Method and Prototype for a Practical Ball Vision Device

Dave Gutz

22nd December 2006

18 Pickering Street, Danvers, MA 01923,
davegutz@alum.mit.edu

Abstract

With careful setup and the computational methods presented herein, a skilled consumer can use off-the-shelf frame-grabber equipment to accurately predict baseball strikes and speed. Using a 1:7 and 1:3 scale prototypes, a moving baseball's position is measured in real time at 30 frames per second within one-half ball width 3 standard deviations and it's impact with a zone 50 ball diameters further on predicted to within three ball width 3 standard deviations. Successful detection relies on optimum camera position, small field of view, good lighting, efficient spectral segmentation, efficient perceptual grouping, and a standard vector triangulation. The entire method is documented and the application software available for standard Linux. Results are displayed real-time with pitch statistics in pictorial format from an umpire's viewpoint. A record-playback mode is available for complete visual record. The system is run from a GTK+ graphical user interface. Up to 8 times improvement is foreseen with already available hardware in the form of IIDC-compliant camera.

1 Introduction

Ever-increasing computing power and new algorithms are progressing towards real-time filtering of image data on the desktop. Simple camera hardware is available. Programs exist, particularly for Linux, which provide easy access for a programmer to the image data stream. There is opportunity now for useful image application filters.

The potential payoff of vision detection systems is high. Presently, the Questec Corporation fields a proprietary device that segments image motion to track baseball pitches. At least two Major League Baseball clubs have contracted - for amounts in excessive of a star player's salary - to install and operate their proprietary system in ballparks. Umpires use it for training. Teams may use it for coaching. The Questec system tracks a ball for its entire trajectory.

This project's goal is to determine the method for calling a baseball pitch - strike or ball - from camera images. Users will want to know velocity as well. The specific tasks are to place a pair of cameras, sample their image streams, correct for camera distortion, segment the ball objects, triangulate ball position, develop a trajectory history, and determine where the ball passes through the strike zone. Users will want near real-time performance. For most of these tasks, the literature provides a starting point for efficient calculation. For the rest, new methods described herein are required.

The task of selecting cameras and placing them is not well documented in prior writing. Sections 8.4, 9, and 10.2 present solutions.

The task of sampling the image stream is a well understood computer system task. Linux is particularly easy to program the application using asynchronous communication and concurrent methods [4][6]. Section 10 develops a prototype application from scratch.

The task of correcting for image distortion is a common need for inexpensive cameras. Several investigators have dealt with it and provide solutions [7][8][9]. Section 8 improves upon one of the published meth-

ods.

The task of identifying the ball objects is most difficult and has a strong effect on accuracy and throughput. Within this task there are two main subtasks: "segmenting" to identify moving objects and "grouping" to identify the balls.

Shi and Malik [5] provide a broad strategy overview of segmentation. In particular, they suggest approaching the detail tasks of segmentation by considering the high level goal. The goal for this work, to identify a moving object, simplifies detail tasks. Baseballs are white, round, and move rapidly. Therefore, the pixels containing pertinent information about the ball will be changing rapidly from a dark background state to a light color from one frame to the next. A simple pre-filtering detects motion, reduces noise, and reduces significantly the number of pixels to be handled; see Section 2.1.

The next identification steps involve organizing the changed pixels into candidate object clusters. This work calculates the affinity of all the pixels to each other using a Gaussian weighting function [5]. Then the work applies a normalized spectral cut technique popularized by Pothen et al. [3] to segment the image. Suppose the matrix of edge weights describing the weight of each pixel with respect to others is given by W. Also suppose that the total weight of all edges of each pixel are given by diagonal matrix D with $D_{i,j} = \Sigma_j W_{i,j}$. An Eigenvalue problem arises whose solution is the second smallest Eigenvalue λ_2, called the Fiedler value, and whose Eigenvector y_2 defines the minimum cut. The implication of an Eigenvalue problem is one of frequencies, hence the term "spectral" segmentation. The general Eigenvalue system is

$$(D - W)y = \lambda D y. \tag{1}$$

This can be transformed into

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}z = \lambda z \tag{2}$$

where $z = D^{-\frac{1}{2}}y$, a transformed candidate cut. The matrix $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$is symmetric semi-positive definite, since $(D - W)$ is known to be semi-positive definite. The lowest Eigenvalue has value $e_1 = 0$ and associated Eigenvector $z_1 = D^{\frac{1}{2}}\tilde{1}$. After solving for the lowest, it is possible to reduce the order and solve for the second smallest. Then the resultant cut is transformed back to units of y.

Many methods are available to solve for the second Eigenvector. Demmel [2] reports that Rayleigh iteration provides quick, sure cubic convergence. Baker [1] provides a concise Rayleigh iteration algorithm. Since the first and lowest Eigenvalue and vector are known a-priori, then the matrix A reduces as follows to eliminate the lowest mode:

$$A_{i,j} = A_{i,j} - z_1 * A_{N,j},\ i = 1, N-1,\ and\ j = 1, N-1, \tag{3}$$

where i, j are matrix element coordinates. The next Eigenvalue and Eigenvector to be solved for will be the second Eigenvector. This reduced order matrix is not symmetric and is therefore not amenable to symmetric solution techniques. The initial guess of the second Eigenvector, z_2, usually a simplified zero vector with one arbitrary element set to 1, may be converted to the coordinates of the reduced order system y_2 by

$$y_{2_i} = z_{1_i} - y_{1_i} * z_{1_{N-1}},\ i = 1, N-1. \tag{4}$$

With a few Rayleigh iterations of the reduced order system the second Eigensolution solves efficiently. The solution progresses with cubic convergence [2]. The result is a spectrally segmented edge graph. The changed pixels are now in clusters.

The spectral segmentation task is computationally intensive and, fortunately, a simple approach is possible. Several investigators [2][5][10] discuss practical multi-level recursive algorithms that speed up the segmentation. Demmel summarizes many of the available methods, including Lanczos sparse matrix operations applied to entire images and Rayleigh iteration [2]. Demmel suggests multi-level recursion as a way to accelerate any method. In the situation of a moving baseball, the knowledge of the end goal allows the number of candidate pixels to be small so multi-level approach was unnecessary for this work.

The second task of identifying the balls is perceptual "grouping" the segments. There are various published methods to select clusters of pixels from the segmentation result to identify candidate objects.

Section 2.5 describes a unique one-pass Eigenvector method.

As a final step in determining balls, a high level algorithm recognizes the signature shape of a moving white ball. The task of correcting for camera distortion is best done around this point. By applying corrections to only the pixels that form the ball, significant computation savings are possible.

There is a challenge to determine synchronous images from asynchronous frame-grabbers. Section 4 presents a simple time interpolation that brings one frame's images in line with the other for triangulation.

The project's final tasks of triangulating ball position, Section 5, and strike zone prediction, Section 6, use basic linear math.

A prototype validates the methodology and uncovers potential problems. Optimum camera placement and lens selection are issues and the prototype work presents one solution. Camera internal and external calibration are covered. Implementing the algorithm uncovers other issues, especially timing. The prototype work demonstrates accuracy and gives a sense of how practical the full size system may be.

2 Method of One-pass Segmentation

The project emphasis is on accuracy and efficiency. This is most important for the segmentation task. Segmentation begins after pre-filtering reduces the number of pixels and identifies the pixels changed by scene motion. Following is a method that segments clusters from the small number of changed pixels in just one pass.

2.1 Pixelmap Pre-Filter

First, the method applies a pre-filter consisting of image difference with a threshold. This has precedent in the methods of dynamic analysis. The filter begins with the standard backward difference

$$\Delta I_{i,j}(t) = I_{i,j}(t) - I_{i,j}(t-T) \tag{5}$$

where I is black-and-white pixel intensity usually between 0 and 255 inclusive; i, j are the pixel coordinates with origin in image top-left; and t denotes time with T the frame update time. The image rate for selected light colored pixels is

$$R_{i,j}(t) = \begin{cases} \Delta I_{i,j}(t), \, abs(\Delta I_{i,j}(t)) > threshold \\ 0, \, otherwise \, not \, selected. \end{cases} \tag{6}$$

The user determines *threshold* to just eliminate normal device noise from the image, typically 10% of full scale.

2.2 Masks

Masks are useful for reducing image clutter which in turn improves throughput. After placing the cameras for optimal resolution there will be some portions of the image that contain clutter such as players' feet, dirt pathways, etc. Also, the edge of the image sometimes has random noise. It is appropriate and beneficial to use a trapezoidal shaped mask that screens only the pixels along the flight path visible by each camera. Throughput improves since the application performs less filtering.

2.3 Edges

Next, the method calculates total edge weight between all the non-zero rate pixels. The method uses light hued pixels only, an approach that reduces pixel number in half and image handling by four. Out of the $n*m$ total image pixels there are N_e of light-hued pixels with non-zero rate. The total edge weight is the sum of the distance Gaussian and the brightness Gaussian given by

$$E_{k,l}(t) = e^{-\left(\frac{\sqrt{(x_k - x_l)^2 + (y_k - y_l)^2}}{a}\right)^2} * e^{-\left(\frac{(I_k - I_l)}{b}\right)^2} \tag{7}$$

where now k and l are indices $k, l < N_e$ and a and b are weighting factors that the user determines from real-world tuning. The method uses pixel value instead of rate in the edge weight of Equation **??**. This

gives better identification of a ball since a ball intensity may have uniform value while the background may be less uniform. Using pixel value like this also reduces false effects of shadows. This is further application of Shi and Malik's principle of high level goal seeking applied to low level technique.

2.4 Segmentation

>From previous work referenced in the Introduction of Section 1, we know the first Eigenvalue and associated Eigenvector and can reduce the system for easier solution of the second Eigenvector. Previous work shows how to apply Rayleigh iteration to solve the second Eigenvector.

