \\ 0 & 0 & 1 & \triangle z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

What are advantages of such a parametrization? The most obvious one is the simplicity of the prediction task in this framework; all we need is to multiply the velocity $v = s/T$ by the time $\tau > T$ we want to predict ahead, as well as "bending" ϕ. The next advantage is that in order to achieve an accurate prediction, we do not need a high-order model with the mostly heuristic tuning of numerous parameters. The price we have to pay is that *filtering is not straightforward.* It turns out that we cannot just apply a low-pass filter in order to recover a DC component from s, but rather we need more elaborate approach which takes into account a probabilistic distribution of s. Figure 7 is a histogram of the experimentally measured density of the computed arc length between triangulated image motion points. This distribution shows the need to use a more sophisticated method than a simple averaging filter, which we have found to be incorrect in being able to correctly estimate the movement of the object between vision samples. The analysis below describes a probablilistic model that correctly models the experimental distribution in Figure 7, allowing us to recover the actual arc length parameter s_0 and the bending angle φ_0 at each sampling instant. While this model introduces more complexity than a standard Cartesian model, we will see below that it is more effective in allowing us to accurately predict and smooth our trajectory.

4.2 Probability Distributions of s and ϕ

In this section, we will motivate the choice of model used to recover the parameter values s_0 and φ_0 given the estimate of the arc length s which we calculate from the triangulated vision data.

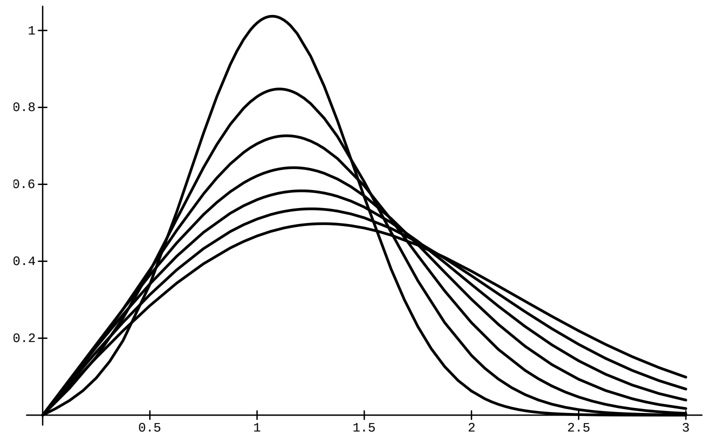


Figure 8: Distribution density $f(s), s_0 = 1, \sigma = 0.4 - 1.0, increment = 0.1$

Let $s = ||Q_{k+1} - P_k||$ be the distance between the object and the next position returned by the vision algorithm. According to figure 6 we have

$$\begin{aligned} s &= \left\| \begin{bmatrix} \cos\varphi_0 & \sin\varphi_0 \\ -\sin\varphi_0 & \cos\varphi_0 \end{bmatrix} \begin{bmatrix} n_t \\ n_n \end{bmatrix} + \begin{bmatrix} s_0 \\ 0 \end{bmatrix} \right\| = \\ &= \sqrt{(n_t' + s_0)^2 + n_n'^2} \end{aligned} \tag{7}$$

where n_t' and n_n' are Gaussian with dispersion σ. According to the definition of the probability distribution, we can write the distribution $F(s)$ as

$$F(s) = \iint_D \frac{1}{2\pi\sigma^2} \mathsf{e}^{-\frac{1}{2}\left[\left(\frac{t-s_0}{\sigma}\right)^2 + \left(\frac{n}{\sigma}\right)^2\right]} \mathsf{d}t\mathsf{d}n \tag{8}$$

where D is a disk of the radius s.

Now by introducing substitution $t = r\cos\theta$, $n = r\sin\theta$ we get

$$F(s) = \frac{1}{2\pi\sigma^2} \int_0^s r \int_0^{2\pi} \mathsf{e}^{-\frac{1}{2}\left[\left(\frac{r\cos\theta - s_0}{\sigma}\right)^2 + \left(\frac{r\sin\theta}{\sigma}\right)^2\right]} \mathsf{d}\theta\mathsf{d}r \tag{9}$$

Distribution density is given as $f(s) = \frac{\mathsf{d}F(s)}{\mathsf{d}s}$ or after differentiation

$$f(s) = \frac{s\mathsf{e}^{-\frac{s^2+s_0^2}{2}}}{2\pi\sigma^2} \int_0^{2\pi} \mathsf{e}^{-\frac{ss_0}{\sigma^2}\cos\theta} \mathsf{d}\theta \tag{10}$$

The last integral can be expressed by a modified Bessel function $I_0(z)$:

$$f(s) = \frac{s}{\sigma^2} \mathsf{e}^{-\frac{s^2+s_0^2}{2\sigma^2}} I_0\left(\frac{ss_0}{\sigma^2}\right) \tag{11}$$

A graph of $f(s)$ is given in figure 8. Here s_0 is fixed to 1 and σ varies from 0.4 to 1.0. Our job is to recover s_0 given $f(s)$.

