Boston College
Computer Science Department

Senior Thesis 2002

Christopher Fagiani

An Evaluation of Tracking Methods for Human-Computer Interaction

Prof. James Gips

Evaluation of Tracking Methods for Human-Computer Interaction

Christopher Fagiani
Computer Science Department
Boston College
Fulton Hall
Chestnut Hill, MA 02467

Abstract

Tracking methods are evaluated in a real-time feature tracking system used for human-computer interaction (HCI). The Camera Mouse, a HCI system that uses video input to manipulate the mouse cursor, was used as the test platform for this study. The Camera Mouse was developed to assist individuals with severe disabilities in using computers, but the technology may be used as a method for HCI for those without disabilities as well. The initial location of the feature to be tracked is set manually by the user by clicking on a portion of the image window and the system tracks its location in subsequent frames. Tracking methods were evaluated in terms of the best combination of computational efficiency and accuracy. Trackers evaluated include feature matching using normalized correlation coefficients and optical flow, in the form of a Lucas-Kanade tracker. Each tracking algorithm is evaluated both with and without the addition of Kalman Filters. The effects of 2-D (x and y location) 4-D(location and velocity in the x and y directions), and 6-D (location, velocity, and acceleration in the x and y directions) are examined and analyzed. The normalized correlation coefficient tracker, without Kalman Filtering, was found to be the best tracker in terms of suitability to the human-computer interaction system implemented.

1 Introduction

The current human-computer interaction paradigm, based around the keyboard and mouse, has seen little change since the advent of modern computing. Since many desktop and laptop computers now come with cameras as standard equipment, it is desirable to employ them for next-generation human-computer interaction devices. By relying on visual input rather than tactile input, users are free to use their hands for other tasks while interacting with their computers, ultimately enabling software designers to develop new models of user interaction.

Systems that interpret visual data have become more prevalent as computer capabilities have expanded. In order to more effectively utilize the power available in standard desktop computers and to expand the range of devices that utilize computer vision techniques, algorithms must be developed that lessen the load on the processor without sacrificing robustness or accuracy. If successful, this will enable applications, such as a camera-based pointing device, to be efficient enough to have only a negligible impact on computer system performance. Such advances would open the door for more intelligent human-computer interfaces, allowing users to move far beyond what is possible with the standard keyboard and mouse combination. The foundation for any such system must be laid with efficiency in mind. Useful applications of vision systems are few if the vision processing takes up the majority of system resources. This being said, it follows that a robust vision system must be quick, accurate and use a reasonable amount of processing power.

Such interfaces also have especially relevant applications for people who cannot use the keyboard or mouse due to severe disabilities. The traditional human-computer interfaces require good manual dexterity and refined motor control. Users with certain disabilities, however, may lack these abilities. The primary goal of this research project is to refine tracking methods currently in use in such an application as to provide a version of the system with less overhead, thereby expanding the range of uses to which it may be applied. As tracking accuracy increases, the amount of user intervention required for system setup and initialization will decrease. This will move human-computer interface technology closer to the possibility of operating without the use of one's hands at all.

The real-time feature tracking system to be used as the testing platform for this experiment is the

Camera Mouse [1,2,21], a system developed at Boston College that uses video input to control the mouse cursor. Upon start-up, a feature is selected for tracking by clicking on it in the vision window. The system then tracks that feature in real time, moving the mouse cursor accordingly.

The purpose of introducing the Kalman Filter [19] is to reduce the amount of processor time needed by the tracking application, thereby allowing for third-party applications to have a greater share of CPU time. The Kalman Filter was applied in two distinct ways: estimating the feature location in every frame and in every other frame. Using the filter every frame allowed for a smaller search area for the tracking algorithm whereas, when using the Kalman filter every other frame, the Kalman estimate was used instead of running the tracking algorithm on that frame. Both methods significantly reduce the CPU time needed by the system, but the scheme in which the Kalman filter is used in every frame is more efficient in terms of CPU load. It is anticipated that tracking systems that are less taxing upon CPUs will eventually enable vision-based tracking systems to be installed in a wide variety of technology, from intelligent vehicles, to almost any other application where head tracking and feature analysis is required.

This paper describes the algorithms and computer vision techniques used to refine a real time feature tracking system in terms of demand on the CPU in such a manner that does not significantly impact tracking accuracy.

Among the methods presented, one employs visual information about the motion of a feature in conjunction with the mathematics of the Kalman Filter to alternately measure and estimate the location of the feature. This method did reduce CPU load, but the reduction in tracking accuracy is prohibitive to using this approach.

The normalized correlation coefficient tracking without any Kalman Filters proved to be the most accurate algorithm. This method, in addition to being fairly accurate, was slightly more demanding on the CPU than the next-best method, the Lucas-Kanade (LK) tracker [18].

Utilizing the normalized correlation coefficient tracker on a smaller search area while using the Kalman filter to estimate feature position every frame proved to be the least CPU intensive method tested. While this was not as accurate as the normalized correlation method without Kalman filters, the processing time was considerably less.

Since computers double in power approximately every eighteen months [17], tracking accuracy was deemed more important than computational efficiency for this application. With this in mind, the normalized correlation coefficient is the tracking method best-suited to this computer vision system.

1.1 Previous Work

Within the field of computer vision, there is a proliferation of different tracking algorithms [6,7,9]. Many are variations upon a common theme [3,5]. Optical flow [4,5,6,7,15], inertial tracking [11], the Condensation algorithm and probabilistic systems [9] have all been used to address the problem of identifying and following movement in visual systems. While, in general, it is possible to regard any algorithm for the computation of the optical flow as a solution to the tracking problem [4], the Lucas-Kanade tracking method addressed the feature tracking problem through the utilization of optical flow to detect motion [14, 5]. The LK method utilized a process of factorization to interpret the image data. Similarly, Kalman Filtering [19] has been applied to many of these techniques [7,9,11] in an attempt to smooth out noise and reduce feature drift [7, 13].

In this paper, a system, which processes real-time image data consistently at approximately 27-29 frames per second is presented. Since the intended use of this system is as a driver for a pointing device, drift of the actual feature being tracked is not relevant, so long as the direction and magnitude of the motion can still be accurately measured. Therefore, minimizing drift, as described in Rahimi et al. [13] is not as appropriate to this application as other types of feature trackers. Even if the feature being tracked drifts to an entirely different region of the face, the system will generally still perform as desired. It is only if drift becomes so pronounced that the feature drifts from the object being tracked to the background that it is detrimental to system performance to the point where it is unusable.

Though still usable with large amounts of drift, methods that have minimal drift are still preferred since the Camera Mouse system becomes much harder to use as the tracked region drifts farther and farther from the true location.

The Camera Mouse system, as described in [1,2,21], has been modified to run on one computer and one program that integrates the functionality of both the driver and the vision tracker. This implementation of the Camera Mouse system requires less hardware than that described in [1,2,21] since the end-users no longer need the digital acquisition board nor the external switch used for alternating between the Camera Mouse and the traditional mouse.

2. System Overview

The system may be broken down into two or four steps, depending on whether or not Kalman Filtering is being employed. If the filters are used, after initialization, the program is in a "training" step in which the error covariance matrix is initialized [12]. After training the error covariance matrix, the feature position is tracked