2.5 Eigenvector Clustering

Now with the second Eigenvector, the method groups it in one pass. Consider that pixels of the same group will have a strong Gaussian weight and will have similar Eigenvector value. The strategy, therefore, is to sort the Eigenvector according to magnitude then step through looking for magnitude to change from previous value by a predetermined threshold. When the value changes the method starts constructing a new group of pixels.

How does the method handle the case of a single object? Since often one ball is visible, then just one cluster may exist and would not properly segregate. To handle this case the method arbitrarily adds two edges to every segmented image, making $N_e + 2$ edges total. The edges are made arbitrarily using two closely associated points in a lower corner of the image and assigning zero affinity between each corner pair and any other pixel. The two extra clusters that result ensure a strong segregation of the Eigenvector solution. There are always at least two clusters. When finished with lumping pixels, the method simply deletes clusters of two or fewer pixels. This approach, combined with a good selection of threshold, works satisfactorily for several objects as well as robustly isolating a single object.

The method completes segmentation by applying distortion corrections to the pixel locations i and j. By correcting the grouped pixels after the rate and threshold calculations, fewer operations are necessary. The corrected locations are now floating point values denoted x and y. Section 8 details device internal calibration method and demonstrates improvement over previous methods.

3 Higher Level Grouping

The method uses four steps for one-pass grouping from the clusters as follows:

1. Connect each pixel in a cluster to all other pixels with a line segment. See the light, dotted lines in Figure 1.

2. Determine the perimeter using the outermost segments of the lines connecting the pixels. If one of the segmented pixels appears inside of the outermost line segment then weight the perimeter toward that pixel by half. The solid lines and dots of Figure 1 illustrate the approach and show the resultant pseudo pixels that weight the perimeter toward actual pixels.

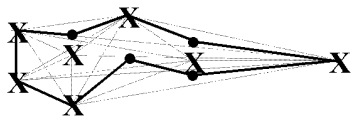
3. Determine properties of the cluster. Calculate the centroid of the cluster by integrating the moments of slices taken through the cluster. Calculate value as the average gray-scale pixel value. Calculate density by dividing the area by the number of segmented pixels. Determine shape by the aspect ratio of the smallest rectangle that contains the object. Reject clusters with low density of undesirable shape. Use the centroid for the object position.

4. Join clusters that are close in centroid position.

5. Join clusters that are close in perimeter.

6. Reject clusters that touch image edges or mask edges. It is assumed that part of the ball smear is obscured.

The shape shown in Figure 1 illustrates that the ball motion smears the captured image. Advance knowledge of shape simplifies the grouping strategy. For detecting a moving baseball, desirable objects have an oblong shape with high density and light color.



KEY:

X	- PIXELS
.....	- PIXEL CONNECTIONS
•	- INTERPOLATED PSEUDO-PIXELS
——	- PERIMETER

Figure 1: Method to determine object perimeter.

4 Time Sorting Objects

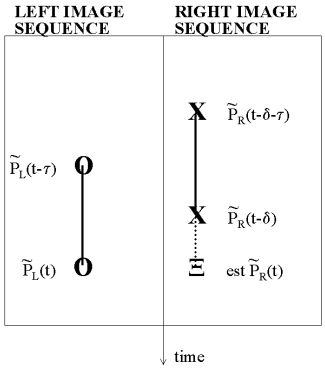
The method uses three steps for one-pass selection of balls:

1. Sort all objects in time order. Using the most recent ball, the selection logic looks for an object in the opposite camera. Figure 2 illustrates three cases that cover all possible combinations for interpolating the right-hand camera images.

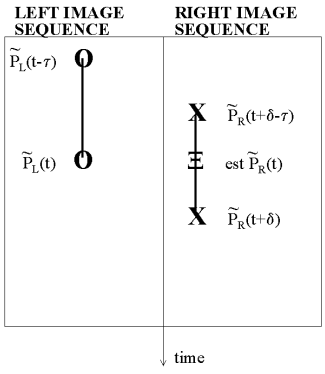
2. Interpolate to estimate ball position assuming constant velocity between frames. Figure 2 illustrates the interpolation result using the symbol Ξ. The parameter δ denotes small time skew between right and left image sequence streams and the parameter τ denotes nominal frame update time.

3. The method rejects balls with large time difference. This has the additional advantage of reducing effects of time slips in frame-grabbing.

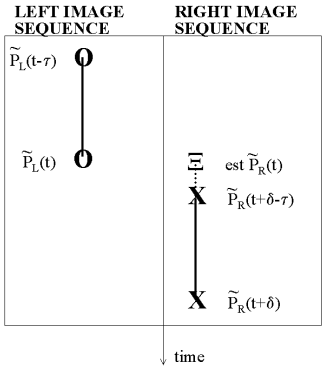
This method shows how knowing in advance the approximate ball speed and direction helps simplify the computations.



(a) One camera's objects trail the other's.



(b) One camera's objects straddle the other's.



(c) One camera's objects lead the other's.

Figure 2: Interpolating time: the possible combinations.

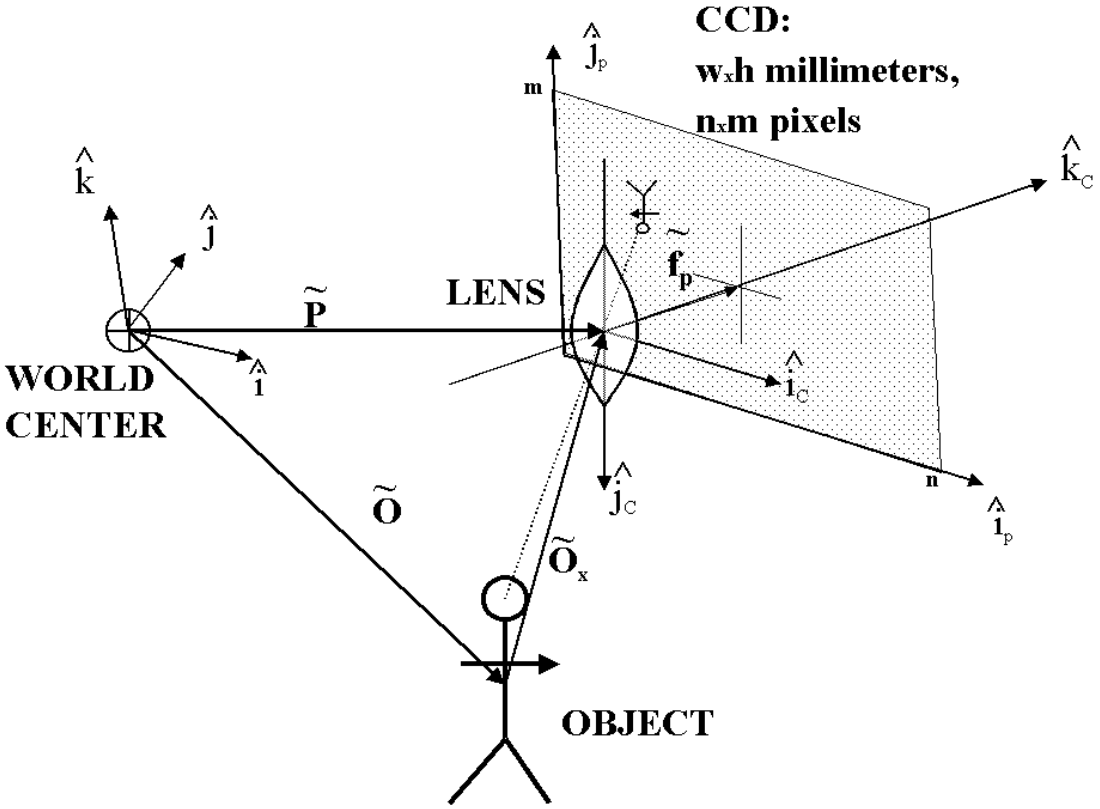


Figure 3: Definition of Coordinate Systems for Transformation. The lens is treated as a pinhole.

5 Method of Triangulating Ball Position

5.1 Object Triangulation

The method uses a pinhole camera model to transform pixel coordinate to a direction vector in world coordinates from the camera to the object. Figure 3 defines the three coordinate systems required. Let the two-dimensional pixel coordinate system be $(\hat{i}_p, \hat{j}_p)$. Its coordinate origin is at the top left corner of the image. The viewpoint of the user is upside-down in the back of the dotted pixel plan shown in Figure 3. Let the lens be a pinhole that directs all light through the origin of the second camera-centered coordinate system at $(\hat{i}_c, \hat{j}_c, \hat{k}_c)$. The camera coordinates are aligned with the image viewer's normal x-y coordinate system view in the center of the image with the z axis coming out of the image away from the object. The lens is distance f from the image sensor exactly in the image center. Units of $\tilde{f}_p$ are pixels.

The goal is to transform the pixel coordinates to the world coordinates $(\hat{i}, \hat{j}, \hat{k})$. The first transformation from image coordinates to camera-centered coordinates is

$$\tilde{v}_c = (i_p - n/2)\hat{i}_c + (m/2 - j_p)\hat{j}_c + f_p\hat{k}_c. \quad (8)$$

Note that the direction to the object from the camera-centered system is now a rotation applied to $\tilde{v}_c$. It is common to call the rotation the "direction cosine transformation," which we denote here by matrix $ACPW$, so that

$$\tilde{v} = ACPW \bullet \tilde{v}_c \quad (9)$$

with direction

$$\hat{v} = \frac{\tilde{v}}{|\tilde{v}|}, \quad (10)$$

a 1 by 3 column direction vector.

The vectors to the object from each camera are $R\hat{v}_R$ and $L\hat{v}_L$, where L and R are magnitudes of left and right cameras respectively. Combined with the cameras' world positions, $\tilde{P}_R$ and $\tilde{P}_L$, the resulting vector equations define the object location in world coordinates. Figure 4 illustrates these definitions.

