Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches

Dongheng Li, David Winfield, Derrick J. Parkhurst
Human Computer Interaction Program
Iowa State University, Ames, Iowa, 50010

Abstract

Knowing the user's point of gaze has significant potential to enhance current human-computer interfaces, given that eye movements can be used as an indicator of the attentional state of a user. The primary obstacle of integrating eye movements into today's interfaces is the availability of a reliable, low-cost open-source eye-tracking system. Towards making such a system available to interface designers, we have developed a hybrid eye-tracking algorithm that integrates feature-based and model-based approaches and made it available in an open-source package. We refer to this algorithm as "starburst" because of the novel way in which pupil features are detected. This starburst algorithm is more accurate than pure feature-based approaches yet is significantly less time consuming than pure model-based approaches. The current implementation is tailored to tracking eye movements in infrared video obtained from an inexpensive head-mounted eye-tracking system. A validation study was conducted and showed that the technique can reliably estimate eye position with an accuracy of approximately one degree of visual angle.

1. Introduction

The use of eye tracking has significant potential to enhance the quality of everyday human-computer interfaces. Two types of human-computer interfaces utilize eye-movement measures – active and passive interfaces. Active interfaces allow users to explicitly control the interface though the use of eye movements [8]. For example, eye typing has users look at keys on a virtual keyboard to type instead of manually depressing keys as on a traditional keyboard [9]. Such active interfaces have been quite effective at helping users with movement disabilities interact with computers. These techniques may also be useful for normal interface usage given that when users intend to select an icon in a graphical user interface, they typically first look at the icon and thus selection can potentially be speeded with eye tracking [16]. On the other hand, passive interfaces monitor the user's eye movements and automatically adapt themselves to the user. For example in video transmission and virtual reality applications, gaze-contingent variable-resolution display techniques actively track the viewer's eyes and present a high level of detail at the point of gaze while sacrificing level of detail in the periphery where it is not distracting [13, 14].

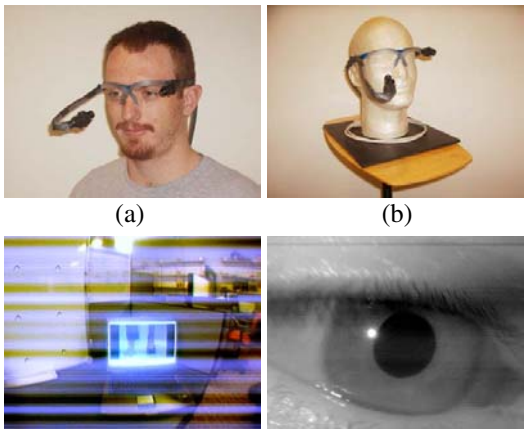

Figure 1: (a)&(b) Head-mounted eye tracker (c) Image of a scene obtained by the eye tracker. (d) Image of the user's right eye illuminated with infrared light. Note the clearly defined dark pupil and the specular reflection of the infrared LED. Also note the degree of line noise present in (c)& (d) due to the low-cost construction based on consumer-grade off-the-shelf parts.

While eye tracking has been deployed in a number of research systems and to some smaller degree consumer products, eye tracking has not reached its full potential. Importantly, eye tracking technology has been available for many years using a variety of methods (e.g., Purkinje-reflection based, contact-lens based eye coil systems, electro-oculography; see [19] for a survey of classical eye-tracking technology). The primary obstacle to integrating these techniques into human-computer interfaces is that they have been either too invasive or too expensive for routine use. Recently, the invasiveness of eye tracking has been significantly reduced with advances in the miniaturization of head-mounted video-based eye-trackers [15, 1]. Remote video-based eye tracking techniques also minimize intrusiveness [6, 10], however can suffer from reduced accuracy with respect to head-mounted systems. Given these advances, the most significant remaining obstacle is the cost. Currently, a number of eye trackers are available on

Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)
1063-6919/05 $20.00 © 2005 **IEEE**

the market and their prices range from approximately 5,000 to 40,000 US Dollars. Notably, the bulk of this cost is not due to hardware, as the price of high-quality digital camera technology has dropped precipitously over the last ten years. Rather, the costs are associated with custom software implementations, sometimes integrated with specialized digital processors, to obtain high-speed performance.

This analysis clearly indicates that in order to integrate eye tracking into everyday human-computer interfaces, the development of widely available, reliable and high-speed eye-tracking algorithms that run on general computing hardware need to be implemented. Towards this goal, we have developed a hybrid eye-tracking algorithm that integrates feature-based and model-based approaches and made its implementation available for distribution in an open-source package. In combination with low-cost head-mounted eye-tracking systems [18], there is a significant potential that eye tracking will be successfully incorporated into the next generation of human-computer interfaces.