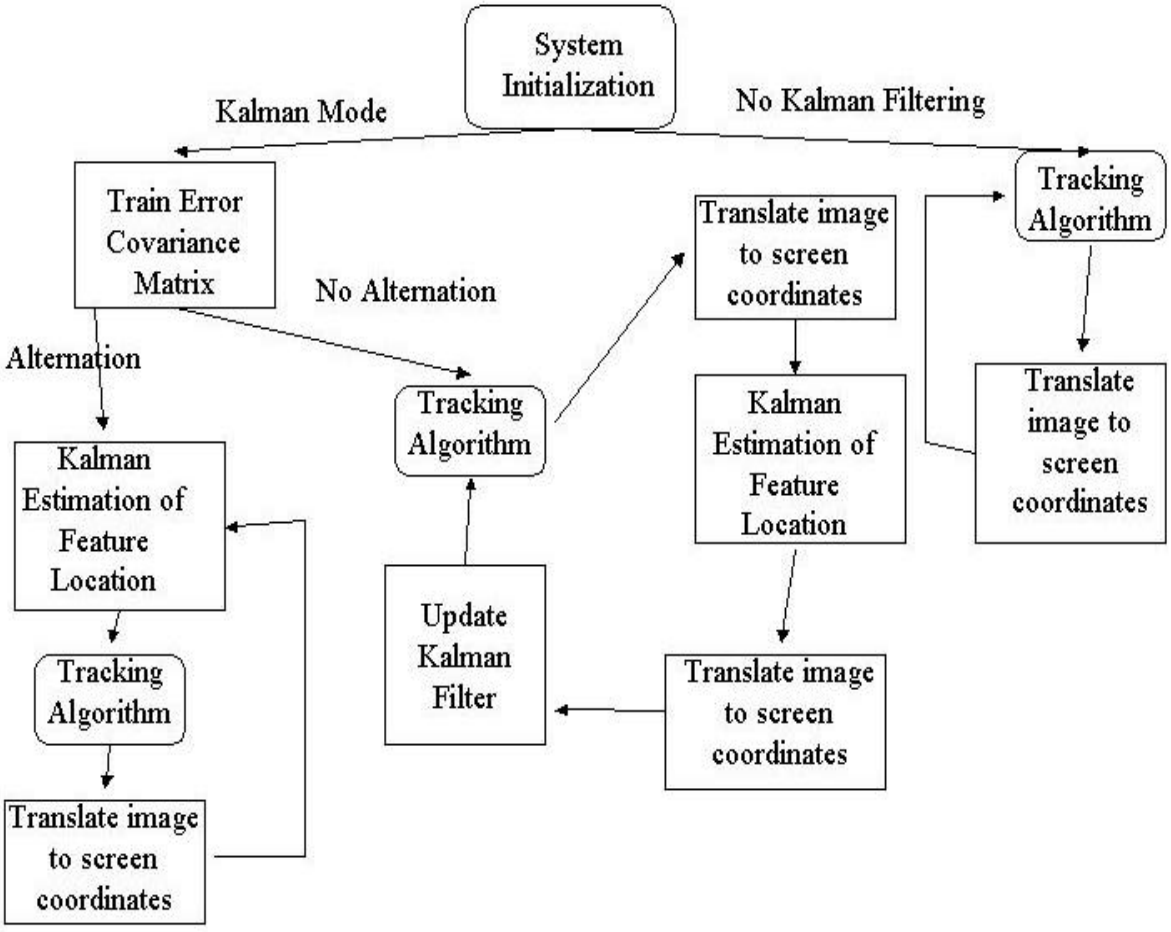

Figure 2.1 The system may be run in one of two branches: Kalman mode and tracking without Kalman Filters. Both modes make use of either the normalized correlation coefficient or the LK tracker, depending on what is selected at start up.

using the chosen tracking algorithm on even frame numbers and estimated using the Kalman Filter on odd frames. After each frame the Kalman filter equation is updated in order to refine its estimate of the position.

When Kalman filtering is not being used, the feature is tracked using only the selected tracking algorithm.

2.1 Normalized Correlation Tracking

The normalized correlation coefficient tracking method computes the correlation coefficient for each point a search area that is centered around the position of the feature in the previous frame [See Equation 2.1.1].

(Eq 2.1.1)

$$R(x,y) = \frac{n\sum_{i=0}^{n} x_i y_i - \sum_{i=0}^{n} x_i \sum_{i=0}^{n} y_i}{\sqrt{\left(n\sum_{i=0}^{n} x_i^2 - \left(\sum_{i=0}^{n} x_i\right)^2\right)}\sqrt{\left(n\sum_{i=0}^{n} y_i^2 - \left(\sum_{i=0}^{n} y_i\right)^2\right)}}$$

where $n \Sigma x_i y_i - \Sigma x_i \Sigma y_i$ is the covariance of the average brightness values of the pixels in the search area, $(n\Sigma x_i^2 - (\Sigma x_i)^2)$ is the standard deviation of the average pixel brightness values in the x direction, and $(n \Sigma y_i^2 - (\Sigma y_i)^2)$ is the standard deviation of the average brightness values in the y direction, and n is the value for the height and width of the feature.

The normalized correlation tracker predicts the location of the feature in the current frame based on the point in the search area that yields a correlation coefficient which best matches the correlation coefficient calculated in the previous frame. Since the tracking is based on search results obtained in the previous frame, this searching algorithm may classified as a recursive filtering method. The search space for equation 2.1.1 is fixed therefore the variance of the correlation estimate is also considered to be fixed.

The search area utilized for this paper was 25 pixels in each direction from the center of the feature (a 50x50 pixel search area). The size of the feature being tracked was 15x15 pixels.

The correlation coefficient acts as a metric for measuring the degree to which a region of the current image matches the corresponding region in the previous frame. This method allows the feature location to be updated at a rate of approximately 30 times per second.

The algorithm operates by calculating the normalized correlation coefficient for every point in the search area, which is centered around the feature's location in the previous frame. The point with the highest correlation is declared to be the new location of the feature. If no point in the search area has a correlation coefficient above a certain threshold, the feature is declared to be "lost" and the system should be re-initialized.

2.2 Lucas-Kanade Tracking

The LK [14, 18] tracker is based upon the principle of optical flow and motion fields [3, 4, 15]. It allows for recovery of motion without assuming a model of motion [14]. The implementation used was that of the Intel OpenCV library.

While the OpenCV implementation of the LK algorithm allows one to track multiple points simultaneously, only one point was tracked in this study. Tracking more than one feature may have improved the LK tracker's performance, but such a method was not implemented in this system.

Optical flow is defined as an apparent flow of image brightness. Calculations of optical flow assume that if I(x,y,t) is the brightness of pixel x,y at time t, then the brightness of every point in the image does not change in time. This is not to say that pixels at the same location in an image sequence must be the same brightness, but rather that the same feature always appears with the same brightness, no matter where in the image it appears.

If the displacement of a pixel from one image frame to another is defined as (dx,dy) then, via the constant brightness assumption, we may state that:

(Eq. 2.2.1)

$$I(x+dx,y+dy,t+dt) = I(x,y,t)$$

Where dt is defined as the difference in time between the new image and the old image.

and:

(Eq 2.2.2)

$$\partial I/\partial x\, dx + \partial I/\partial y\, dy + \partial I/\partial t\, dt + \ddot{E} = 0$$

If u is defined as dx/dt and v is defined as dy/dt, then one may derive the optical flow constraint equation:

(Eq 2.2.3)

$$- \partial I/\partial t = \partial I/\partial x \, u + \partial I/\partial y \, v$$

Feature position estimate is carried out by computing the minimum of the sum of squared differences in pixel brightness values within a given window. Using partial derivatives to minimize the sum of squared errors yields:

(Eq. 2.2.4)

$$\sum_{x,y} W(x,y) I_x^2 \, u + \sum_{x,y} W(x,y) I_x I_y \, v = - \sum_{x,y} W(x,y) I_t I_x$$

and:

(Eq. 2.2.5)

$$\sum_{x,y} W(x,y) I_x I_y \, u + \sum_{x,y} W(x,y) I_y^2 \, v = - \sum_{x,y} W(x,y) I_t I_y$$

Where W(x,y) represents the Gaussian window over which the Lucas-Kanade filter computes the optical flow.

The point generated via equation 2.2.5 is that which minimizes the error in brightness values between the feature in frame t-1 and in frame t. Pixels are grouped according to a window size parameter. In the case of equations 2.2.4 and 2.2.5, the width and height of the window is defined by W(x,y) and x and y represent the coordinates of pixels over which we sum.

2.3 Kalman Mode

The Kalman Filter operates through the repeated updates to the matrices that describe the system. Upon initialization, all that is necessary is the proper specification of the state vector, the observation matrix, transition matrix, dynamics model, and measurement covariance matrix [16]. The time update equations project the current state estimate ahead in time and the measurement update adjusts the projected estimate by an actual measurement that is obtained through the use of, in this system, either the LK tracker or the normalized correlation coefficient tracker.

The Kalman Filter, as implemented in the Intel OpenCV library, produces its estimates through the use of a feedback control: the filter first estimates the system state at some time which it then updates using feedback in the form of measurements. These measurements, however, are subject to process and measurement noise and they must be adjusted accordingly. Because of the noise, the Kalman Filter time-update equations must use an error covariance matrix when projecting the state forward. Similarly, the measurement update equations must update the approximations of the covariance matrix to increase the probability that the system will generate an improved *a posteriori* estimate.

The Kalman Filter measurement is of the form:

(Eq 2.3.1)

$$z_k = H_k + v_k$$

Where v_k is a random variable representing measurement noise and H is a matrix which relates the state to the measurement.

The step in which the Kalman Filter is used to estimate the feature location utilizes the Kalman time update equations:

(Eq. 2.3.2)

$$X^-_{k+1} = A_k \, X_k$$

Where X^-_{k+1} is the *a* priori state estimate at step k given knowledge of the process prior to step k, X_k is the *a posteriori* state estimate at step k given the measurement z_k, and A_k is a matrix which relates the state at time step k to the state at step k+1.