The triangulation is over-specified such that each of three planar projections produces two direction magnitudes thus six scalar results for a three dimensional object location. But the vectors $R\hat{v}_R$ and $L\hat{v}_L$ from each camera rarely intersect at a point therefore the extra information may be used to improve the accuracy and predict an error bounds. The goal is to solve the system

$$\left\{ \begin{array}{l} \tilde{P}_L + L\hat{v}_L = \tilde{O} \\ \tilde{P}_R + R\hat{v}_R = \tilde{O} \end{array} \right\}$$

which reduces to

$$[\hat{v}_R \ \ -\hat{v}_L] \left\{ \begin{array}{c} R \\ L \end{array} \right\} = \tilde{P}_L - \tilde{P}_R. \quad (11)$$

Equation **??** further reduces to three scalar equations with two scalar unknowns R and L. There are redundant solutions. The average of the redundant solutions determines an estimate of $\tilde{O}$ and the difference determines an estimate of the error bounds.

The solution is undefined whenever the matrix $[\hat{v}_R \ \ -\hat{v}_L]$ is singular, $det([\hat{v}_R \ \ -\hat{v}_L]) = 0$. This is a mathematical statement of the situation where the two direction vectors are parallel, pointed at each other. This is not really possible with a practical camera setup because for optimum field of view, discussed in Section 5.2, the cameras will have strongly oblique viewing angles and be invisible to each other.

5.2 Triangulation Accuracy

An analysis of accuracy yields an estimate of optimum camera position. The preceding method of triangulating with intersecting rays inherently produces a head-on accuracy of half the root-sum-square the variation of two images. If uncertainty in each image is equal then the total predicted uncertainty of both combined is $\sqrt{2}/2$ times the uncertainty of one image. The divisor two used is because the uncertainty of intersection and the over-specification of the vector solution allows an average to be made of the two solutions. The root-sum-square factor arises from the

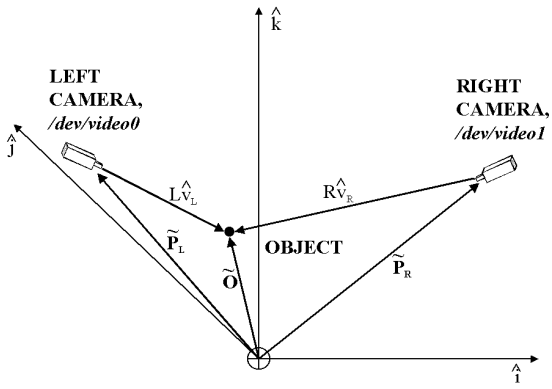


Figure 4: Coordinate Transformations

statistical sum of two independent image uncertainties.

The triangulation process gives up some accuracy to camera orientation because to get solid ray intersections in all directions the cameras need to be obliquely placed. The following analysis defines optimum camera angle ranges to minimize errors from image uncertainty.

As a practical consideration, recall that the cameras should not see each other and thereby create a potential head-on vector intersection and consequent zero determinant.

Figure 5 illustrates planar views of image variation effect on triangulation results. The dashed line represents the error of one image pixel projected into the world space and shows the projected world errors Δx, Δy, and Δz. The camera angles vary between a vertical angle of Φ and a horizontal swept back of Θ. It simplifies the analysis to assume that the cameras are symmetrically placed and the object is midway between the cameras. The resulting triangulation errors appear in Figures 6, 7, and 8. These resulted from using an *Octave* sensitivity simulation perturbing (i_p, j_p) by $(1, 0)$ in one camera. The units are the fractional pixels the triangulated image would shift for a one pixel shift in one of the camera images.

The resulting range of camera angles producing a

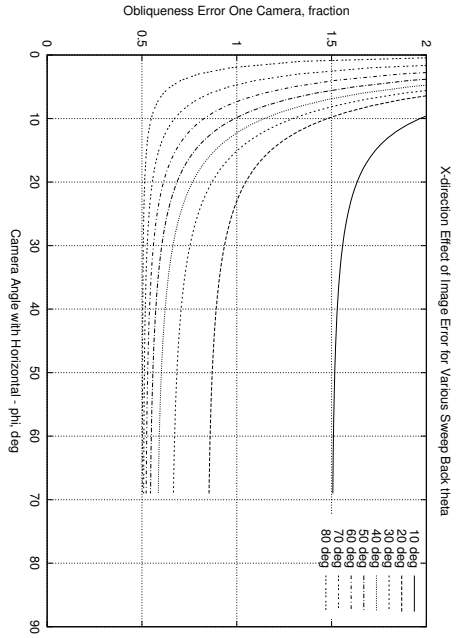


Figure 6: Accuracy in x direction, pixels/pixel for one camera.

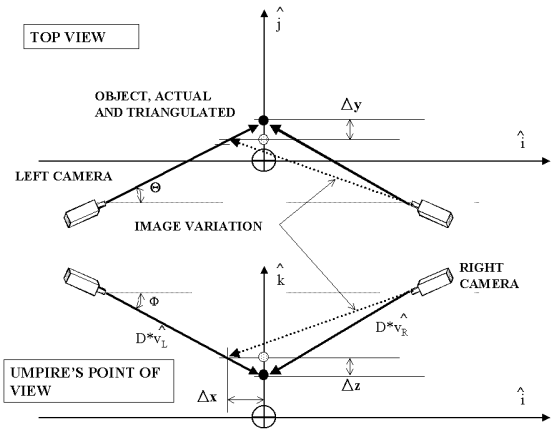


Figure 5: Error Estimation

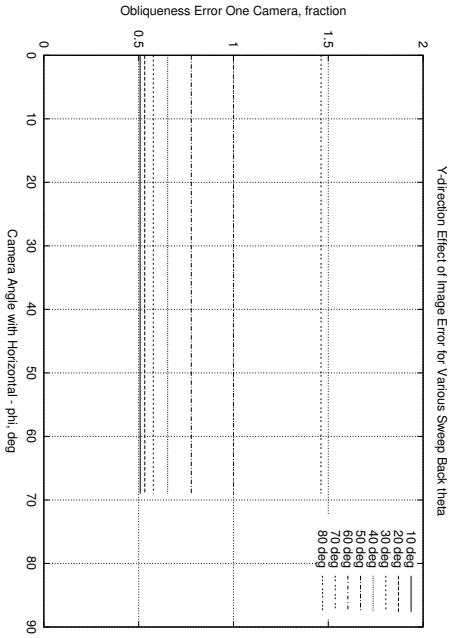


Figure 7: Accuracy in Y direction, pixels/pixel for one camera.

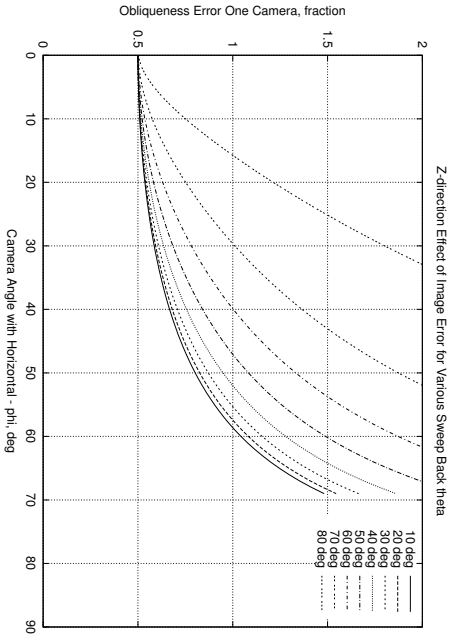


Figure 8: Accuracy in Z direction, pixels/pixel for one camera.

Downward, deg.	20	20	40	40	30
Forward, deg.	40	60	40	60	50
x	1.06	0.85	0.92	0.78	0.95
y	0.92	1.41	0.92	1.41	1.08
z	0.78	0.92	1.06	1.41	0.99

Table 1: Triangulation accuracy in fraction of pixels object uncertainty per pixel of image uncertainty, two cameras root-sum-squared.

simultaneously small percent error in all directions is 40 to 60 degrees for sweep-back, Θ, and 20 to 40 degrees for elevation, Φ. Some users may wish to emphasize a certain projection accuracy, for example x more than z, and would use a larger Φ and/or smaller Θ. Table 1 is a prediction of overall accuracy of triangulated object location. At 50 degrees forward pointing sweepback, $\Theta = 50$, and 30 degrees downward pointing angle with horizontal, $\Phi = 30$, the predicted uncertainty is uniformly 1 pixel in all orientations.

6 Method of Calculating Trajectory to Strike Zone

The method of calculating trajectory is simply to keep a history of detected balls and extrapolate to the strike zone.

There are practical considerations that improve accuracy and repeatability. For example, the method performs best predicting trajectory of balls viewed for three frames just in front of the target and outside the range of other motion. Thus, a small field of view is preferred. This is because the ball appears larger with more pixels and because there are no other objects to interfere. The drawback is loss of ability to observe the whole trajectory. Section 9 discusses this in more detail. Also, averaging two past values reduces variation by half. Horizontal flight curvature effects may be significant but were not studied. And for the z-direction, a gravity acceleration adjustment to triangulated velocity extrapolated to the target reduces z variation significantly.

The final method to predict a strike is shown in Figure 9. The present object location $\tilde{P}(t)$ and velocity $\tilde{V}(t)$ are known from triangulation. Let S be an arbitrary scalar on the velocity required to produce an exact intersection of $S\tilde{V}(t)$ with the zone at estimated $\tilde{P}(t + T_Z)$, where T_Z is the time to zone. The vector equation is

$$\tilde{P}(t) + S\tilde{v}(t) = x_s\hat{i} + z_s\hat{k} \tag{12}$$

with solutions $S = -P_y/v_y$ and

$$\left\{ \begin{array}{c} x_s = P_x - P_y v_x/v_y \\ z_s = P_z - P_x v_z/v_y \end{array} \right\}. \tag{13}$$

7 External Calibration

Section 5 showed that the triangulation calculation requires a rotation matrix - direction cosines - that transform camera pixel coordinates to world coordinates. In this method determining the rotation matrix, orienting the cameras in space, is called external calibration to distinguish it from camera device internal calibration. It is required to know the cameras' world Cartesian coordinates a-priori, through a plumb and tape measurement. To determine the remaining three angles of rotation of each camera we need to know the direction vectors to three known still target locations provide directions for an inverse triangulation. Three balls are required for three directions. By using the same three balls for both cameras the method allows a triangulation check after zeroing in.