It is apparent from the figure 8 that the peak value of $f(s)$ depends on σ, and drifts towards higher values as σ grows. The expectation for s also depends on σ. In particular, we have

$$s_1 = E(s) = \int_0^\infty sf(s)\mathsf{d}s = \sigma u\left(\frac{s_0}{\sigma}\right) \tag{12}$$

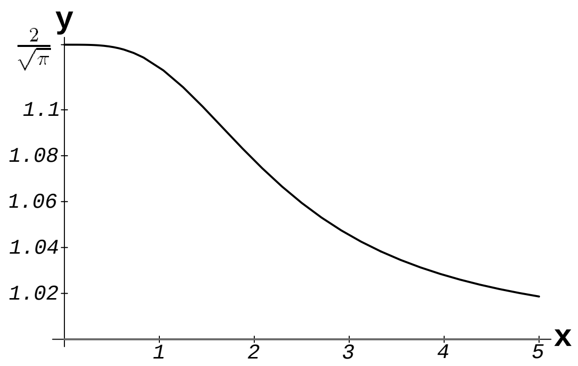


Figure 9: $y = u_1(x)$

where

$$u(x) = \sqrt{\frac{\pi}{2}} \mathsf{e}^{-x^2/4} (I_0(\frac{x^2}{4}) + \frac{x^2}{2}(I_0(\frac{x^2}{4}) + I_1(\frac{x^2}{4}))) \tag{13}$$

Here σ is the constant for the given system and it is related to s_0. In order to estimate σ we will use second-order moment:

$$s_2^2 = E(s^2) = \int_0^\infty s^2 f(s) \mathsf{d}s = s_0^2 + 2\sigma^2 \tag{14}$$

Equations 12 and 14 are derived in appendix B.

Now by eliminating s_0 from (12) and (14) we have

$$1 = z \; u\left(\frac{\sqrt{p^2 - 2z^2}}{z}\right) \tag{15}$$

where $p = s_2/s_1$ and $z = \sigma/s_1$. Now by setting $x = \frac{\sqrt{p^2-2z^2}}{z}$ we end up with an equation (see appendix C)

$$u_1(x) = \frac{\sqrt{x^2+2}}{u(x)} = p \tag{16}$$

Equation 16 relates our known control inputs ($p = s_2/s_1$) to x. We can create a table of values for this function offline, and then by interpolation calculate a value of x given p.

Let $x_0(p)$ be the solution of (16). Now we can express s_0 and σ as functions of s_1 and s_2 as follows:

$$s_0 = s_2 \frac{x_0\left(\frac{s_2}{s_1}\right)}{\sqrt{2 + x_0\left(\frac{s_2}{s_1}\right)^2}} \tag{17}$$

$$\sigma = s_2 \frac{1}{\sqrt{2 + x_0\left(\frac{s_2}{s_1}\right)^2}} \tag{18}$$

This method requires little on-line computation - an interpolation table of values of u_1 is all we need to recover the arc length parameter s_0. Figure 7 is the experimentally measured density of s_1 taken from the triangulated optic-flow fields. This distribution's resemblance to figure 8 (the theoretical density) is clear.

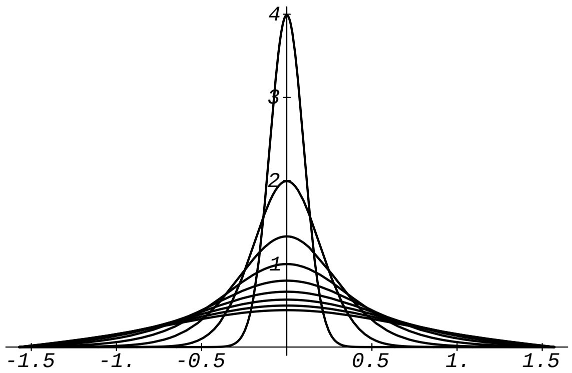


Figure 10: Distribution density $f(\phi)$

To find the bending parameter ϕ_0, we use the same technique as for the distribution of s, and we get the following formula:

$$f(\phi) = \frac{\cos(\phi - \phi_0)}{\sqrt{2\pi}k} \mathrm{e}^{-\frac{\sin^2(\phi-\phi_0)}{2k^2}}$$

where $k = \sigma/s_0$ and $\phi - \phi_0 \in (-\pi/2, \pi/2)$. It is obvious that f is symmetric around ϕ_0, which also means that the expectation $E\phi = \phi_0$. Hence, we so not need to perform a non-linear filtering to recover ϕ_0.

The graph of f for $k = 0.1$ to 0.9 and $\phi_0 = 0$ is given in figure 10.