(Eq 2.3.3)

$$P^-_{k+1} = A_k P_k A^T_k + Q_k$$

Where A_k is the same matrix as in Eq. 2.3.2, P^-_k is the *a priori* estimate error covariance, and Q_k is the process noise covariance.

On frames where the tracking algorithm is used to find the feature location, the various matrices that comprise the Kalman Filter are updated using the Kalman Measurement Update Equations:

(Eq. 2.3.4)

$$K_k = P^-_k H^T_k (H_k \, P^-_k H^T_k + R_k)^{-1}$$

Where K is the Kalman gain matrix, R_k is the measurement noise covariance and the other matrices are the same as in equations 2.3.2 and 2.3.3.

(Eq. 2.3.5)

$$X_k = X^-_k + K_k(z_k - H_k X^-_k)$$

Where each term represents the same matrix as in above equations.

(Eq. 2.3.6)

$$P_k = (I - K_k H_k) P^-_k$$

Where each term represents the same matrix as in above equations and I is the identity operator.

Kalman Filters of 3 different dimensions were evaluated in this system. The filters considered were of 2, 4 and 6 dimensions of motion. The two degree filter considered the feature's horizontal and vertical position. The four dimensional filter considered these values as well as velocity in both the horizontal and vertical direction. The six dimensional filter, in addition to the dimensions tracked by the four dimensional filter, considered acceleration in both directions.

The system was designed to implement Kalman Filters in two ways: with and without alternation. Running the system with alternation means that the system will use the Kalman Filter to estimate feature position in odd frames and the tracking algorithm on even frames. Operating without alternation is only applicable to the normalized correlation coefficient algorithm. When the system is run in this mode, the Kalman Filter is used on every frame to estimate the feature position and then the normalized correlation coefficient is used to search the area around the predicted location.

At each iteration, the system approximates the feature location by using equations 2.3.1 through 2.3.6. This method differs from the non-Kalman correlation coefficient method in that the area searched by the normalized correlation coefficient is reduced by three quarters. This translates to a three quarters reduction of the number of times the system will have to compute equation 2.1.1.

When operating in Kalman Mode with alternation, the system has three distinct steps. First, the error covariance matrix is approximated using training data [12]. This was achieved by calculating the covariance of the true feature location and the location predicted by the tracking method (either LK or normalized correlation coefficient) for a given number of frames on a saved input sequence. The true feature locations in this sequence were hand-marked. This allowed for an approximation of the measurement noise covariance encountered by the system.

Once the covariance matrix has been trained, the system tracks the feature using the selected tracking algorithm on even frames using equations 2.3.4 through 2.3.6, and estimates the feature location via the Kalman Filter's *a posteriori* state matrix on the odd frames using the state matrix resulting after the application of equations 2.3.2 and 2.3.3.

The Kalman Filter's state matrix represents the filter's perception of the system state at time t. For the 2-D case, the state matrix reduces to a vector containing the x and y location of the feature. In the 4-dimensional case, the state matrix contains the location and velocity. The 6-dimensional case populates the state matrix with the feature location, velocity and acceleration. The measurement matrix differs from the state matrix in that the system sets it during the non-Kalman frames via the results of applying the chosen tracking algorithm.

When switching from using a 2 dimensional filter to a 4 or 6-D filter, the number of parameters remain the same, but the size of the matrices change. When performing the update, however, one still needs only to update the position of the feature since, according to Kohler, this is the only one, of the three components being measured, that can be directly measured from the video input [10]. Values for acceleration and velocity could be calculated using the position of the feature through a series of frames, but the Kalman filter will calculate these values as needed through the components of the state matrix.

2.4 Motion Translation and Mouse Movement

After the feature location is determined, the movement relative to the previous frame is calculated. This measurement is used to determine how far and in what direction the mouse cursor should be moved on the screen.

The movement of the feature within the image window is translated to screen coordinates through a function that takes into account the screen resolution, the size of the image being analyzed and the apparent motion of the feature (See Equations 2.4.1 and 2.4.2). Before translating image plane coordinates to screen coordinates, the image window is first reduced so that only the interior three quarters of the image is considered (i.e. the edge of the image window does not correspond to the edge of the screen).

(Eq 2.4.1)

$$c_x = \begin{cases} \dfrac{((0.5\,d_x - p_{x,i})m_x)}{0.5\,d_x} + 0.5\,m_x \text{ if } d_x/4 < p_{x,I} < 3d_x/4 \\ 0 \text{ if } p_{x,I} < d_x/4 \\ 1 \text{ if } p_{x,I} > 3d_x/4 \end{cases}$$

Where d_x is the width of the image (in pixels), m_x is the width of the monitor (in pixels), and $p_{x,i}$ is the x coordinate of the predicated feature location in frame i. The image is mirrored in the horizontal direction to make it easier for users to coordinate head movement with screen movement, therefore $0.5d_x - p_{x,I}$ in the above equation is negative when the feature moves to the user's right.

(Eq 2.4.2)

$$c_y = \begin{cases} \dfrac{((0.5\,d_y - p_{y,i})m_y)}{0.5\,d_y} + 0.5\,m_y \text{ if } d_y/4 < p_{y,I} < 3d_y/4 \\ 0 \text{ if } p_{y,I} < d_y/4 \\ 1 \text{ if } p_{y,I} > 3d_y/4 \end{cases}$$

Where d_y is the height of the image (in pixels), m_y is the height of the monitor (in pixels), and $p_{y,i}$ is the y coordinate of the predicated feature location in frame i.

Both of the above equations assume that the monitor resolution has a 3:4 aspect ratio, the same as is used for the input images.

This method of translation makes it easier for the user to move the mouse to the edge of the screen while also reducing the likelihood that the tracker will lose sight of the image due to its movement out of the camera's field of view.

3. Hardware

Input was gathered using a Matrox Meteor II video capture board in conjunction with a Sony EVI-D30 color video CCD camera. Video input was also gathered with a Sony Handycam Hi-8 video camera. The system used for development and testing was a dual 1GHz Pentium III machine with 256 MB of PC-133 SDRAM. Grayscale and color images were processed at 320x240 pixels.

4. Experiments

Experiments were conducted on different series of images representing different ranges of movement. Input images were gathered with the subject moving his head randomly as well as while using a mouse-based spelling program, Midas Touch [See Figure 4.1]. All video data was gathered under normal laboratory lighting (florescent bulbs); no special lights were used.

The subject was asked to make random or erratic movements for some input sequences to simulate the types of movements encountered with users with severe disabilities such as Cerebral Palsy. Such users often have difficulty moving the mouse pointer from one point to another in a straight line with constant speed. Because it is primarily people with disabilities who use the Camera Mouse system, the evaluation of tracking methods needed to account for both smooth and more erratic movement.

Feature selection was performed in such a manner to maximize the likelihood that the system would be able to differentiate the feature from its surrounding area. Areas with significant contrasts in color or texture, such as the eyebrow or nostril, provide regions with sharp edges that facilitate accurate tracking [8]. The edges of these features provide the best regions to track since they usually differ significantly in color from the flesh surrounding them. Features tracked in this study include the edge of the eyebrow closest to the nose, the corner of the nostril, and the bridge of the subject's eyeglasses.

4.1 Quantitative Experiments

Three different input sequences were captured using 3 different people. In one sequence, the subject was asked to make random head

movements and in the other two, the subject used the system to control the Midas Touch spelling application.

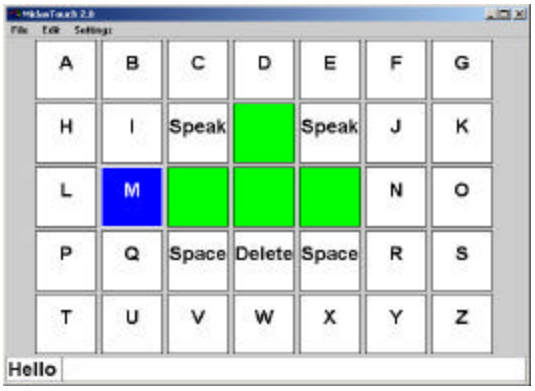

Figure 4.1 The Midas Touch spelling application. Users spell words by moving the mouse cursor over a character and then dwelling within a small radius of pixels for a second to generate a mouse click. Subjects were asked to spell the word "hello" during trials.

When using the Midas Touch spelling program, the subjects were instructed to spell the word "Hello." The subjects did not make any movements that were not associated with performing this task (i.e. subjects did not look away from the screen during input capture), except in the case of the sequence with random movement where the subject was asked to move his head in any direction he wished. The subjects were positioned approximately two feet from the camera and they remained at this distance for the duration of the input sequences.