It turns out that moving balls are no different than still balls for the same light source so external calibration is complete after working with just the still images.

For external calibration we apply the assumption that direction vectors from both cameras intersect exactly. Each camera has three camera-coordinate direction measurements,

$$\hat{v}_c = \left[\begin{array}{ccc} \hat{v}_{c1} & \hat{v}_{c2} & \hat{v}_{c3} \end{array} \right], \tag{14}$$

one for each object. The objects are at

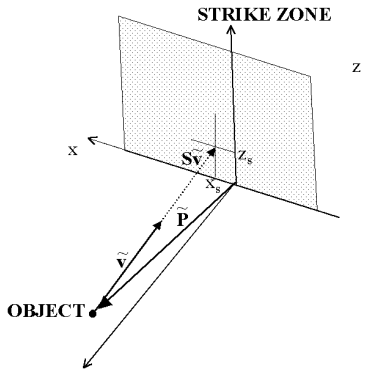


Figure 9: Strike Zone Calculation

$$\tilde{O} = \left[\begin{array}{ccc} \tilde{O}_1 & \tilde{O}_2 & \tilde{O}_3 \end{array} \right]. \quad (15)$$

One must be careful to normalize the direction as well as the object position column vectors individually to form the two three-by-three matrices

$$\hat{O} = \left[\begin{array}{ccc} \hat{O}_1 & \hat{O}_2 & \hat{O}_3 \end{array} \right]. \quad (16)$$

The rotation matrix is the solution to

$$ACPW \bullet \hat{v} = \hat{O}. \quad (17)$$

Recall the general matrix property $A \bullet \tilde{x} = (\tilde{x}^T \bullet A^T)^T$. To find $ACPW$ the method solves for each column of $ACPW^T$ then transposes to determine $ACPW$. The method then checks the result by calculating the error bounds and predicted positions for the three fixed objects.

The application program performs this portion of zeroing in by using as input the world locations of the three objects, the world locations of the two cameras, and the six pixel coordinates of the three objects for both cameras.

8 Camera Device Internal Calibration

Cameras introduce distortion through lens aberration and sensor imperfections. To measure velocity accurately, exact internal calibration of the camera device characteristics is important. Both the camera and frame-grabber introduce distortion mostly at image edges. The three main effects addressed by this work are lens focal length calibration, sensor aspect ratio calibration, and device radial distortion. A functional test of each camera is required because camera specifications are approximate for general system sizing studies such as placing cameras and selecting lenses. Also, each camera has a unique variation from nominal on the order of 5%. Interactive calibration corrections described below correlate the imperfectly measured pixel coordinates with known world direction.

8.1 Focal Length

Focal length determines the image magnification. Referring again to Figure 3, the $\tilde{f}_p$ vector is the focal length applied to triangulation. It is possible to estimate the actual focal length by taking several snapshots of a gauge placed at various distances from the camera [9]. Figure 10 shows that by similar triangles

$$\frac{f_p}{\Delta p_i} = \frac{D_i + \Delta d}{L} \tag{18}$$

where D_i is a position of the grid from an arbitrary datum on the camera, Δd is the unknown uncertainty in lens position from the camera datum that will fall out of regression, L is the width of a grid square in units of D, Δp_i is the measured pixel width of the grid square, and f_p is the lens focal distance from the camera sensor in pixels. Rearranging to isolate f as an intercept yields

$$D_i \left(\frac{\Delta p_i}{L} \right) = -\Delta d \left(\frac{\Delta p_i}{L} \right) + f_p. \tag{19}$$

A first order regression with $\frac{\Delta p_i}{L}$ as the independent parameter and measurements of $\frac{\Delta p_i D_i}{L}$ as the dependent parameter yields f_p as the intercept and $-\Delta d$ as the slope.

The prototype system cameras exhibited differences from specification values of 2.5% and 6.9%.

8.2 Sensor Aspect Ratio

Aspect ratio is the ratio of image width to height. The camera sensor introduces this error by having a CCD element that is not perfectly square. Also, the specification values may be general definitions for a product line and may not exactly match the camera used. An aspect ratio calibration removes this source of error.

Aspect ratio calibration falls out of the radial distortion calibration described in Section 8.3 that follows. The prototype system exhibited differences from specification values of 2.5%.

8.3 Radial Distortion

This is a two step process designed to accurately compensate for lens radial distortion, sometimes called "barrel distortion." It is the "fish-eye" effect of wide-angle lenses. Sensor aspect ratio falls out of calibration for radial distortion.

There is a widely used method published by Tsai in [7] and [8]. The Tsai regression approach determines effective image center, aspect ratio, and aberration coefficient. The method here is quite similar to the Tsai method with some important differences in correcting for image center.

Both the proposed and Tsai methods use the cubic distortion model

$$r = \rho + \alpha\rho^3 \tag{20}$$

where r is undistorted image radius from center, ρ is image radius from center, and α is a coefficient to be determined by calibration [9]. The Tsai method typically biases the measured pixels (i_p, j_p) to a virtual image center (i_m, j_m) that is not at the physical center. In other words, $i_m \neq \frac{n}{2} + 1$ and $j_m \neq \frac{m}{2} + 1$. In this work, however, it was discovered that a linear correction for distance from image center has a stronger effect. The best accuracy results from using $i_m = \frac{n}{2} + 1$ and $j_m = \frac{m}{2} + 1$ in

$$\frac{r}{\rho} = R_0 + \alpha(\frac{\rho}{n})^2 + B(\frac{i_p - i_m}{n}) + C(\frac{(j_m - j_p)(\frac{n}{m}\frac{1}{AR})}{n}) \tag{21}$$

where R_0, α, B, C are the four aberration coefficients needed and AR is the sensor aspect ratio of $\frac{n}{m}$ pixels that produces a square image as determined initially by the focal length measurement. The sensor aspect ratio is fine-tuned with the regression for the four aberration coefficients. The primary difference with the Tsai technique is eliminating the variable image center and using a linear image location scalar. The four coefficients combine with focal length and aspect ratio to make six total factors. Note that n is used to scale both i, j. This has the simplifying effect of lumping all sensor aspect ratio effects in the term $\frac{n}{m}\frac{1}{AR}$.

The steps required to determine the coefficients necessary to use Equation **??** are:

1. Take a still shot of a grid placed distance D from each camera. At least an eleven by eleven grid is recommended. It is useful to have a grid crossing near dead center in the image. Record (i_p, j_p) measured for each grid intersection.

2. Calculate $(\frac{\rho}{n})^2$, $\frac{i_p - i_m}{n}$, and $\frac{(j_m - j_p)(\frac{n}{m}\frac{1}{AR})}{n}$ for each grid intersection.

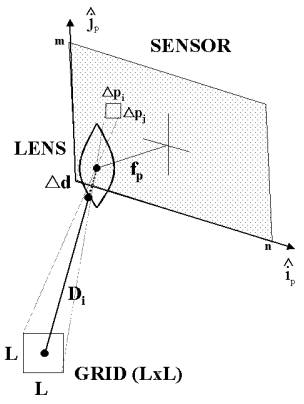


Figure 10: Focal Length Calibration Illustration

3. Calculate predicted (i, j) for each grid intersection from the focal sensitivity.

4. Make a scatter plot of $\frac{r}{n}$ as a function of $(\frac{\rho}{n})^2$. Fit a first order regression to the data with R-squared correlation displayed. Interactively adjust the sensor aspect ratio to maximize R-squared correlation and collapse the scatter into tighter patterns.

5. Perform a three-step regression of R_0, α, B, C to Equation **??**. The R_0 and α result from a regression on $\frac{r}{\rho}$ versus $(\frac{\rho}{n})^2$. Then B results from a regression on the residual from the first regression versus $\frac{i_p - i_m}{n}$ and C results from a regression on the residual from the second regression versus $\frac{(j_m - j_p)(\frac{n}{m}\frac{1}{AR})}{n}$.

If a regression tool is unavailable the user may set R_0 to 1.0, and B and C to 0.0 and perform a single regression for α and accept an accuracy loss. At least the first regression provides markedly improved internal calibration.

Alternatively the Tsai method is available in public domain. One may adapt Tsai results by applying the relationships $i_m = Cx$ and $j_m = Cy\frac{dpy}{dyx}sx$ where Cx, Cy, dpy, dpx, sx are standard Tsai program outputs and α and focal length have the same meaning in both methods.

Section 10.3 demonstrate calibration results for a prototype. Prototype accuracy (n=640) is ±3 pixels with the improved method compared to ±15 pixels for the Tsai method. Without either correction, accuracy is ±30 pixels, equivalent to 7.0 ball widths. With improved method, accuracy of 3 pixels is approximately 0.7 ball widths.

Aberration would manifest itself especially at the edges of the image where the distance from image center is large and the radial distortion large. There are some offsetting effects that reduce the effect on triangulation accuracy, to be discussed later. The aberration errors always affect velocity measurements, however.

8.4 Practical Considerations

As a practical matter it is helpful to orient the cameras for best accuracy. In other words, let the objects pass through the least distorted portion of the image of both cameras. This means the path of the objects is through the image center and follows a radial line from one image corner to the opposite corner. This happens naturally using the optimum compound angle setup. It is helpful also to point the camera so the object is in or just past the center of the frame

for the strike zone estimation. This has the added benefit of reducing shadows.