4.3 Smoothing of the Control Inputs

In the previous section, we showed how to extract parameters s_0 and ϕ_0 from the updated positions determined from the vision system. The signals s_1, s_2^2 described in equations 12 and 14 are in fact the smoothed versions of the expectations of the control signals s, s^2 which are the arc length and the arc length squared. The smoothing filter we use to compute these signals is a moving-average (MA) filter using a Kaiser window [25]. This filter provides the largest ratio of signal energy in the main lobe and a side lobe, which usually results in a filter of lower order. The windowing function is given by

$$w_K(n) = \frac{I_0(\beta\sqrt{1-(1-2n/M)^2})}{I_0(\beta)}$$

where I_0 is the modified zeroth-order Bessel function, β is the shape parameter which defines the width of the main lobe and M is the order of the filter. According to [25], β and M are given by

$$M \approx \frac{A - 7.95}{14.36\triangle\omega}$$

and

$$\beta = \begin{cases} 0.1102(A-8.7), & A \geq 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21), & 21 < A < 50 \end{cases}$$

where A is the stopband attenuation and $\triangle\omega = (\omega_r - \omega_c)/\omega_s$, ω_r is the stopband frequency, ω_c is the passband frequency and ω_s is the sampling frequency.

We have adopted $A = 30$ and $\triangle\omega = 0.05$ which results in $M = 30$. Since the frequency of the vision algorithm is about 60 Hz, the overall length of the window is about 0.5 seconds. We also apply this MA filter to the bending parameter ϕ.

The implementation of MA filter is straightforward: once the weights are computed off-line, a window of length M of measurements is retained and each sample is multiplied by an appropriate weight in the sampling period, which requires M multiplications and $M-1$ additions. This allows reasonably wide windows (even up to several hundreds entries) to be used in computing the smoothed signal.

4.4 Prediction and Synchronization

The host computer controls the initial vision processing and subsequent computation of control parameters described above. The host computer is able to predict ahead the trajectory using the derivation of velocity and curvature in equations (4) and (5). These updated predictions are sent to the trajectory generator that is actually controlling the robot arm. The trajectory generator is a separate system that has two parallel tasks: a low-priority task which reads the serial line receiving updated control signals and high-priority task which calculates the transformation equation and moves the manipulator. Those two tasks communicate via shared memory. The job of the robot controlling program is to synchronize its two tasks (i.e. to obtain mutual exclusion in accessing shared data), to unpack input packets read from the serial line, and to update the joint servos every 30 msec.

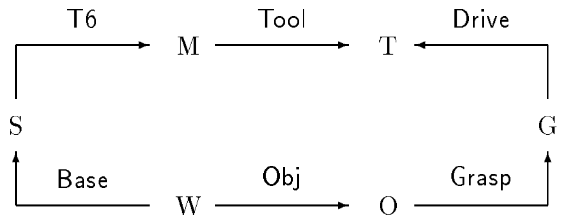
The asynchronous nature of the communication between the host computer and the trajectory generator can result in missed or delayed communications between the two systems. Since the updating of the robotic arm parameters needs to be done at very tightly specified servo rates (30 msec), it is imperative that the trajectory generator can provide updated control parameters at these rates, regardless of whether it has received a new control input from the host. Therefore, we have implemented a fixed gain $\alpha - \beta - \gamma$ filter as part of the trajectory generator [39]. This filter provides a small amount of prediction to the trajectory parameters if the control signals from the host are delayed.

We are using RCCL [20] 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 [36]). 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 11).

5 MOTOR COORDINATION FOR GRASPING

The remaining part of our system is the interception and grasping of the object. We have examined the human psychological literature in order to find useful paradigms for robotic visual-motor coordination strategies that include arm movement and grasping from visual inputs. In this section we briefly describe some relevant theories and their relation to our own work.

There are several theories on the organization of skilled human motor control. Richard Schmidt [19] has proposed a theory of generalized motor programs, or movement *schemas*. In this view, a skilled action is composed of an ordered set of parametrized motor control programs of short duration (less than 200 msec), each of which accomplishes one part of the task. As one program is completed, the next one is executed. Generalized motor programs accomplish several objectives: (1) they specify *which muscle* to move in a given motion; (2) the *order of contraction* of the muscles; (3) the *phasing* within the sequence, i.e., the temporal relationships among the contractions; (4) the *relative force* of each element. At the initiation of a skilled task, the parameters of the motor control program are determined by sensory input and task demands, and then the programs are executed to completion. If the wrong program is selected for some reason, the program cannot be stopped by use of sensory information. An example of this can be seen in the motor activity associated with



Graph nodes represent coordinate frames:

- 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

Graph edges represent 4×4 coordinate transforms:

- Base is constant transform between W and S
- T6 is variable transform computed by RCCL in each sampling interval
- Tool is variable transform defined by the 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 11: Transform Equation.

playing table tennis. In moving the arm to hit the ball, the motion of the racket is determined before the beginning of the swing and visual input has little effect after the initiation of motion. As an example of Schmidt's theory, the skilled task of grasping a moving object could be partitioned into two motor control schemas: one to position the arm and a second one to control the grasping action.

The schema concept maps into Von Hofsten's ideas about the development of grasping skills in children [44] He believes there are two separate sensorimotor systems responsible for reaching: one for approaching the target and one for grasping it. During early childhood, the precise timing between these two systems develops as the child learns how to catch. The reaching system develops first, before a child is capable of grasping. But even before he is capable of closing his hand at precisely the right moment, he has begun to develop the ability to move his hand toward a moving object and predict the location at which his hand will intercept the object. With growth, a child learns to control the timing between reaching and grasping, that is, to close his hand at the correct moment. Experimental evidence has shown that there is a window of approximately 14 msec during which the hand must begin closing. Unlike Schmidt, however, Von Hofsten does not consider vision and grasping to be two mutually exclusive tasks [43] Visual tracking is used to guide the reaching arm *during* its motion, not only before motion. A coordinated motion is a combination of perceptual schemas and motor schemas (see Iberall and Arbib [7]).