Evaluation of algorithms was performed in terms of accuracy of the tracking. Accuracy was measured by comparing $P_{d,i}$, a vector representing the distance (in pixels) the mouse cursor should have moved (assuming perfect tracking) in the i^{th} frame, and, $P_{t,i}$, a vector representing the distance the mouse cursor was actually moved in the i^{th} frame [See Equation 4.1]. The mean of the Euclidean distance between these two points over all frames in the input sequence was taken as a single value to quantify tracker performance relative to one another.

(Eq. 4.1)

$$\text{mean error} = \frac{1}{n}\sum_{i=0}^{n-1} \| (\vec{P}_{d,i} - \vec{P}_{t,i})^2 \|$$

Where n is the length of the image sequence (in frames), P_{di} is the number of pixels the tracker predicted the cursor should have been moved in the i^{th} frame, and $P_{t,i}$ is the true number of pixels the mouse cursor should have moved in the i^{th} frame.

This method of evaluation, as opposed to measuring the location of the feature, does not penalize the system for errors made at the beginning of the trial more than those made near the end. The error calculation takes only the change in mouse coordinates from the previous frame to the current frame into account, not the absolute error encountered since the start of the tracking. In this sense, the error is non-cumulative. Similarly, as long as the motion being tracked is in the same direction and magnitude as the actual motion, the tracker is evaluated favorably.

Other evaluations of feature trackers have employed an error metric that measures the drift in feature position from the ground-truth feature position [20]. This evaluation method, while well-suited to systems solely concerned with feature tracking, is less relevant to the Camera Mouse application. The Camera Mouse system does not need to track the same feature to operate properly. The movement of the mouse cursor will remain true to the user's wishes as long as the direction and magnitude of feature motion can still be measured. This distinction makes the effects of drift less significant; therefore the evaluation method must take this into account.

Input image sequences were pre-captured at a rate of 30 frames per second and hand-evaluated. In each series, a feature was selected and the location of that feature in each frame was recorded. Each image in the input sequence was opened in Paint Shop Pro and the location, in pixels, of the center of the chosen feature was noted in a text file. This information was later used by the system to determine the error for the measurement in a given frame. A subset of this data was also used to train the error covariance matrix for the Kalman Filters. Approximately 10 seconds of input was hand-classified (300 frames).

Since the establishing of the "true" feature location in every frame was performed by hand, it is subject to some degree of human error. Though the utmost care was taken to select the exact location of the center of the feature in every frame, some error (on the order of 1 or 2 pixels) may have occurred.

4.2 Qualitative Experiments

Input data of a much longer duration than the hand-classified sequences was recorded on videotape. Video was recorded while the subject used the system to control a number of applications including Midas Touch, Eagle Aliens, Speech Staggered, and Eagle Paint [See Figure 4.2]. Nine minutes of video input was captured in this manner.

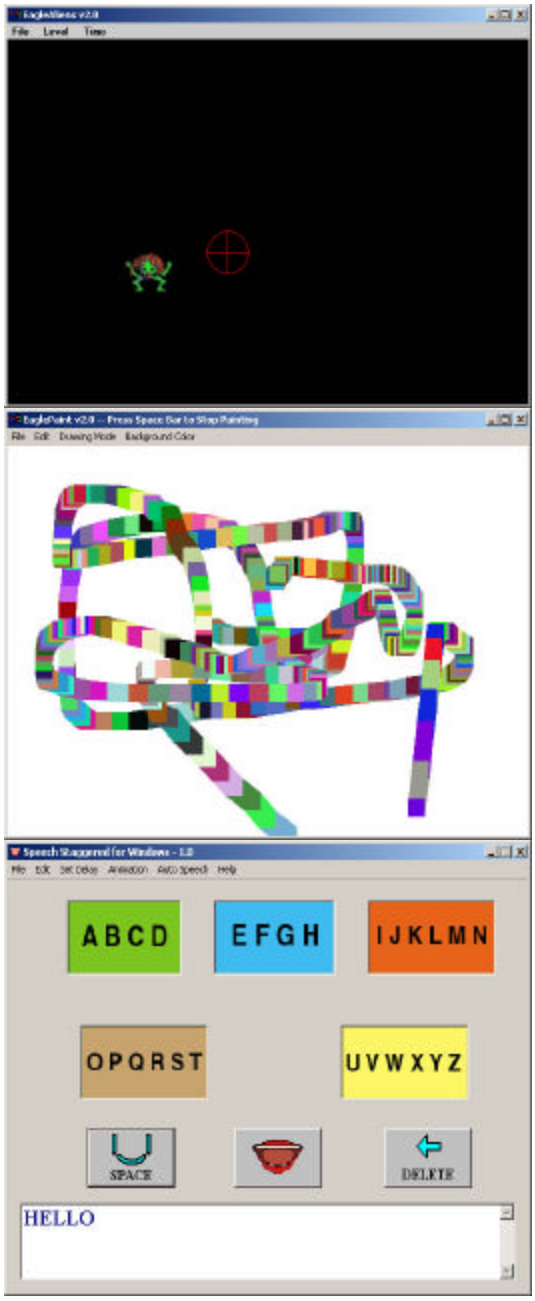

Figure 4.2 The applications used during video capture. From top to bottom: Eagle Aliens, Eagle Paint, and Speech Staggered.

The four different programs were selected because of the different types of movement each requires. Mouse movement while using Eagle Aliens is characterized by sudden changes in direction (as the user is trying to hit aliens that appear in random locations). Erratic, seemingly random movements characterize mouse movement associated with Eagle Paint since the user is able to "paint" by moving the mouse cursor anywhere in the application window. The movement during this application may be slow and deliberate or fast and wild. The two spelling programs, Midas Touch and Speech Staggered are similar in that both require the user to move the mouse over a desired character or group, respectively, and hold the cursor in the same region for a period of time to generate a mouse click.

Evaluation of each tracker on the videotaped data was carried out by hand. If the tracker lost the feature, it was reinitialized. The number of times the operator needed to manually reset the tracker for each tracking method on each input sequence and the amount of drift was recorded. Drift was measured in terms of the degree to which the tracked region deviated from the position to which it was initialized. Due to the length of the input sequences (over 16,200 frames) the true feature location was not recorded for every frame. The drift was estimated through observation of the movement of the tracked region in the video image.

5. Results and Discussion

Qualitatively, the LK tracker without Kalman Filtering and the normalized correlation coefficient tracker performed with similar levels of reliability. Both were accurate enough to allow the user to manipulate the mouse cursor as desired. While the tracking did drift, motion was conserved and therefore it had no apparent effect on the tracking [See Figure 5.1]. When the user moved, the region tracked by the system moved approximately the same distance and in the same direction as the true feature.

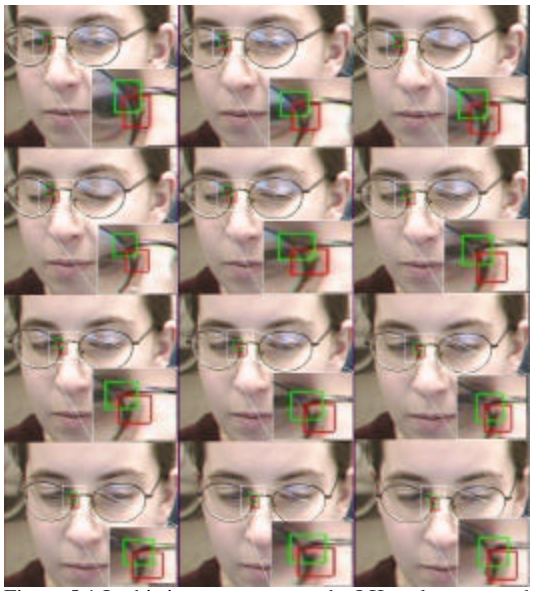

Figure 5.1 In this image sequence, the LK tracker was used without Kalman Filters. The region containing the tracked feature is blown up in each image to make it easier to see. The lighter square denotes the ideal feature location whereas the darker square denotes the location returned by the tracking algorithm. The area tracked in this sequence is the left side of the bridge of the glasses.

Features tracked were deemed suitable for tracking if they were easily differentiable from their immediate surroundings. This quality may be observed by examining the correlation coefficients observed when a template is compared with itself [See Figure 5.2]. Even when the template is only a small number of pixels away from the exact position, the correlation coefficient is seen to be significantly less than 1, the value for a perfect match.

For methods that utilized the Kalman Filter, both the measurement and process noise covariance matrices changed little during training. The process noise matrix, was assumed to be zero for this study. The measurement noise matrix remained fairly stable, with covariance values on the order of 1 and 2 pixels.