In summary, the process of internally calibrating the device is primarily for a new or unknown camera and/or frame-grabber configuration. Use a standard image width and height. One would take still snapshots of a grid placed at various arbitrary distances from the camera over the expected range of operation. Needed next is a still snapshot of a large, screen-filling grid at least eleven by eleven with a grid crossing near image center. One measures the grid crossings and performs a multi-variable regression for the coefficients of Equation **??**.

9 Sizing a Vision System

"Sizing" determines the optimum lens focal length for field of view and determines image size setting for resolution. A focal length too large will produce a small field of view and too few balls for a trajectory calculation. At least 4.5 balls are needed to predict trajectory. This allows for smearing across the image and up to one ball asynchronicity between cameras. An image size too large will use too many machine cycles and make real time processing impossible. At least 3 pixels are needed to see the ball. To make the optimization, start with defining the maximum object velocity, V_{max}, the distance from the camera to the object, D, the frame update time, τ, the camera sensor width, ω, and the ball diameter, B. Also, for sizing the cameras should already be in the optimal orientation Φ angle above horizontal and Θ angle swept back.

In addition, the frame-grabber throughput must be sufficient to capture a complete image in each frame.

Also, selecting CCD size is a high level economic decision. One would iterate the methods described below to trade camera CCD size (larger has better resolution) with cost.

9.1 Field of View

The objective for field of view sizing is to determine the focal length, f, to provide just enough field of view, W, to see four and one-half balls, $N_b = 4.5$. A distance / time calculation produces the relationship $W \geq (N_b - 1)\tau V_{max} cos(\Theta)$. The focal length relates to field of view according to $W = \omega M$, and $f = \frac{D}{M+2+\frac{1}{M}} \approx \frac{D}{M}$, where M is the magnification factor. Making the substitution and rearranging different ways gives

$$f \leq \frac{D\omega}{(N_b - 1)\tau V_{max} cos(\theta)} \tag{22}$$

which may be expressed another way as

$$D \geq \frac{(N_b - 1)\tau V_{max} cos(\Theta) f}{\omega}. \tag{23}$$

9.2 Image Resolution

The objective for resolution sizing is to determine image width in pixels, n, to provide just enough pixels to see three pixels on the object, $\mu_b = 3$. Similar triangles give $\mu_b = \frac{B}{W} n$ and $W = \frac{\omega D}{f}$. Making the substitution and rearranging different ways gives

$$f \geq \frac{\mu_b D \omega}{Bn}, \tag{24}$$

$$D \leq \frac{nBf}{\mu_b \omega}, \tag{25}$$

which may be expressed another way as

$$n \geq \frac{\mu_b D \omega}{Bf}. \tag{26}$$

The parameters n, μ_b, N_b are unit-less.

9.3 Break-even Sizing

The field of view and resolution sizing are opposing requirements. The condition for equality in Equations **??** and **??** is

$$n = \frac{\mu_b (N_b - 1)\tau V_{max} cos(\Theta)}{B}. \tag{27}$$

The goals for sizing are to determine this n then f and D.

9.4 Sizing Summary Example

An example illustrates how to size the system. Consider a practical 1/3-inch CCD camera. Per typical specifications in Table 3 the sensor width is $\omega = 4.8$ mm. A 100-MPH fastball is $V_{max} \approx 150$ fps. Ball size is $B = 2.75$ inches. The frame-grabber rate is 30 frames per second for $\tau = 0.0333$ seconds. The calculation for break even image size with $\mu_b = 3$ and $N_b = 4.5$ is 147 pixels from Equation **??**. A standard image size of 160 pixels therefore provides a nice image thumbnail with good resolution so $n = 160$ pixels is the choice with resolution better than needed. Equations **??** and **??** predict Table 2 for a forward pointing angle $\theta = 50$ degrees. For example, the focal length for a 1/3-inch CCD camera placed at 100 feet from the desired triangulation point would be 40 mm.

Using a zoom lens is an impractical decision. It is attractive to be able to move cameras between sites that have different camera distances. However a set of camera distortion factors need to be generated to cover different ranges of focal length, because camera internal calibration changes with focal length. A zoom lens could be made to work with pre-generated set of factors.

An auto-iris is a useful feature to cancel the very strong disturbances in accuracy caused by light variation.

Table 2: Focal Length Sizing Example for 1/3-inch CCD, $\omega = 4.8$ mm, $V_{max} \approx 150$ fps, $n = 160$, $\theta = 50$, $B = 2.75$, $\tau = 0.0333$, $\mu_b = 3$ and $N_b = 4.5$.

D, feet	resolution f, mm	field of view f, mm
10	3.9	4.3
20	7.9	8.5
30	11.8	12.8
40	15.7	17.1
50	19.6	21.4
60	23.6	25.6
70	27.5	29.9
80	31.4	34.2
90	35.3	38.4
100	39.3	42.7
110	43.2	47.0
120	47.1	51.3
130	51.1	55.5
140	55.0	59.8
150	58.9	64.1

Table 3: CCD Typical Specification Sizes. Cost increases with size.

CCD Size Class	width w, mm	height h, mm
1/4 inch	3.6	2.7
1/3 inch	4.8	3.6
1/2 inch	6.4	4.8
2/3 inch	8.8	6.6
1 inch	12.8	9.6

10 Prototype Vision System

A scale model prototype is an effective demonstration of the method and principals presented so far. The sizing method Section 9 provides an estimate of proper setup and orientation. Developing a prototype also forces the application software to scale to different image sizes and different cameras. And accuracy performance can be checked. It is reasonable to expect the results to scale to a larger system.

10.1 Prototype Description

10.1.1 Prototype Hardware

Two off-the-shelf US Robotics 3Com Web Cam with Brooktree Bt848 video capture cards are the prototype vision devices. The application computer platform is standard Linux/i686 kernel 2.2.12-20 (Redhat 6.1) on a 350 MHz Pentium II with 100 MHz motherboard. At least 128 MB is necessary to achieve near real-time performance with 160x120 images in both cameras.

Later, a better set of cameras and lenses repeats the experiment with a larger prototype.

10.1.2 Application Software Modes

The application program, *ump*, has two operating modes: record/playback and real-time. The record mode runs off-line another application derived from the *xawtv streamer* program, an interface to the video4linux *bttv* utilities, to record a few seconds of images to a raw image file for each cameras. After recording, the *ump* program automatically begins playback. The user can also request the application to playback stored files. For two image streams, up to 320x240 size the streamer program does not slip using the Pentium II system with 256 MB memory. This mode is RAM intensive and produces a result approximately the same time after the event as the length of the event itself. Analysis of a five second sequence would appear five seconds after the completion of the pitch.

The real-time mode enters playback immediately and reads synchronization information from a file descriptor that is being filled by a forked *streamer* process. The *streamer* program performs the image differentiating and maps the results in shared interprocess communications (IPC) memory for the application to use [6]. This mode is CPU intensive and slips with a fully loaded 160x120 image size but not 120x90 image size. The full loading is equivalent to 1.5 balls visible continuously so there is no slippage with 160x120 image size in practice with the Pentium II platform.

10.1.3 Application Software Segmentation

In playback mode the *ump* program reads the two raw image files with associated time stamps and synchronizes their usage with a threaded producer/consumer scheme [6] [4]. The threads differentiate each image from previous image storage following Equation **??** with $threshold = 30$, except for real-time mode, where it is done more efficiently by the *streamery* application, and calculate graph edge weights using pixel spacing and pixel brightness difference in a Gaussian weighting scheme following Equation **??**with $a = 4$ pixels for $n = 160$ full scale and $b = 30$ counts for 256 full-scale intensity. A segmentation computation is next [5], that calculates the second Eigenvector minimized cut using Rayleigh iteration with floating point precision in 6-7 iterations. A unique sorting algorithm described in Section 2 selects the minimum cut directly from the Eigensolution for efficient one-pass performance. Before determining shape the producer threads throw out the dark clumps, determined best by average pixel value, not by average pixel rate, then correct the image for aberration. Aberration correction is done here after segmentation to minimize throughput. Lines connect all the pixels in a candidate clump as shown in Figure 1. Proceeding left to right the bottom perimeter is figured as the minimum line value. So if there are no pixels in the column an interpolation is made. However, if there is a pixel in the column the minimum line value is averaged with the actual pixel location. This is repeated in the opposite direction along the top until the clump is encircled. If there is just one pixel in a column the same pixel is used to average both top and bottom perimeter values. With potential objects identified, the segmentation threads perform some higher-level sorting. It is assumed that one ball is present and close clumps become one. Finally, the thread pass the candidate objects onto to a consumer thread which triangulates position. Multiple balls will pass through and become rejected later in the consumer thread. Balls that contact the mask edge are expendable.

10.1.4 Application Software High-level Sorting

The consumer thread sorts the candidate balls in time order. Using the most recent ball, the consumer looks for an object in the opposite camera. Figure 2 illustrates the selection. The three cases shown in Figure 2 cover all possible combinations for the right-hand camera. The consumer thread interpolates camera-detected objects to produce the latest predicted objects for both cameras simultaneously. The consumer thread rejects interpolation required for more than two frames in a camera. Then with the latest predicted images of the same time stamp it triangulates ball position. It keeps track of previous positions so next it predicts ball velocity and averages with previous ball velocity if available. Finally, the consumer predicts the strike zone impact according to Figure 9. The application assumes that just one ball is possible.

10.2 Prototype Sizing

It is impractical to make a new lens for the inexpensive 3Com web-cams so the objective of sizing the prototype system is to determine the optimum field of view dimensions. The specification for the 3Com camera shows a 1/4-inch CCD and $f = 3.8$ mm. Per typical specifications the sensor width is $\omega = 3.6$ mm. The system frame rate is 30 frames per second for $\tau = 0.0333$ seconds. To strive for a true scale model the prototype uses the same image size that the 1/3-inch camera example would use, $n = 160$ pixels. A readily available ball is a white steel marble, $B = 0.5$ in, so the prototype incorporates this. Using the same sizing criteria for the 1/4-inch CCD prototype as for a potential 1/3-inch CCD, $N_b = 4.5$, $\mu_b = 3$, $\Theta = 50\,deg$, the Equations ?? and ?? predict Table 4. According to Equation ??, the limit of D, distance of camera from objects, is limited to 27.9 inches to be able to see the ball with sufficient resolution of 3 pixels. As a result the cameras are about 27 inches from the shooter pipe shown in Figure 11. The cameras point to a place approximately 18 inches in front of each camera. Therefore from Table 4 the maximum object speed is 230 inches/second for sufficient field of view for four and one-half images.