Vision is used during the reaching phase of the task for what psychologists call "prospective control". Prospective control corresponds to predictive filtering, as used by control theorists. In grasping a moving object, it is necessary for the hand to move not to the current position of the object, but to plan ahead to where it will be shortly. Vision, rather than haptics, provides the basis of prospective control because touch cannot provide the anticipatory information required to predict the course of a moving object. There are two predominant theories about what visual schema is used to track a moving object and aid in predicting the intersection of the reaching hand and that object. Lee [28] proposes the use of vision to measure the expansion of the image on the retina in order to estimate the time until contact. The attraction of this theory is that humans would not need to compute the velocity and location of the moving object, but would calculate the more useful time-until-contact information. A person catching an object uses this image to compute when to begin the correct motion commands (usually at about 300 msec before the actual grasp). Von Hofsten disputes the use of retinal expansion information because it is clear that people are able to track targets in which there is no such expansion, such as objects that are circling or passing across the field of view. He suggested an alternative schema in which people calculate the distance to a moving object by using the vergence angle to the object. Vision seems to be used predominantly to track the moving object, but the catcher also tracks his hand during reaching to aid his nonvisual proprioceptive senses, that is, to help judge the position of his hand in relation to the environment. Finally, vision must be used during the reaching phase to orient the hand correctly in relation to the object that is being caught.

We also note a relevant fact for human contact and grasping of objects. The central factor to the final grasp is the time of the onset of hand closure. In early childhood (up to about 5 months), closing the hand is triggered primarily by touch. Children tend to begin grasping only when they are already in contact with the object. By the time a child is 13 months old, however, the hand begins closing before touch. We take the view in this paper that our robotic system is past early childhood - we will begin closing the hand before actual contact is made.

The initial strategy we have adopted in picking up the object is an open loop strategy, similar in spirit to the pre-programmed motor control schemas described in the psychological literature. Schmidt's schema theory holds that for tasks of short duration, perception is used to find a set of parameters to pass to a motor control program. It is not used during the execution of a task. When grasping a moving object, for example, once vision determined the trajectory of the object, the reach and grasping motor schemas take over with no interference from vision.

In our implementation of this strategy, vision is not used to continually monitor the grasping, but only to provide a final position and velocity from which the arm is directed to very quickly move to the object. This automatic movement is done by establishing coordinate frames of action for each of the components of the system and solving transformation equations (see Figure 11).

The transformation equations permit dynamic updating of the 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. This positional information from the vision system is used to update the **Obj** transform in Figure 11. The other transforms in the equation are known, and this allows the system to solve for the **Drive** transform which is the transform used to update the manipulator's joints and develop a straight line path in Cartesian coordinates that will bring the hand into contact with the moving object. Because the movement of the hand requires a small amount of time during which the object may have moved, the object's trajectory is predicted ahead during the movement using the $\alpha - \beta - \gamma$ predictor. By keeping the fingers of the hand spread during this maneuver, no actual contact takes place until the gripper reaches the position of the moving object. Once this position is achieved, the gripper is commanded to close and grasp the object.

6 EXPERIMENTAL RESULTS

We have implemented the system described above in order to demonstrate the capability of the methods. The goal was to track a moving model train, intercept it, stably grasp it and pick it up. The train was moving in an oval trajectory; however, the system had no *a priori* knowledge of this particular trajectory. The velocity of the train was 10-20 cm/s. In this section we present some results obtained by experiments. First, in figure 12 we have the actual measured arc length signal s_1 (black) and the filtered signal s_0 (gray). It is noticeable that s_0 is somewhat *below* the expected value of s_1. The nature of s_1 is quite noisy; however, the analysis described in section 4 was able to accurately extract the correct control signal. The arm control is particularly smooth and jerk free, stable over time (the tracking is continuous for many revolutions of the train) and is highly accurate in being able to intercept and grasp the object between the jaws of the gripper as it moves. Figure 13 shows the moving object's trajectory points computed by the vision algorithm (black) and the commanded control signals after filtering (gray). As can be seen, the control system is able to accomplish its task of both smoothing for noise and extracting an accurate position of the moving object.

Because we are using a parallel jaw gripper, the jaws must remain aligned with the tangent to the actual trajectory of the moving object. This tangential direction is computed directly from the calculation of the bending parameter φ during the trajectory modeling phase and is used to align joint 6 of the robot to keep the gripper correctly aligned. This correct alignment allows grasping to occur at any point in the trajectory.

Figure 14 shows 3 frames taken from a video tape of the system intercepting, grasping and picking up the object. The system is quite repeatable, and is able to track other arbitrary trajectories in addition to the one shown.