The features chosen for this study include the corner of an eyebrow, a nostril, and the bridge of the subject's glasses. All three of these exhibit the above quality, though, of the features used, the eyebrow was the most easily differentiated from itself whereas the glasses template was slightly harder to differentiate from itself. Since all features used in this study exhibited this quality, the choice of the feature itself was deemed to have a negligible impact on the results. Had the features been harder to differentiate from themselves, it is likely that the error rates for each of the tracking methods tested would have been greater.

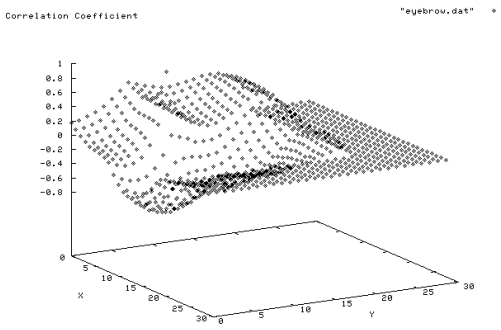

Figure 5.2 A three-dimensional plot of the correlation coefficient values obtained when trying to match a template consisting of the corner of the subject's eyebrow with itself. The peak in the graph represents the location at which an exact match is obtained.

5.1 Results from Quantitative Experiments

When tracking with the normalized correlation coefficient and Kalman filtering (without alternation), the tracking was accurate while the feature movement was relatively slow. Sudden movements, in which the feature moved out of the search area, caused significant drift but did not result in a complete loss of tracking. This is significant since a total loss of the feature would require user intervention to re-initialize the tracker.

Type of Kalman Filter	Correlation Coefficient	Lucas-Kanade Tracker
No Kalman Filtering	CorrNoKalman	LKNoKalman
Kalman Filtering without Alternation	2DCorrNoAlt 4DCorrNoAlt 6DCorrNoAlt	2DLKNoAlt 4DLKNoAlt 6DLKNoAlt
Kalman Filtering with Alternation	2DCorrAlt 4DCorrAlt 6DCorrAlt	2DLKAlt 4DLKAlt 6DLKAlt

Figure 5.3: A summary of the abbreviations used to denote each tracking combination in the subsequent charts in this section. The prefixes 2D, 4D and 6D refer to the dimension of the Kalman Filter being applied.

In terms of processing time, the normalized correlation coefficient tracker with no Kalman Filters was the most demanding. The least demanding was the normalized correlation coefficient with a 4 dimensional Kalman Filter [See Figure 5.4]. Despite the large time required for the normalized correlation tracker, the system was still able to operate at approximately 30 frames per second. Since the increase in computing time still allowed for the processing of real-time data, the cost was deemed acceptable for this type of system.

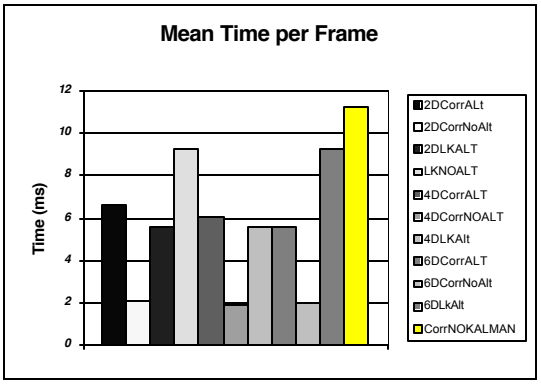

Figure 5.4 The mean time, in milliseconds, needed to process a frame of data using every possible combination of trackers and Kalman Filters implemented in the system.

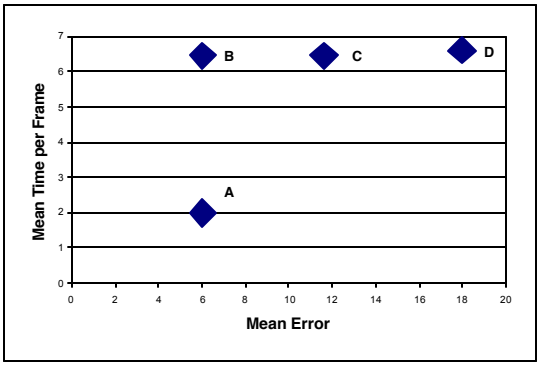

Figure 5.5 Plot of mean tracker error versus computation time. Point A represents a group of trackers consisting of 2DCorrNoAlt, 4DCorrNoAlt, 6DCorrNoAlt. Point B represents the normalized correlation coefficient tracker with no Kalman Filtering and the LK tracker with no Kalman Filtering. Point C represents a group of trackers consisting of 2DCorrAlt, 4DCorrAlt, and 6DCorrAlt. Point D represents the performance of the 2DLKAlt, 4DLKAlt and 6DLKAlt trackers. The best balance between time and accuracy is yielded by the normalized correlation tracker with no Kalman Filtering.

In addition to the differing levels of demand placed upon the CPU, each algorithm performed with a differing degree of accuracy. In some cases, as expected, a decrease in computation time was accompanied by a decrease in accuracy, as when applying Kalman Filtering with alternation to the normalized correlation coefficient tracker. This, however, was not always the case. Though the normalized correlation coefficient tracker used more time to locate the feature than the LK tracker, it was still deemed to be the best suited for this application since it produced a smaller error. This being said, the tracker with the best balance between time and accuracy was the normalized correlation coefficient tracker without Kalman Filtering [See Figure 5.5].

Though the graph in Figure 5.5 shows that the normalized correlation tracker with a 2D Kalman Filter and no alternation performed with approximately the same mean error and a smaller time requirement, that method was not determined to be optimal because of its propensity to drift when the user makes sudden movements. When examined on data in which the subject exhibited random movement, the normalized correlation tracker, when combined with Kalman Filtering, performed with a much higher error than the version without Kalman Filtering [See Figure 5.6].

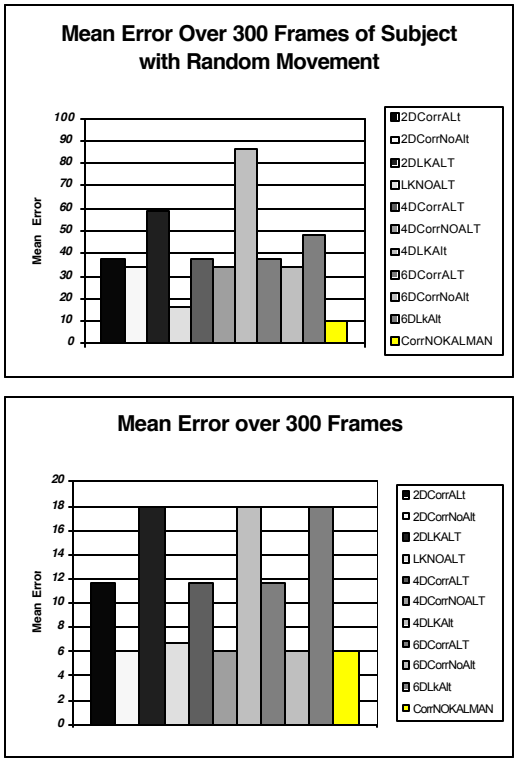

Figure 5.6 Mean error for each tracking method for input with random movement (top) and input in which the subject used the Camera Mouse to control computer applications (bottom). Though the normalized correlation tracker with and

without Kalman Filtering performed similarly on some input, the version without Kalman Filtering worked better on erratic movement. Mean feature velocity in sequence with random movement was 110.2 pixels per frame and the mean velocity in the bottom sequence was 55.8 pixels per frame. Each entry in the graph represents the same tracking method as in Figure 5.3.

On the image series in which the subject was using the Camera Mouse to manipulate the mouse cursor in a third-party application, the normalized correlation coefficient tracker performed with the highest level of accuracy [See Figures 5.7 and 5.8]. Though some drift was observed, the motion recorded was consistent with the movement of the feature and, therefore, the system performed as desired.

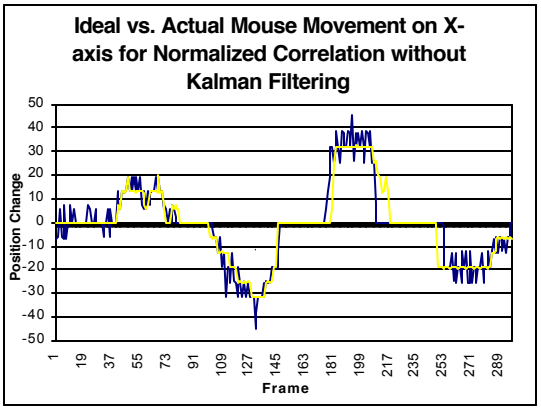

Figure 5.7 The normalized correlation coefficient tracker without Kalman filters performed the best of all tested tracker/filter combinations. The dark line is the actual number of pixels the mouse should have been moved and the lighter line is the amount the system moved the mouse pointer.