Table 4: Sizing Example for 1/4 inch 3-COM CCD. 27.9 inches is the resolution limit for n=160 image width.

D, inches	V_{max}, inches/second
12	153
18	230
24	306
27.9	356

The ball diameter ratio and the magnification ratio predict that the prototype is a 1:5 scale model. As it turns out, the shooter pipe is capable of shooting marbles at a maximum speed of 170 inches/second and the pipe shoots slightly upwards so for the scale model six frames in each camera are visible with balls of sufficient resolution with each shot.

Table 5 shows the sizing comparison with a full-scale system. It is reasonable to expect the accuracies reported in Section 10.6 to apply to the full-scale system sized like the 1/3-inch full-scale result of Table 5. Since the velocities do not scale exactly, there is risk that smearing effects will be different in the full-scale than the 1:7 scale having slower ball speeds. Smearing effects may influence results.

An easy way to summarize the similitude is that field of view should be fifty ball diameters. Practically speaking, two camera each with a 5-50 mm zoom could achieve this easily with virtually any field configuration and camera placement.

10.3 Prototype Internal Calibration

Snapshots of cards with grids on them placed at various distances about 18 inches from the prototype cameras produce the data needed to correct the devices for focal length and aberration. Figure 12 shows one snapshot for camera */dev/video0* with a 1-inch gridded card placed 10 inches away. Another shot with a 1 inch aberration grid placed 18 inches away is shown in Figure 13. Here there are more grids shown than the recommended number of 11 for ade-

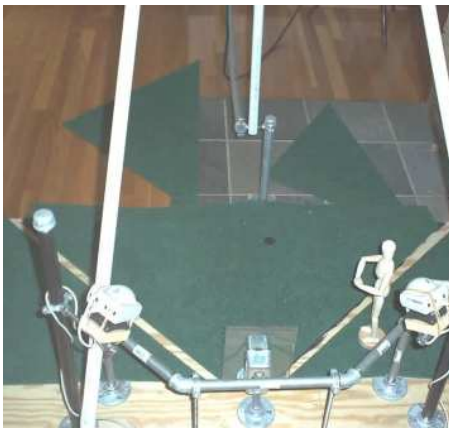


Figure 11: Umpire view of scale model prototype.

Table 5: Prototype Sizing Results and Similitude. Neglect ϕ .

Feature	Prototype	Full Size	Scale
Designation, nominal CCD size	1/4"	1/3"	–
CCD width, ω (mm)	3.6	4.8	1.33
Ball diameter, B (in)	0.5	2.75	5.5
Image resolution, n x m	160x120	160x120	–
Distance to balls, D (ft)	1.5	100	66.7
Focal length, f (mm)	3.8	40	10.5
Magnification, M	120	760	6.4
Forward point, θ (deg)	50	50	–
Frame rate, τ (sec)	0.0333	0.0333	–
Ball pixels, μ_b	3	3	–
Number of balls, N_b	4.5	4.5	–
Ball speed, V_{max} (ft/sec)	19 (13 MPH)	160 (110 MPH)	8.4
Sensitivity, (in/pixel)	0.11	0.90	8.2
Field of view, W (ft)	1.4	12	8.6

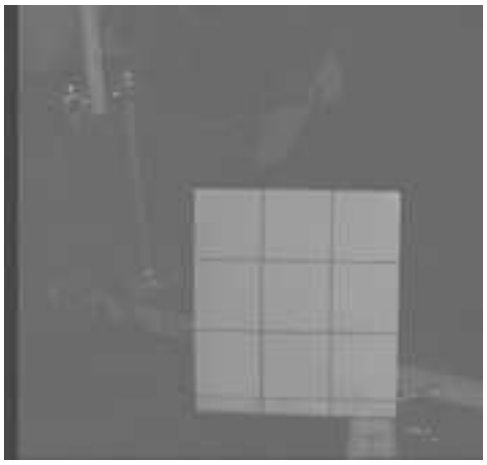


Figure 12: Focal length snapshot for 3Com camera at 10 inches.

quate internal calibration.

Results of the first attempt to internally calibrate using the method of Tsai appear in Figure 14. Prototype accuracy (n=640) is ±15 pixels for the Tsai method. The distribution contains some spreadsheet files to assist in this calibration. Without correction, accuracy is ±30 pixels. The effect of 30 pixels is approximately 7 ball widths if gone uncorrected. The aberration errors directly affect velocity measurements, and to a lesser extent, triangulation accuracy.

The large Tsai aberration error is corrected by the approach of Equation **??**. Figure 15 shows the result after applying Equation **??**. Accuracy is much improved to ±3 pixels.

A large number of pixels was used for the this initial calibration but may not be necessary.

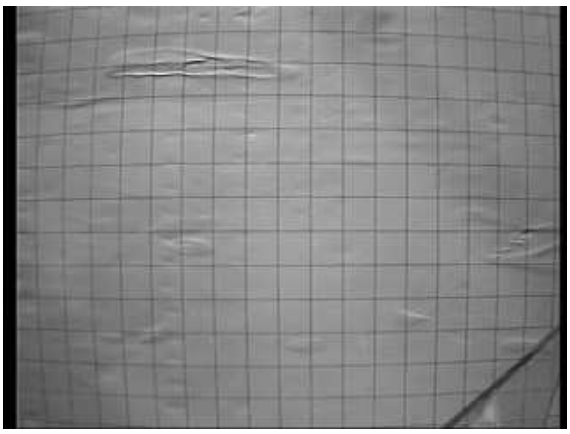


Figure 13: Aberration grid snapshot for 3Com camera at 18 inches. There are more lines than necessary.

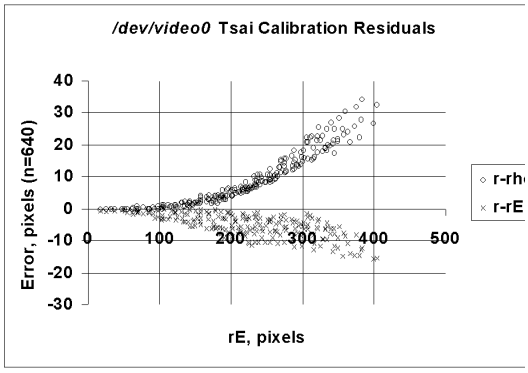
10.4 Prototype External Calibration

The documentation with the distribution of the application explains the steps required to shoot blank photos, weave them with still shots of stationary balls with known position, and convert the data into the proper form to transform pixel coordinates accurately to world object location. This closely follows the steps described in Section 7.

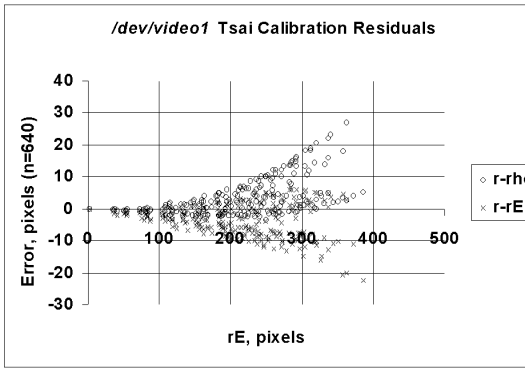
10.5 Prototype Test Method

The prototype used the application's record / replay mode of operation to save files for pictures and replay. To pitch a ball one rolls marbles down the conduit. The ball strikes the Plexiglas zone plate that has a piece of carbon paper and plane graph paper clipped to it. The program takes a three second shot and stops recording. Immediately, the program automatically replays and within two seconds calculates the projected strike zone at each frame where a moving ball is visible. A spreadsheet made an excellent way to compare the carbon paper measurements with the application program video measurements.

Figure 16 shows the sequence of shots from one run. Note that the ball is quite small, barely over 2 pixels width at the shooter tube that is about 28

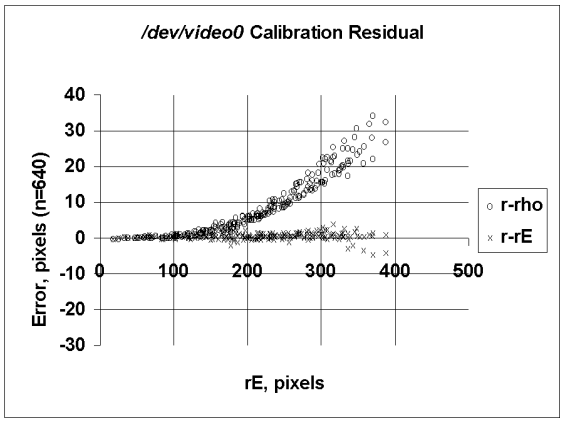


(a) Video device /dev/video1 results.

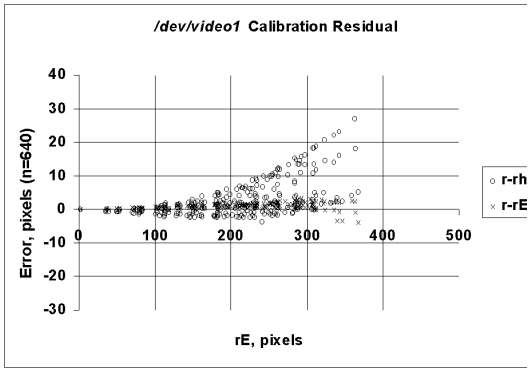


(b) Video device /dev/video1 results.

Figure 14: Device Tsai Internal Calibration Residual



(a) Video device /dev/video0 results.