7 SUMMARY AND FUTURE WORK

We have developed a robust system for tracking and grasping moving objects. The system relies on real-time stereo triangulation of optic-flow fields and is able to cope with the inherent noise and

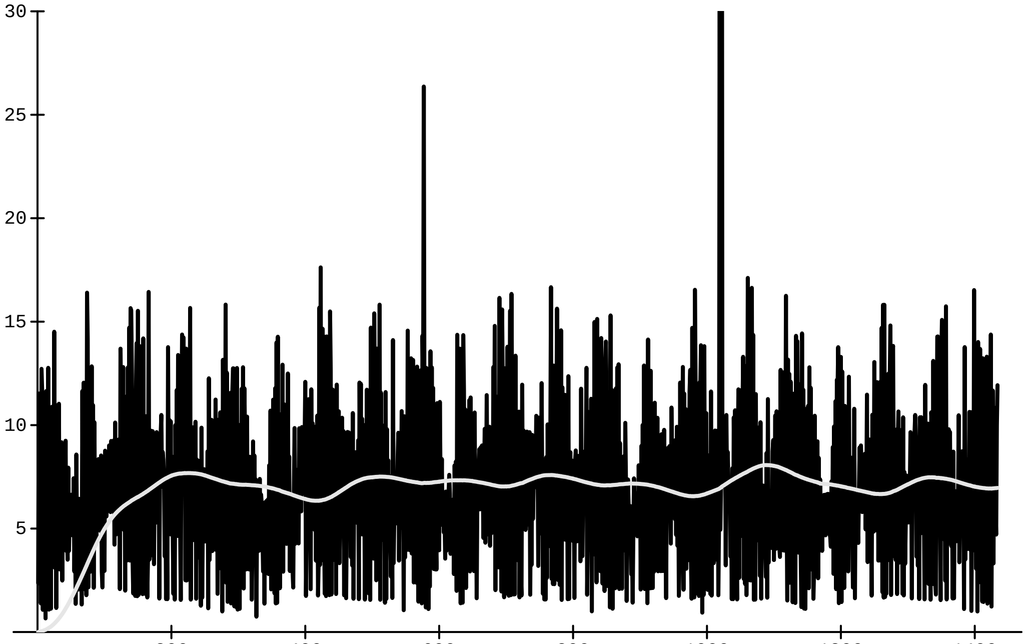


Figure 12: Input signal s_1 (black) and filtered signal s_0 (gray)

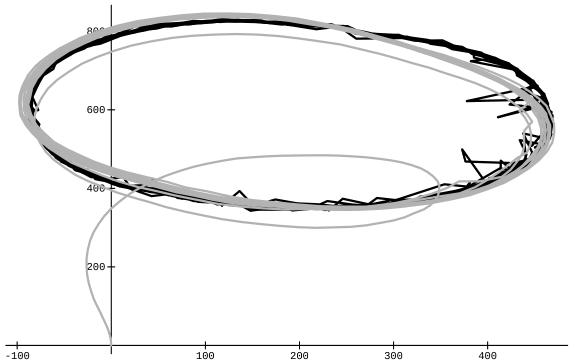


Figure 13: Input trajectory (black) and filtered trajectory (gray)