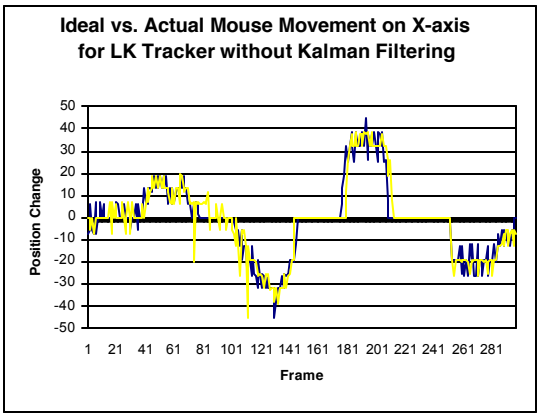

Figure 5.8 The LK tracker without Kalman filters performed the next best of all tested tracker/filter combinations. The dark line is the actual number of pixels the mouse should have been moved and the lighter line is the amount the system moved the mouse pointer. One can see that, though the two lines have a similar level of disparity as those in figure 5.3, this tracking method has a higher mean error due to the more frequent occurrence of large errors. One such occasion may be seen around frame number 110 in this figure.

With random movement, the normalized correlation coefficient-based tracker performed slightly better than the LK tracker and much better than all the other combinations of tracking and filters [See Figure 5.9]. While this behavior is to be expected, based on the trackers' relative performances on the non-random movement, the degree to which the normalized correlation coefficient tracker outperformed the LK tracker was more pronounced on the random data.

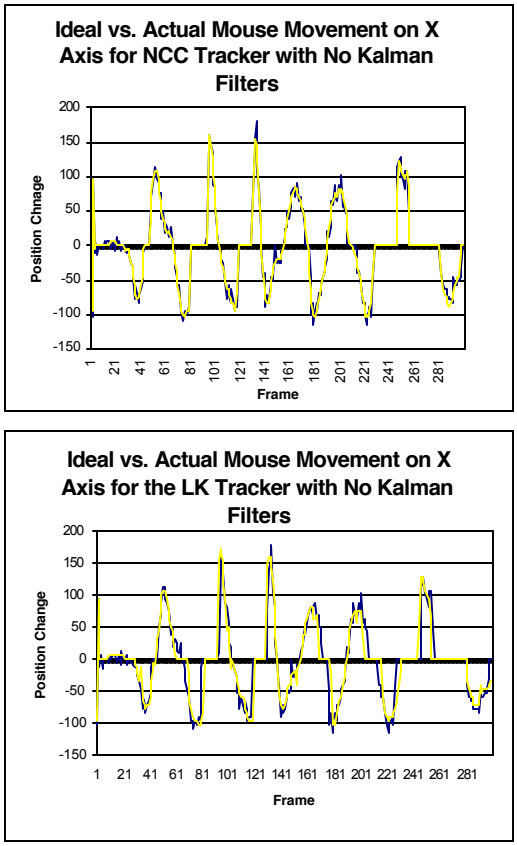

Figure 5.9 Performance of the normalized correlation coefficient (top) and the LK tracker (bottom), both without Kalman Filtering, on data in which the subject exhibited random movement. The performance of the normalized correlation coefficient tracker was slightly better that that of the LK tracker.

All of the trackers were more accurate when measuring horizontal movement than when measuring vertical movement. Even for the most accurate tracker, the normalized correlation coefficient tracker, there was a significant disparity between the accuracy along the X axis and along the Y axis [See Figure 5.10]. This may

have occurred due to the possibility that lighting changes may occur more rapidly with vertical movement than horizontal. Also, because of the position of the camera, moving the head up and down in a nodding movement may cause the tracked feature to become occluded more quickly or to undergo a deformation that makes it harder to track.

Other reasons for the performance difference could lie in the fact that the vertical movement is more erratic than horizontal movement. By comparing the dark lines in the graphs in figures 5.10 and 5.11, one may observe that the vertical motion (Figure 5.10) is more varied than the horizontal motion (Figure 5.11). This is furthered by the examination of the local variance of the mouse movement in the horizontal and vertical directions (See Figures 5.12 and 5.13). The graph for the variance is much smoother in the horizontal direction than the vertical. This may account for why the trackers performed much better in the horizontal direction.

In all of the cases examined, the range of movement was much greater in the horizontal direction. The average range for horizontal movement was 247 pixels, which spans 100% of the range of allowable motion in the x direction. The average range for vertical movement, however, was 88 pixels which only corresponds to 48% of the possible movement in the vertical direction.

The addition of Kalman Filters did not improve the performance of either algorithm. On the frames in which the tracking algorithms were run, the system performed as before, but in the frames in which the Kalman Filter was used to estimate the position, the system recorded a estimated movement of approximately zero pixels [See Figure 5.11].

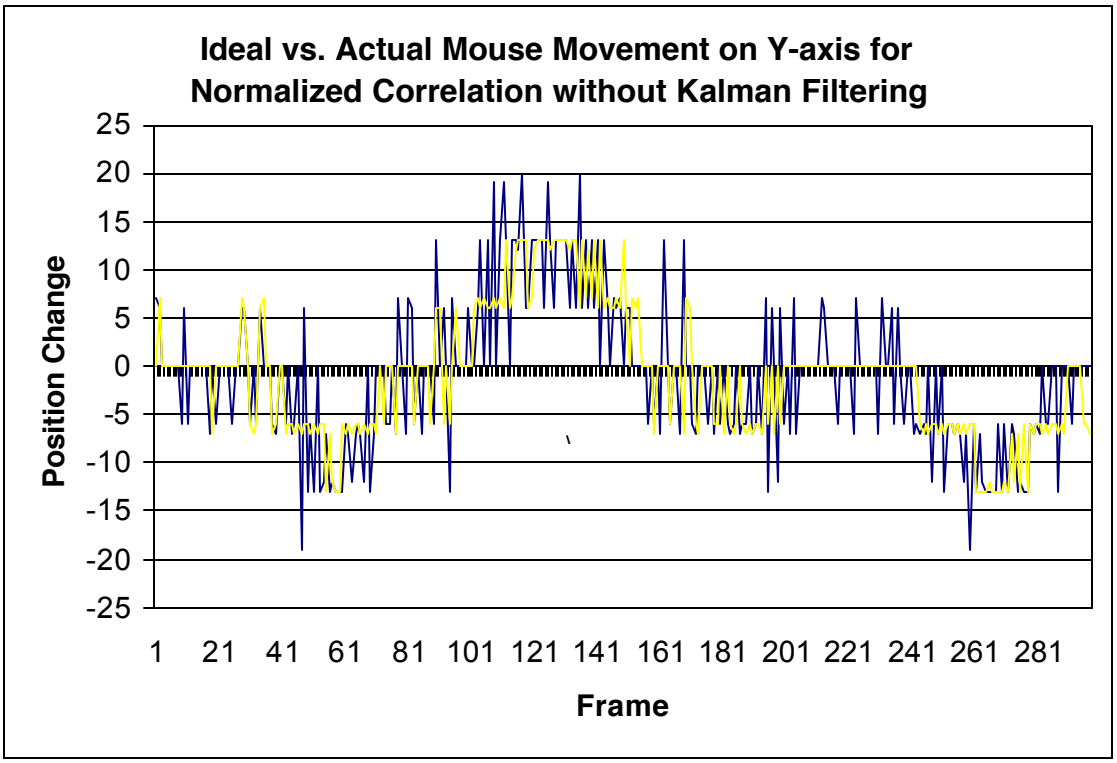

Figure 5.10 Even for the most accurate tracker, the normalized correlation coefficient tracker, tracking along the vertical axis was not as accurate as along the horizontal axis.