(b) Video device /dev/video1 results.

Figure 15: Device Internal Calibration Residual

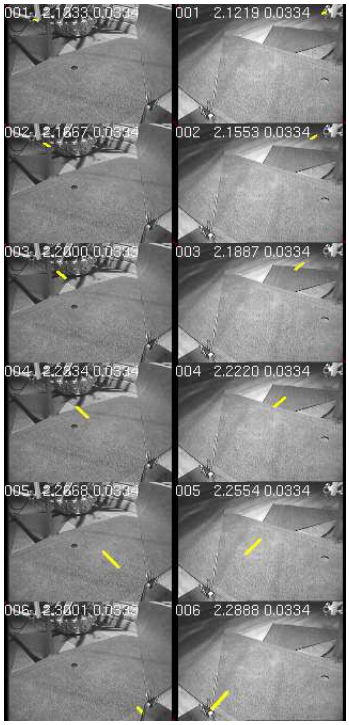


Figure 16: Event sequence record, 160x120, 30 fps. Ball is yellow.

inches from the cameras. At this moment the ball is moving slightly upward toward the camera. As it moves along it creates longer smear marks in the frame capture time.

The study varied image size, light intensity, calibration methods, frame rate, background color, computer RAM, image sub-sampling, number of averages, effect of gravity on vertical strike zone prediction, and real-time sequencing for real-time mode.

10.6 Prototype Test Results

Table 6 shows measured variation versus image size. The table is repeated three times in different units. There are various effects to be gleaned from these results.

It was unnecessary to center the strike measurements with a bias. The position result X is within measurement accuracy.

Internal device calibration has a smaller than expected effect on variation. This seems to be due to care taken to pass the object diagonally across the image and go through the center at the point of strike zone prediction to minimize aberration effects.

The horizontal direction X strike predictions vary within 1/2-ball width for the 160x120 configuration. This means that 99.9% of the time the call will be accurate to 1/2-ball. Observe that the variation in pixels is approximately constant with configuration, implying that a resolution effect drives the X direction variation. It should be reported, too, that averaging more samples of balls appeared to improve variation, but most of the benefit was realized by averaging two balls.

The vertical direction Z strike predictions vary significantly. This result appears to be a truly random effect, as evidenced by a normality test for various effects such as velocity, position at instant of calling strike, and number of averages used. The cause is not really known, though it is probably not related to image resolution considering the effect noticed for the X direction.

Note that Z direction performance is less important for an umpire because batter knee height varies. It may not be practical to call vertical strike distance without additional technology to provide batter knee height. These considerations discouraged further effort to understand Z variations.

Though internal calibration mildly affects variation it is unimportant for a good velocity measurement. This is believed to be due carefully centering the ball motion in the image where there is minimum aberration.

The results show that a larger image improves accuracy. This is despite "smearing" of the ball as the frame-grabber takes longer to sample the image. At the limits of throughput, approximately 320x240 for the prototype, the ball smears nearly the entire distance spanned between frames. It would seem that knowing the edges of the ball more accurately improves overall centroid prediction despite the longer smear.

Configuration	|X|	$3\sigma_x$	|Z|	$3\sigma_z$	Vy
	inches	inches	inches	inches	inches/sec
Baseline 160x120	0.06	0.23	0.59	0.54	-145
No Internal Calibration, 160x120	0.05	0.34	0.47	0.56	-149
Baseline 320x240	0.06	0.09	0.49	0.23	
2x Sub-sample 320x240	0.02	0.11	0.55	0.75	
	balls	balls	balls	balls	balls/sec
Baseline 160x120	0.12	0.46	1.18	1.08	-290
No Internal Calibration, 160x120	0.10	0.68	0.94	1.12	-298
Baseline 320x240	0.12	0.18	0.98	0.46	
2x Sub-sample 320x240	0.04	0.22	1.10	1.50	
	pixels	pixels	pixels	pixels	pixels/sec
Baseline 160x120	0.5	1.9	4.9	4.5	-1208
No Internal Calibration, 160x120	0.4	2.8	3.9	4.7	-1242
Baseline 320x240	1.0	1.5	8.2	3.8	
2x Sub-sample 320x240	0.3	1.8	9.2	12.5	

Table 6: Prototype variation results, 0.5 inch ball, sample size N=9. Sensitivity is 0.11 inches/pixel at 18 inches for 160x120 image and 0.06 inches/pixel at 18 inches for 320x240 image. Normal mode of operation is 2x sub-sample 160x120.

Results also show that sub-sampling does not degrade accuracy. This refers to a simple technique of using every other pixel value for segmentation. This is an important practical result, allowing significant increase in program throughput.

The prototype functions as a pure real-time system. A new computer is capable of better. The process is CPU intensive and runs fully loaded only up to $n = 120$ as shown in 17. The number of pixels changed in proportion to image width squared. Up to a point, $n = 120$ in Figure 17, the CPU can handle the increased pixels. But even then the real-time operation experiences dropouts as shown in Figure 18 due to normal asynchronous process scheduling. Additional experiments showed that the frame-grabber drops images. Further, it does not improve with buffering. Approximately three times per second the frame-grabber drops three frames. This is one-quarter the total number of frames. Fortunately, full loading does not occur in practice - it represents a stress test. The prototype works at $n = 160$.

The real-time system cannot record the entire image as it goes but will save and plot crude cartoon plots. The workaround to recording, should they be needed, is the record / playback mode. The mode queries the operator when a pitch is about to begin. When the operator replies, the prototype begins to grab. When it is finished the prototype immediately segments the images and calls the pitch. A five second capture will be available five seconds after the pitch. A large image sequence file is saved. The saved file contains the full base image for future reference. Then the prototype waits for the next request.

There were further results gleaned from the prototype. External calibration method was found to be unimportant. Three still balls placed flat over the playing field gave the same accuracy as three still balls placed on sticks through the flight path.

Light intensity was very important. Dim light resulted in many skipped images. Background consistency was important. There needs to be contrast between the ball and the background so light dirt or reflections causes skipped images if they adjoined a ball smear. The best situation was a dark green back-

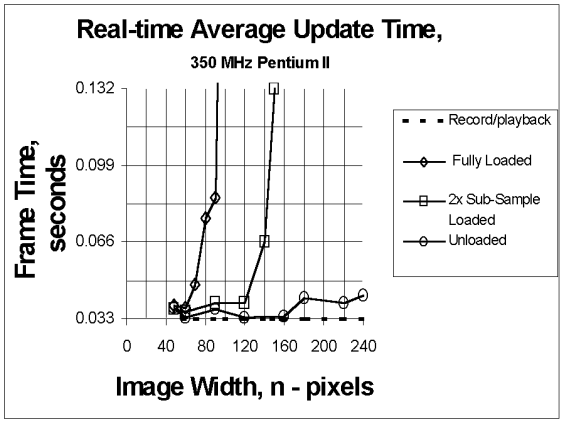
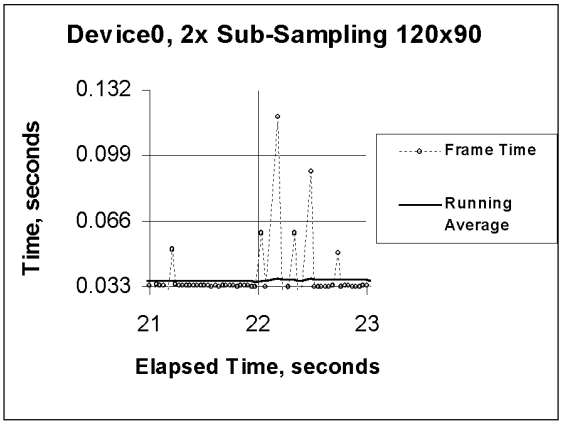


Figure 17: Prototype real time frame time average - maximally loaded frame in two devices simultaneously.



(a) /dev/video0

Figure 18: Real time performance 120x90 image with 2x sub-sampling - maximally loaded frame in two devices simultaneously.

ground with a five hundred Watt work floodlight five feet above the ball path. An auto-iris lens is recommended for a full size system.

After external calibration the prototype did not require any fine-tuning test shots. A moving ball is detected in a different place than a standing ball, one would expect. But mean accuracy was within measurement error.

The prototype program successfully self-adjusted segmentation constants and internal calibration constants for different image sizes.

A sample screen shot, Figure 19, demonstrates the application's interface. The recent result for record/playback is shown and the application is waiting for command perform the next capture. In real-time mode, the next detected ball will refresh the result.

11 Risk Mitigation

The prototype work reveals several sources of variation in the approximate order of importance shown in Table 7:

12 Larger Scale Model Prototype Test

The small webcam cameras were swapped out for two Sony SPT-M124 1/3 inch CCD black and white cameras running at 30 frames per second and having 380 lines horizontal resolution. They were run with 320x240 resolution. To mitigate light exposure risk, dc-driven auto iris CS-mount F1.4 lenses made by Tamron, 13VG550T, were mounted to the cameras. The lenses were 5-50mm zoom to allow proper field of view no matter the camera placement.

A 1-inch diameter ball was used and lenses were zoomed appropriately for approximately 50 ball diameters field of view (equivalent to 12 feet for standard baseball size).

After setup and calibration three simple tests were made to measure performance diagnostically. First, repeated still shots of a ball in a known spot were made. Second, repeated motion shots of a ball sliding

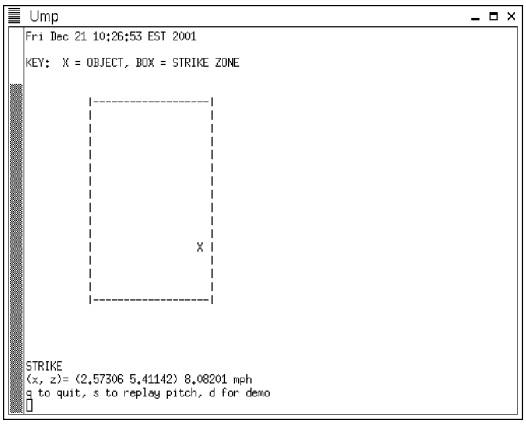


Figure 19: Screen shot of *ump* application interface.