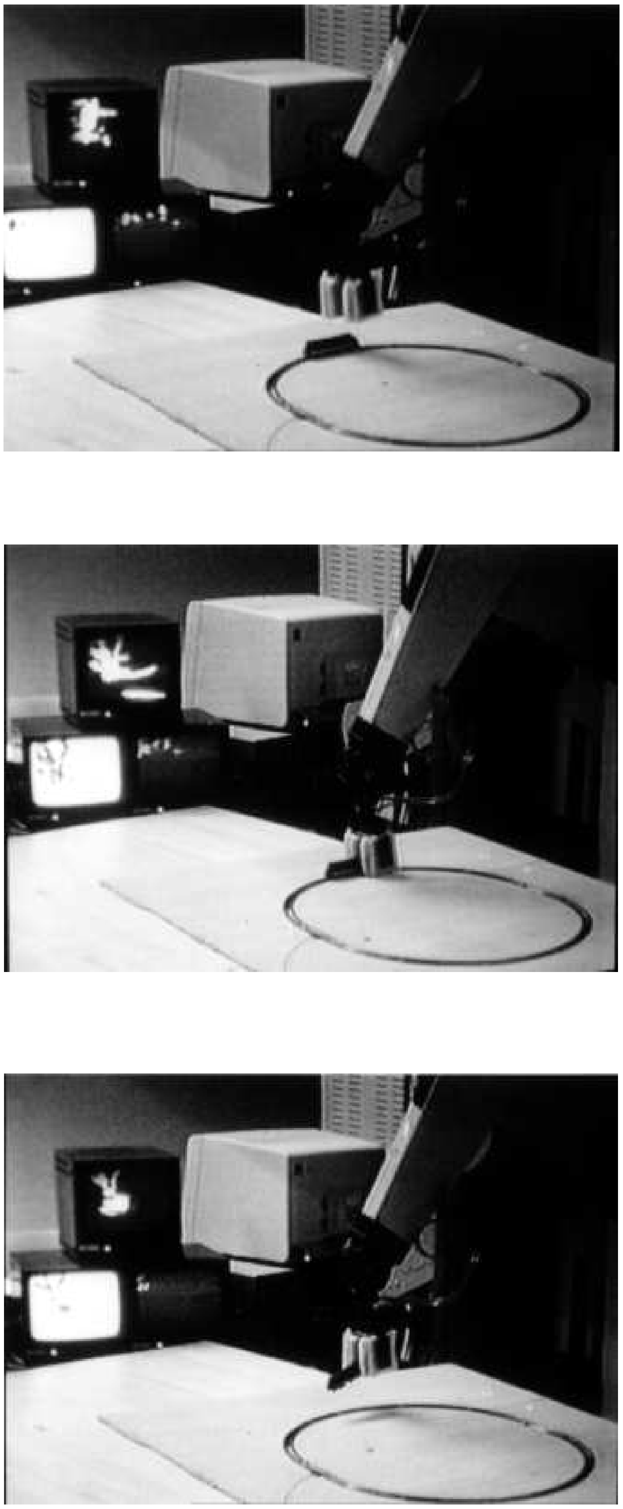


Figure 14: Intercepting, grasping and picking up the object

inaccuracy of visual sensors by applying parameterized filters that smooth and can predict ahead the moving object's position. Once this tracking is achieved, a grasping strategy is applied that performs an analog of human arm movement schemas.

The system is robust in a number of ways. The vision system does not require special lighting, object structure or reflectance properties to compute motion since it is based upon calculating optic-flow fields. The control system is able to cope with the inherent visual sensor noise and triangulation error by using a probabilistic noise model and local parameterization that can be used to build a non-linear filter to extract accurate control parameters. The arm control system is able to cope with the inherent bandwidth mismatches between the vision sampling rate and the servo-update rate by using a fixed gain predictive filter that allows arm control to function in the occasional absence of a video control signal. Finally, the system is robust enough to repeatedly pick up a moving object and stably grasp it.

We are currently extending this system to other hand-eye coordination tasks. An extension we are pursuing is to implement other grasping strategies. One strategy is to visually monitor the interception of the hand and object and use this visual information to update the **Drive** transform at video update rates. This approach is computationally more demanding, requiring multiple moving object tracking capability. The initial vision tracking described above is capable of single object tracking only. If we attempt to visually servo the moving robotic arm with the moving object, we have introduced multiple moving objects into the scene.

We have identified 2 possible approaches to tracking these multiple objects visually. The first is to use the PIPE's region of interest operator that can effectively "window" the visual field and compute different motion energies in each window concurrently. Each region can be assigned to a different stage of the PIPE and compute its result independently. This approach assumes that the moving objects can be segmented. This is possible since the motion of the hand in 3-D is known - we have commanded it ourselves. Therefore, since we know the camera parameters and 3-D position of the hand, it will be possible to find the relevant image-space coordinates that correspond to the 3-D position of the hand. Once these are known, we can form a window centered on this position in the PIPE, and concurrently compute motion energy of the moving object and the moving hand in each camera. Each of these motion centroids can then be triangulated to find the effective positions of both the hand and object and compute the new **Drive** transform. Both computations must, however, compete for the hardware histogramming capability needed for centroid computation, and this will effectively reduce the bandwidth of position updating by a factor of 2.

Another approach is to use a coarse-fine hierarchical control system that uses a multi-sensor approach. As we approach the object for grasping, we can shift the visual attention from the static cameras used in 3-D triangulation to a single camera mounted on the wrist of the robotic hand. Once we have determined that the moving object is in the field of view of this camera, we can use its estimates of motion via optic-flow to keep the object to grasped in the center of the wrist camera's field of view. This control information will be used to compute the **Drive** transform to correctly move the hand to intercept the object. We have implemented such a tracking system with a different robotic system [3] and can adapt this method to this particular task.

References

[1] M. Abramowitz, editor. *Handbook of Mathematical Functions.* National Bureau of Standards, 1964.

[2] E. H. Adelson and J. R. Bergen. Spatio-temporal energy models for the perception of motion. *Journal of the Optical Society of America*, 2(2):284–299, 1985.

[3] P. Allen. Real-time motion tracking using spatio-temporal filters. In *Proceedings of DARPA Image Understanding Workshop*, Palo Alto, May 1989.

[4] P. K. Allen, B. Yoshimi, and A. Timcenko. Real-time visual servoing. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1991.

[5] P. Anandan. Measuring visual motion from image sequences. Technical Report COINS TR-87-21, COINS Dept., University of Massachusetts-Amherst, 1987.

[6] N. A. Andersen, O. Ravn, and A. T. Sorensen. Using vision in real-time control systems. In *American Control Conference*, 1991.

[7] M. Arbib, T. Iberall, and D. Lyons. Coordinated control programs for movements of the hand. Technical Report COINS TR 83-25, Dept. of CS University of Massachusetts, August 1983.

[8] Aspex. *PIPE User's Manual.*

[9] C. Brown. Gaze controls with interaction delays. *Proc. DARPA Image Understanding Workshop*, pages 200–218, May 23-26 1989.

[10] P. J. Burt, J. R. Bergen, R. Hingorani, R. Kolczynski, W. A. Lee, A. Leung, J. Lubin, and H. Shvayster. Object tracking with a moving camera. In *IEEE Workshop on Visual Motion*, pages 2–12, Irvine, CA, March 20–22 1988.