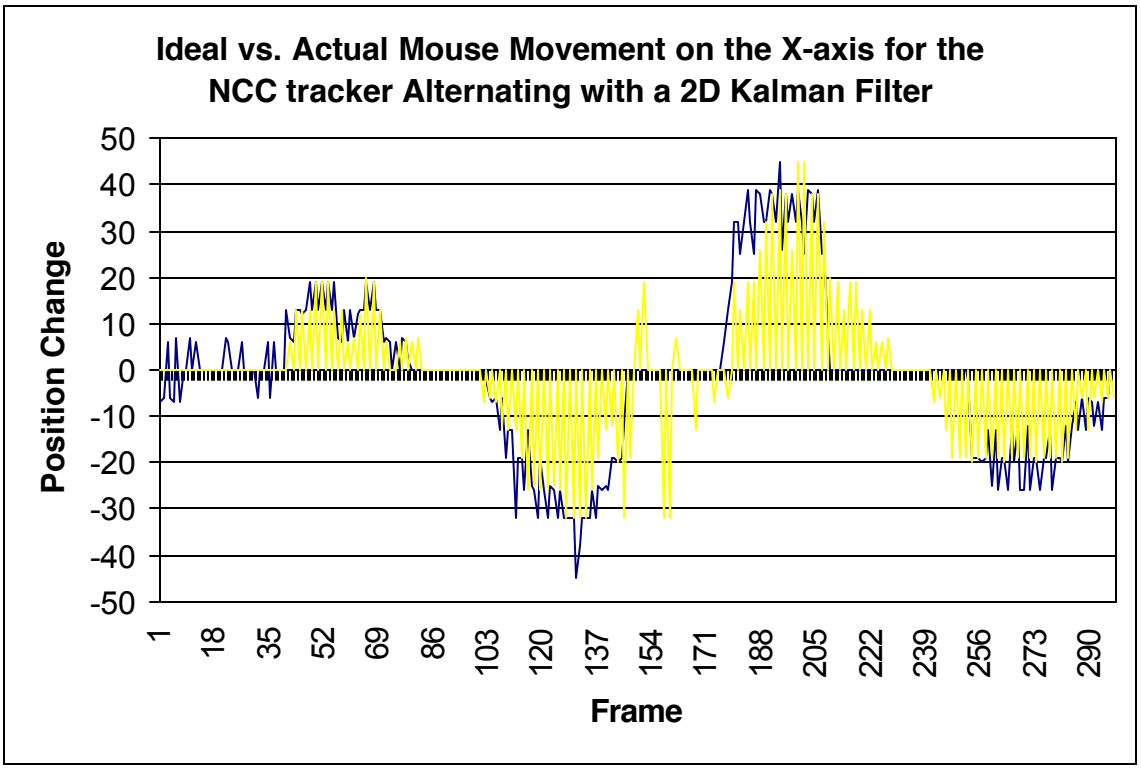

Figure 5.11 Performance of the normalized correlation coefficient tracker with 2D Kalman filtering and alternation. The thrashing exhibited by the graph is indicative of the zero change readings output by the Kalman Filter in odd frames.

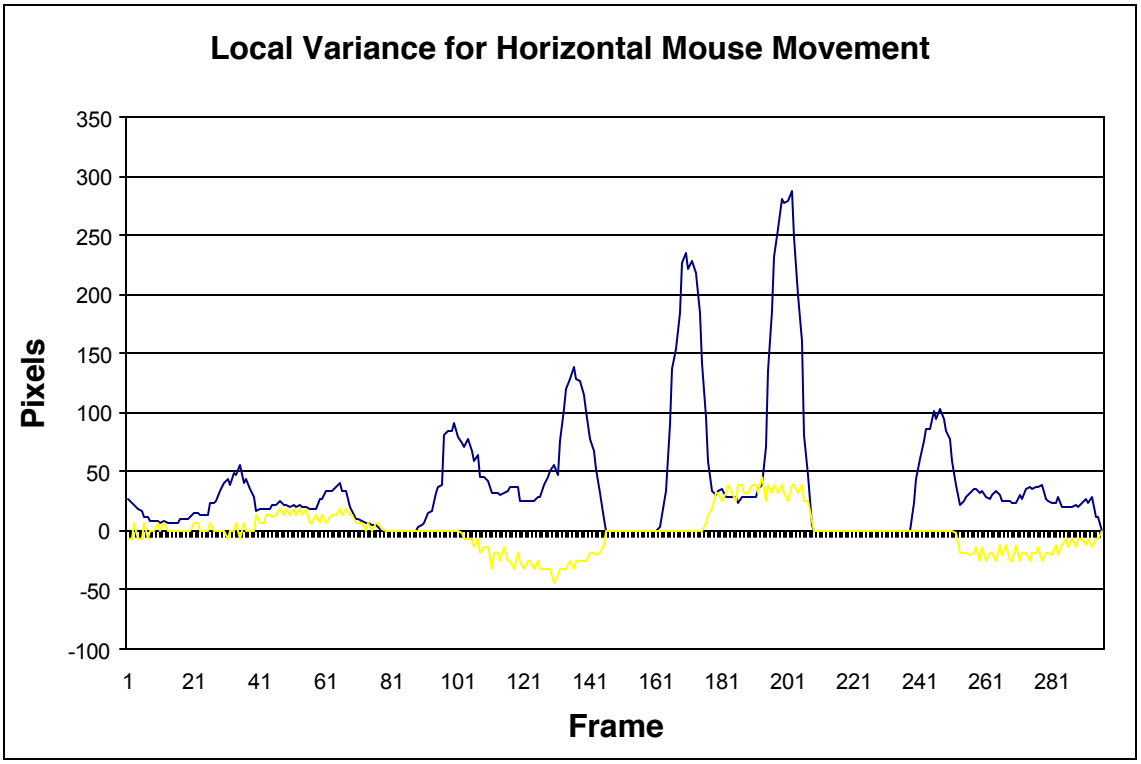

Figure 5.12 The local variance for the true mouse movement in the y direction.

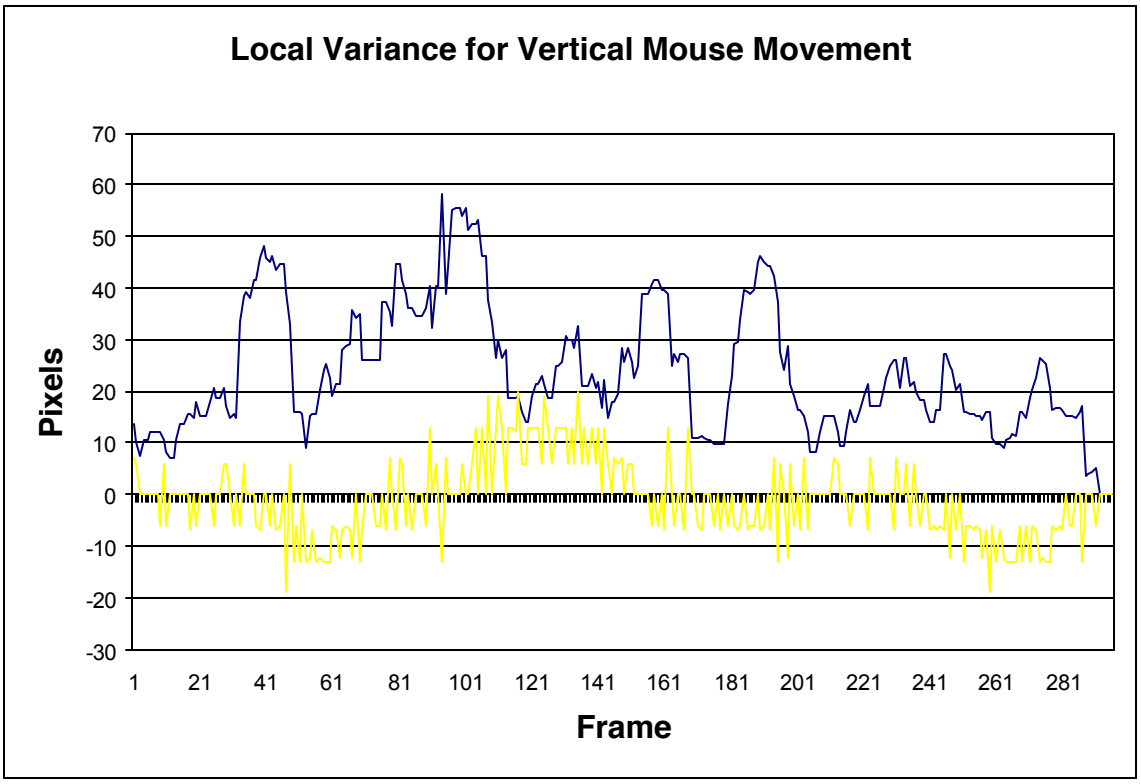

Figure 5.13 The local variance for the true mouse movement in the y direction. The local variance was calculated with a sliding window of size 15. The graph for the horizontal movement (figure 5.12) is much smoother than that for the vertical movement which suggests that tracking in the horizontal direction is easier for the system since the movements are much less erratic and are of higher magnitudes.

Tracking methods employing Kalman Filtering with alternation exhibited a higher error level on frames in which the Kalman Filter was used to estimate feature position [See Figure 5.14]. For frames in which the tracking algorithm was used instead of the Kalman Filter, the system was approximately 2 times more accurate.