Risk	Mitigation (*italics - tried on prototype*)
light intensity too low	daylight, *auxiliary lighting*, measure lumens, low-lux camera
light intensity variable	automatic iris
back-lit ball or dirty ball	*grass field, narrow focus in front of players*, color Gaussian
shadows	*oblique camera orientation, use pixel hue in edge weight*
extraneous motion	*narrow focus in front of players*, color Gaussian
throughput forces low image resolution	*image sub-sampling*, faster computer
real time synchronization dropouts	threaded frame-grabbing, real-time OS, *record / playback*
CCD tolerances	*internal calibration*

Table 7: Proven risk factors and mitigation - the prototype demonstrated mitigation in italics.

down a wire in a known position to a target of known position were made. Finally, repeated pitches were made and measured.

12.1 Internal Calibration

Using these more standard multi-element lenses it was found that internal calibration is more simply corrected than for the fisheye web cam lens discussed earlier. In fact, the need for improved methods goes away. Good results were achieved by taking a shot square on to a 1-inch pegboard, being very careful to align it absolutely square to the camera, using the exact same resolution and lens to be used for measuring pitches, and correcting simply for aspect ratio. The method of Tsai would work fine.

12.2 Still Shots

Twenty repeated shots of the 1-inch ball were made with the ball in an exact location. Each "shot" consisted of about 25 images bursts with a white ball then a black ball. The image bursts were shuffled digitally to produce pseudo motion and run through the detection software. Under these heavily controlled conditions the accuracy is +/-0.1 inches (0.1 ball diameters) as shown in 20. This is equivalent to +/-1 pixel of image shift.

12.3 Wire Shots

About twenty repeated wire runs of the 1-inch ball were made. This was done by constructing a very tight wire and measuring its exact location. A hole was drilled through one white ball and it ran exactly along x=0 line downward sloping in z as shown in the two right side projections of 21. It was expected that this would reveal motion induced sources of error and it showed that uncertainty of the ball detections was within +/-0.40 inches (0.40 ball diameters) as illustrated in the upper left plot of21, where 0.34 and 0.21 have been combined into an overall 0.40 result. This is approximately four times larger than the still shot uncertainty, some accounted for by wire location uncertainty and other by measurement uncertainty of various sources. This is still small as will be made clear by the next measurement.

The final measurement is shown in the lower left plot of 21. Here it is seen that uncertainties in ball detections combined with extrapolations a long distanct to the target have leveraged into a large overall target strike predictions of +/-2.8 inches (2.8 ball diameters). These errors are accurately confirmed by simple linear extrapolation of the data and amount to the biggest single shortcoming of this system.

Carrying the linear extrapolation point a little further, it would predict that proportional improvement could be achieved by proportional improvements to the components. For example, a four times faster camera frame rate would reduce the errors four times. Such a camera is available using IIDC-compliant devices. Also, a two times denser resolution would reduce the errors two times. A faster computer would achieve this. And a 1/2 inch CCD would improve resolution too.

12.4 Pitches

Finally, thirty-two pitches were made. The cameras, lenses, frame-grabbers, and segmentation all worked as desired and identified several balls per pitch as expected. The single pixel image uncertainty combined with large extrapolation errors resulted in an accuracy of +/-2.5 inches (2.5 ball diameters) on a 3-sigma basis, summarized in 22, in exact agreement with the previous section of more controlled conditions.

13 Conclusions and Recommendations

The method presented here efficiently detects and selects single moving baseballs that are sufficiently lit and have good contrast with the background. The system requires careful setup and zeroing in which result in excellent accuracy within a half ball's width three standard deviations accuracy. Extrapolation from detected ball positions to a distant plate is expectedly poor at about 3 ball width's three standard deviations accuracy. The Pentium II proto-

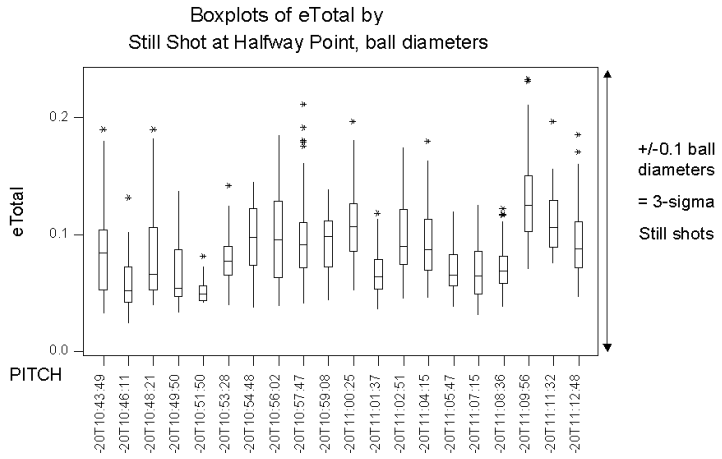


Figure 20:

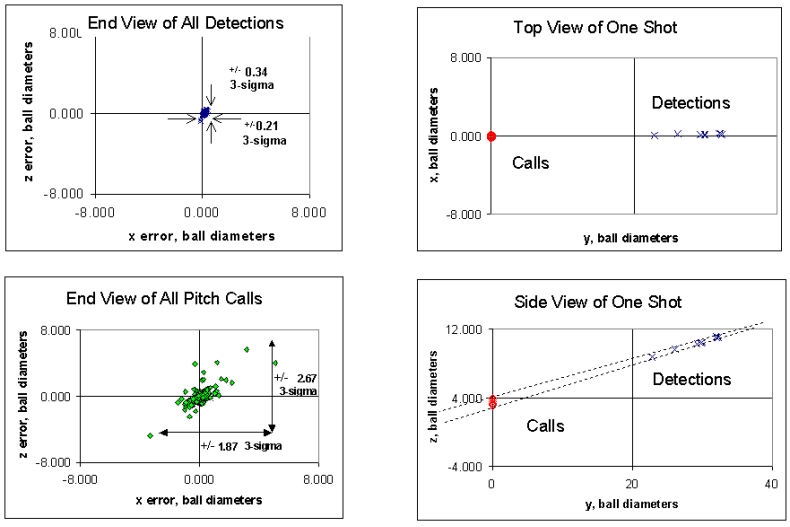


Figure 21:

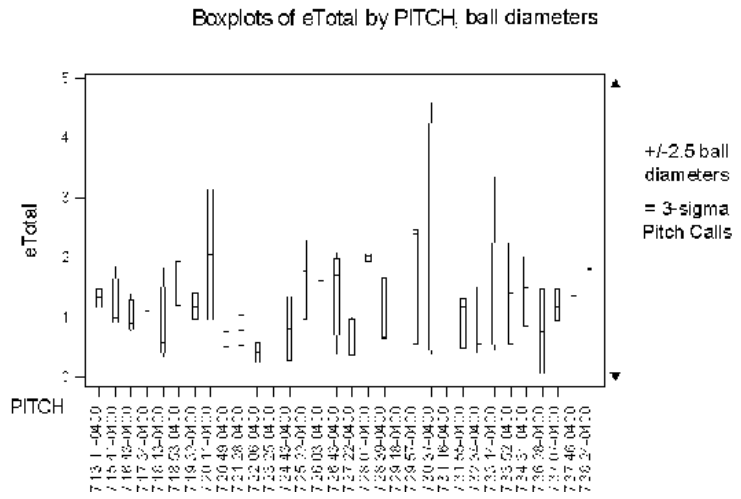


Figure 22:

type works real-time. For stored images, a practical record-playback loop was demonstrated. This mode saves the original image sequence for later analysis. The method performs best with the cameras focused on the field area between pitcher and batter to mask unnecessary player motion. This should be possible given the field of view produced. Accuracy can be improved further using faster computers with larger images and faster frame rates. Camera internal device calibration is unimportant for strike calling accuracy if camera setup is done carefully. The applications and instructions are available for Linux.

It will become necessary at some point to upgrade the program to new interface hardware. Other devices will supersede PCI frame-grabber cards eventually and provide improved resolution. By combining a four times frame rate with two times resolution it should be possible to reduce 3 ball width's error down to 3/8 ball width error.

And finally, it is inevitable to use the method with a 1/2-inch CCD camera in an outdoor setting. There are real lighting issues to be solved - an auto-iris worked well for the prototype. Nighttime lighting effects need to be studied. Also there needs to be a clever way to position the cameras in a weatherproof case at the right height with a rigid mast. Plus, the setup must contend with long wire runs. Given the ease with which the camera and computer equipment produces accurate results, it seems likely that such a prototype will appear at a Little League game near you.

References

[1] Louis Baker. *C Tools for Scientists and Engineers.* McGraw-Hill, 1989.

[2] J. Demmel. Applications of parallel computers: Graph partitioning, part 2, berkeley cs267 lecture notes. http://www.cs.berkeley.edu/ demmel/cs267/lecture18/lecture18.html, April 1996.

[3] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[4] Kay A. Robbins and Steven Robbins. *Practical Unix Programming.* Prentice Hall PTR, Upper Saddle River, New Jersey 07458, 1996.

[5] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Conf. on Comp. Vision and Pattern Recognition*, June 1997.

[6] W. Richard Stevens. *UNIX Network Programming Volume 2 - Interprocess Communication*. Prentice Hall PTR, Upper Saddle River, New Jersey 07458, second edition, 1999.

[7] R. Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[8] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[9] Jonathan Mark Walton. Terrain mapping from a mobile platform through optical flow techniques. Master's thesis, Massachusetts Institute of Technology, 1995.

[10] D. Weinshall Y. Gdalyahu and M. Werman. Stochastic image segmentation by typical cuts. In *IEEE Conf. on Comp. Vision and Pattern Recognition*, June 1999.