[11] P. J. Burt, C. Yen, and X. Xu. Multi-resolution flow-through motion analysis. In *Proceedings of the IEEE CVPR Conference*, pages 246–252, 1983.

[12] B. F. Buxton and H. Buxton. Computation of optic flow from the motion of edge features in image sequences. *Image and Vision Computing*, 2, 1984.

[13] J. J. Clark and N. J. Ferrier. Control of visual attention in mobile robots. *IEEE Conference on Robotics and Automation*, pages 826–831, May 15-19, 1989.

[14] P. Corke, R. 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, 1989.

[15] P. J. B. *et al.* Object tracking with a moving camera. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1989.

[16] I. D. Faux and M. J. Pratt. *Computational Geometry for Design and Manufacture.* John Wiley & Sons, 1980.

[17] J. T. Feddema and C. S. G. Lee. Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera. *IEEE Transaction on Systems, Man and Cybernetics*, 20(5), 1990.

[18] R. Goldenberg, W. C. Lau, A. She, and A. Waxman. Progress on the prototype pipe. In *IEEE Conference on Robotics and Automation*, Raleigh, N. C., March 31-April 3 1987.

[19] H. H. H. Cruse, J. Dean and R. Schmidt. Utilization of sensory information for motor control. In H. Heuer and A. F. Sanders, editors, *Perspectives on Perception and Action*, pages 43–79. Lawrence Erlbaum, 1987.

[20] V. Hayward and R. Paul. Robot manipulator control under unix. In *Proc. of the 13th ISIR*, pages 20:32–20:44, Chicago, April 17-21 1983.

[21] D. Heeger. A model for extraction of image flow. In *First International Conference on Computer Vision*, London, 1987.

[22] B. K. P. Horn. *Robot Vision.* M.I.T. Press, 1986.

[23] B. K. P. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1983.

[24] N. Houshangi. Control of a robotic manipulator to grasp a moving target using vision. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1990.

[25] L. B. Jackson. *Digital Filters and Signal Processing.* Kluwer Academic Publishers, 1986.

[26] E. W. Kent, M. O. Shneier, and R. Lumia. Pipe: Pipelined image processing engine. *Journal of Parallel and Distributed Computing*, (2):50–78, 1985.

[27] A. J. Koivo and N. Houshangi. Real-time vision feedback for servoing robotic manipulator with self-tuning controller. *IEEE Transaction on Systems, Man and Cybernetics*, 21(1), 1991.

[28] D. Lee, D. Young, P. Reddish, S. Lough, and T. Clayton. Visual timing in hitting an accelerating ball. *Quarterly Journal of Experimental Psychology*, 35A:333–346, 1983.

[29] S. Lee and Y. Kay. An accurate estimation of 3d position and orientation of a moving object for robot stereo vision: Kalman filter approach. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1990.

[30] S. W. Lee and K. Wohn. Tracking moving objects by a mobile camera. Technical Report MS-CIS-88-97, University of Pennsylvania, Department of Computer and Information Science, Philadelphia, November 1988.

[31] R. C. Luo, R. E. M. Jr., and D. E. Wessell. An adaptive robotic tracking system using optical flow. In *IEEE Conference on Robotics and Automation*, pages 568–573, Philadelphia, 1988.

[32] W. T. Miller. Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on Systems, Man and Cybernetics*, 19(4):825–831, July/Aug 1989.

[33] H. H. Nagel. On the estimation of dense displacement vector fields from image sequences. In *Workshop on motion: Representation and Perception*, pages 59–65, Toronto, 1983.

[34] N. Papanikolopoulos, T. Kanade, and P. Khosla. Vision and control techniques for robotic visual tracking. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1991.

[35] N. Papanikolopoulos, P. K. Khosla, and T. Kanade. Adaptive robotic visual tracking. In *American Control Conference*, 1991.

[36] R. Paul. *Robot Manipulators.* MIT Press, Cambridge, MA, 1981.

[37] B. S. Y. Rao and H. F. Durrant-Whyte. A fully decentralized algorithm for multi-sensor Kalman filtering. Technical Report OUEL 1787/89, Dept. of Engineering Science, University of Oxford, 1989.

[38] R. B. Safadi. An adaptive algorithm for robotics and computer vision application. Technical Report MS-CIS-88-05, Department of Computer and Information Science, University of Pennsylvania, January 1988.

[39] R. B. Safadi. An adaptive tracking algorithm for robotics and computer vision application. Master's thesis, University of Pennsylvania, 1988.

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

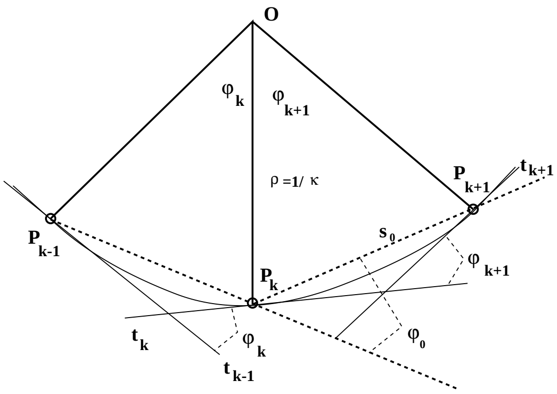


Figure 15: Trajectory curvature κ

[41] A. Singh. An estimation-theoretic framework for image-flow computation. In *Proc. International Conference on Computer Vision (ICCV-90)*, Kyoto, Japan, December 1990.