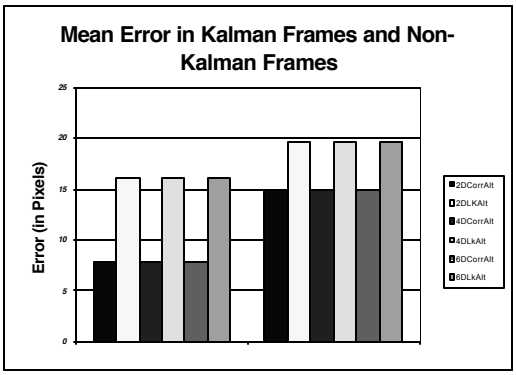

Figure 5.14 For trackers using Kalman Filtering, the error for frames in which the Kalman Filter was used to estimate feature position was approximately 2 times larger than in those which utilized the tracking algorithm. The set of bars on the left side of the graph represent error in "non-Kalman" frames and the set of bars on the right represent errors encountered for frames in which the Kalman Filter was used for feature estimation.

5.2 Results from Qualitative Experiments

Tracker performance was comparable to the above results for the videotaped input as well. Both the LK tracker with no Kalman Filtering and the normalized correlation coefficient tracker with no Kalman Filtering performed well. The feature was never lost during the trials. The LK tracker exhibited minimal drift while the feature drifted from one eyebrow to the other when the normalized correlation coefficient tracker was employed. This drift corresponds to only approximately 5 millimeters on the subject's face.

The normalized correlation coefficient tracker with Kalman Filtering (2, 4, and 6 dimensions) did not lose the feature, but drift was more apparent. Drift in the 2D case was similar to that observed in the case without any Kalman Filtering, but the 4 and 6 dimensional cases

exhibited slightly more pronounced drift, as the feature drifted from one eyebrow to the other and then to the eyelid. This corresponds to approximately 10 millimeters. The drift was observed while the subject was using Eagle Aliens. This behavior is to be expected since the Eagle Aliens application is characterized by much more rapid movements than the spelling applications. Since the subject may be changing direction of movement suddenly, it is not surprising that the Kalman Filter yields poor estimates of feature location.

When the LK tracker was used in conjunction with Kalman Filtering, the tracker lost the feature multiple times. In the 2-D and 4-D cases, the feature was lost twice per sequence tested and, in the 6D case, the feature was lost an average of nine times on each sequence tested. In each of the cases where the feature was lost, the tracked region drifted from the eyebrow to the forehead, then off the face and onto the background. This corresponded to approximately 40 millimeters of drift along the subject's face. In addition to drifting along the face, the tracking performed by LK tracker with 4D and 6D Kalman Filters was jittery thereby making it difficult for the subject to generate mouse clicks by holding the cursor steady for one second. Drift was encountered while the subject was using both Eagle Paint and Eagle Aliens, both of which are characterized by faster movement than the spelling applications used in the trials .

6. Conclusions and Future Work

Based on the empirical evidence, the normalized correlation coefficient tracker is the best suited for this human-computer interaction real-time vision application. The algorithm, though not the least computationally expensive, provides a highly-accurate tracker. This is of the utmost importance for a system used as a driver for a pointing device. The more reliable the tracker, the less often the user will have to intervene to relocate the search area manually. This is even more important for a system such as the Camera Mouse, which is primarily used by individuals with disabilities. The fact that the algorithm is many times more expensive than the normalized correlation coefficient is mitigated by the fact that computer processing power is still doubling approximately every 18 months [17].

Though the Kalman Filter is applicable to many different tracking situations, the Camera Mouse is an application for which it does not always perform optimally, especially when the subject exhibited erratic movement. A conjecture for an explanation of this is that human movement may be neither smooth nor Gaussian. Based on the experiments performed, the Kalman Filter was not found to improve system performance significantly.

Further improvements could be made upon the Camera Mouse system by augmenting the LK tracker with basic logic and checking for physical impossibilities. Occasionally, the location returned by the tracker is so far from the last position that it cannot be accurate. Incorporating such an outlier-rejection method would reduce the number of times a feature is lost.

An additional problem encounter with the Camera Mouse is the occlusion of some features as the subject moves. The occurrence of this event can be reduced by the conscientious selection of a feature to track, as some features are better-suited to tracking than others. Of the features tracked in this study, the eyebrow proved to be the best, (see Section 5) but further testing may identify additional suitable features. Beyond this, however, it would be beneficial to develop methods to identify and rank features other than the one being tracked at any given moment. This would establish a framework for selecting a different feature upon which to base the tracking if the feature currently being tracked becomes undesirable (due to occlusion, poor lighting, or another factor). Tracking multiple features and selecting between them adds to the computational complexity of the system therefore one must take care to ensure the system is still able to operate in real-time. When complex error checking can be applied in real-time, computer vision systems will have the accuracy needed to be stable and reliable HCI devices.

Acknowledgements

I would like to thank my advisors, Professor James Gips and Professor Margrit Betke for their guidance and support during this project.

References

[1] J. Gips, M. Betke, and P. Fleming. The Camera Mouse: Preliminary investigation of automated visual tracking for computer access. In *Proceedings of the RESNA 2000 Annual Conference*, Orlando, FL, July 2000.

[2] The Camera Mouse Project at Boston College. http://www.cs.bc.edu/~gips/CM

[3] M. Otte and H.H. Nagel. Optical Flow Estimation: Advances and Comparisons. *Proceedings of the Third European Conference on Computer Vision,* pages 51-60, Stockholm, Sweden, May 1994.

[4] M.Tistarelli. Multiple Constraints for Optical Flow. *Proceedings of the Third European Conference on Computer Vision,* pages 61-70, Stockholm, Sweden, May 1994.

[5] U. Neumann and S. You. Integration of region Tracking and Optical Flow for Image Motion Estimation. http://graphics.usc.edu/cgit/pdf/papers/Icip98-full-un.pdf

[6] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance. http://www.cs.berkeley.edu/~zephyr/resume/TR-Crw.pdf

[7] T. Tommasini, A. Fusiello, V. Roberto, E. Trucco. Robust Feature Tracking. *Proceedings of the IAPR-AIIA Workshop on Artificial Intelligence and Pattern Recognition Methods,* pages 93-98, Ferrara, Italy, April 1998.

[8] S. Gil, R. Milanese, and T. Pun. Feature Selection for Object Tracking in Traffic Scenes. *Proceedings of SPIE Conference on Photonic Sensors and Controls for Commercail Applications – Intelligent Vehicle Highway Systems,* pages 253-266, Boston, November 1994.

[9] T-J. Cham and J. Rehg. A Multiple Hypothesis Approach to Figure Tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, volume 2*, pages 239-244, Fort Collins, 1999.

[10] M. Kohler. Using the Kalman Filter to Track Human Interactive Motion – Modeling an Initialization of the Kalman Filter for Translational Motion. Technical Report 629, Informatik VII, University of Dortmund, January 1997.

[11] W. Pasman, S. Zlatanova, S. Persa, J. Caarls. Alternatives for Optical Tracking. *International Report, UbiCom*, May, 2001.

[12] J. Hoffbeck and D. Landgrebe. Covariance Matrix Estimation and Classification with Limited Training Data. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 7*, pages 763 - 767, July 1996.

[13] A. Rahimi, L-P. Morency, T. Darrell. Reducing Drift in Parametric Motion Tracking. In *Proceedings of the Eighth IEEE International Conference on Computer Vision, Volume* 1, pages 315-322, June 2001.

[14] C. Tomasi and T. Kanade. Shape and Motion from Image Streams Under Orthography: A Factorization Approach. *International Journal of Computer Vision*, Volume 9, Number 2, pages 137-154, 1992.

[15] B.K.P. Horn and B. G. Schunck. Determining Optical Flow. *Artificial Intelligence,* 17(1-3), pages 285 – 203, August 1981.

[16] E. Brookner. *Tracking and Kalman Filtering Made Easy*. John Wiley & Sons Publishing, page 75, 1998.

[17] G. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, Volume 38, Number 8, April 19, 1965.

[18] Lucas and Kanade. An Iterative Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981.

[19] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, Trans. of ASME, Ser. D, Vol. 82, No. 1, pages 35-45, March 1960.

[20] H. Jin, P. Favaro, and S. Soatto. Real-Time Feature Tracking and Outlier Rejection with Changes in Illumination. In *Proceedings of the Eighth IEEE International International Conference on Computer Vision*, *Volume 1*, pages 684-689, July 2001.

[21] M. Betke, J. Gips, and P. Fleming. The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access For People with Severe Disablilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. In press, April 2002.