[42] G. Verghese, K. G. Lynch, and C. R. Dyer. Real-time motion tracking of three-dimensional objects. In *IEEE International Conference on Robotics and Automation*, Cincinnati, May 13–18 1990.

[43] C. von Hofsten. Catching. In H. Heuer and A. F. Sanders, editors, *Perspectives on Perception and Action*, pages 33–36. Lawrence Erlbaum Associates, 1987.

[44] C. von Hofsten. Early development of grasping an object in space-time. In M. A. Goodale, editor, *Vision and Action: The Control of Grasping*, pages 65–79. Ablex Publishing Company, 1990.

[45] L. E. Weiss, A. Sanderson, and C. P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, RA-3(5):404–417, October 1987.

[46] J. Wiklund and G. Granlund. Tracking of multiple moving objects. In V. Cappelini, editor, *Time Varying Image Processing and Moving Object Recognition*, pages 241–249, 1987.

[47] M. Xie. Dynamic vision: Does 3d scene perception necessarily need two cameras or just one? Technical report, Institut National de Recherche en Informatique et en Automatique, 1989.

A Trajectory Curvature

Here we prove that the trajectory curvature is given by the formula $\kappa = \lim_{T \to 0} \tan \varphi_0 / s_0$ where the following nomenclature is being adopted (see figure 15):

- κ is the trajectory curvature
- T is the sampling interval
- P_{k-1}, P_k, P_{k+1} are three consecutive points along the trajectory
- $s_0 = ||P_{k+1} - P_k||$ is the distance between points P_{k+1} and P_k
- O is the center of rotation
- $\varphi_k = \angle P_{k-1} O P_k$ = angle between the tangent lines t_{k-1} and t_k

- $\varphi_0 = \frac{\varphi_k + \varphi_{k+1}}{2}$ is the angle between lines $\overline{P_{k-1}P_k}$ and $\overline{P_k P_{k+1}}$
- ψ_k is the angle which tangent line t_k forms with x axis

Now we have:

$$\begin{aligned}
&\lim_{T\to 0} \frac{\tan\varphi_0}{s_0} = \lim_{T\to 0} \frac{\frac{\mathsf{d}}{\mathsf{d}T}\tan\varphi_0}{\frac{\mathsf{d}}{\mathsf{d}T}s} = \\
&= \frac{1}{v}\lim_{t\to 0}\frac{1}{\cos^2\varphi_0}\frac{\mathsf{d}}{\mathsf{d}T}\frac{\psi_{k+1}-\psi_{k-1}}{2} = \\
&= \frac{1}{\sqrt{1+y'^2}}\lim_{t\to 0}\frac{\mathsf{d}}{\mathsf{d}T}\frac{\arctan y'_{k+1} - \arctan y'_{k-1}}{2} = \\
&= \frac{y''}{(1+y'^2)^{3/2}} \qquad (19)
\end{aligned}$$

which is the formula for curvature [16].

B Velocity Expectation and Variance

In order to compute the mathematical expectation of the velocity we differentiate the following integral (integral 11.4.31 in [1]):

$$\int_0^\infty \mathsf{e}^{-at^2} I_0(bt)\mathsf{d}t = \frac{1}{2}\sqrt{\frac{\pi}{a}}\mathsf{e}^{b^2/8a} I_0(b^2/8a) \qquad (20)$$

with respect to a and by setting $a = 1/2\sigma^2, b = s_0/\sigma^2$ we get formula (12).

To prove formula (14) we use formula (11.4.29) from [1] (we set $\nu = 0$):

$$\int_0^\infty \mathsf{e}^{-at^2} t I_0(bt)\mathsf{d}t = \frac{\mathsf{e}^{b^2/4a}}{2a} \qquad (21)$$

By differentiation with respect to a and introducing the substitutions for a and b as in (20) we get the formula (14).

C Verification of formulas 13, 14, 15

. The formula 16 follows from 15 as follows: from $x = \frac{\sqrt{p^2-2z^2}}{z}$ we get by solving for z: $z = \frac{p}{\sqrt{x^2+2}}$. After substituting that value for z into 15, 16 follows immediately.

Since we have that $x = \frac{\sqrt{p^2-2z^2}}{z} = \frac{\sqrt{s_2^2-2\sigma^2}}{\sigma}$, after solving for σ we get $\sigma = \frac{s_2}{\sqrt{x_0^2+2}}$ which is equivalent to 18. From 14 follows that $s_0 = \sqrt{s_2^2 - 2\sigma^2}$. By substituting the value for σ, we get $s_0 = \sqrt{s_2^2 - 2\frac{s_2^2}{x_0^2+2}}$. It is easily shown that the last expression is equivalent to 17.