2. Problem statement

As mentioned above, eye-tracking systems can be divided into remote and head-mounted systems. Each type of system has its respective advantages. For example, remote systems are not as intrusive but are not as accurate or flexible as head-mounted systems. In other work, we have developed a low-cost head-mounted eye tracker [18]. This eye tracker consists of two consumer-grade CCD cameras that are mounted on a pair of safety glasses (see Figure 1). One camera captures an image of the eye while the other captures an image of the scene. The two cameras are synchronized and operate at 30hz each capturing 640x480 pixels. In this paper we develop an eye-tracking algorithm applicable for use with images captured from this type of head-mounted system. However, the proposed algorithm could also be applied to video captured with a remote system.

Two types of imaging processes are commonly used in eye tracking, visible and infrared spectrum imaging [5]. Visible spectrum imaging is a passive approach that captures ambient light reflected from the eye. In these images, it is often the case that the best feature to track in visible spectrum images is the contour between the iris and the sclera known as the limbus. The three most relevant features of the eye are the pupil - the aperture that lets light into the eye, the iris - the colored muscle group that controls the diameter of the pupil, and the sclera, the white protective tissue that covers the remainder of the eye. Visible spectrum eye tracking is complicated by the fact that uncontrolled ambient light is used as the source, which can contain multiple specular and diffuse components. Infrared imaging eliminates uncontrolled specular reflection by actively illuminating the eye with a uniform and controlled infrared light not perceivable by the user. A further benefit of infrared imaging is that the pupil, rather than the limbus, is the strongest feature contour in the image (see e.g., Figure 1d); both the sclera and the iris strongly reflect infrared light while only the sclera strongly reflects visible light. Tracking the pupil contour is preferable given that the pupil contour is smaller and more sharply defined than the limbus. Furthermore, due to its size, the pupil is less likely to be occluded by the eye lids. The primary disadvantage of infrared imaging techniques is that they cannot be used outdoors during daytime due to the ambient infrared illumination. In this paper, we focus our algorithm development on infrared spectrum imaging techniques but aim to extend these techniques to visible spectrum imaging as well.

Infrared eye tracking typically utilizes either bright-pupil or dark-pupil techniques (however see [10] for the combined use of both bright-pupil and dark-pupil techniques). Bright-pupil techniques illuminate the eye with a source that is on or very near the axis of the camera. The result of such illumination is that the pupil is clearly demarcated as a bright region due to the photoreflective nature of the back of the eye. Dark-pupil techniques illuminate the eye with an off-axis source such that the pupil is the darkest region in the image, while the sclera, iris and eye lids all reflect relatively more illumination. In either method, the first-surface specular reflection of the illumination source off of the cornea (the outer-most optical element of the eye) is also visible. This vector between the pupil center and the corneal reflection is typically used as the dependent measure rather than the pupil center alone. This is because the vector difference is insensitive to slippage of the head gear - both the camera and the source move simultaneously (see the results of our validation study, below). In this paper we focus on algorithm development for dark-pupil techniques however our algorithm could be readily applied to bright-pupil techniques.

3. Related Work

Eye-tracking algorithms can be classified into two approaches: feature-based and model-based approaches. Feature-based approaches detect and localize image features related to the position of the eye. A commonality among feature-based approaches is that a criteria (e.g., a threshold) is needed to decide when a feature is present or absent. The determination of an appropriate threshold is typically left as a free parameter that is adjusted by the user. The tracked features vary widely across algorithms but most often rely on intensity levels or intensity gradients. For example, in infrared images created with the dark-pupil technique, an appropriately set intensity threshold can be used to extract the region corresponding to the pupil. The pupil center can be taken as the geometric center of this identified region. The intensity gradient can be used to detect the limbus in visible spectrum images [21] or the pupil contour in

infrared spectrum images [12]. An ellipse can then be fitted to these feature points.

On the other hand, model-based approaches do not explicitly detect features but rather find the best fitting model that is consistent with the image. For example, integro-differential operators can be used to find the best-fitting circle [3] or ellipse [11] for the limbus and pupil contour. This approach requires an iterative search of the model parameter space that maximizes the integral of the derivative along the contour of the circle or ellipse. The model-based approach can provide a more precise estimate of the pupil center than a feature-based approach given that a feature-defining criteria is not applied to the image data. However, this approach requires searching a complex parameter space that can be fraught with local minima. Thus gradient techniques cannot be used without a good initial guess for the model parameters. Thus, the gain in accuracy of a model-based approach is obtained at a significant cost in terms of computational speed and flexibility. Notably however, the use of multi-scale image processing methods [2] in combination with a model-based approach hold promise for real-time performance [5].

4. Starburst Algorithm

Presented in this section is an eye-tracking algorithm that combines feature-based and model-based approaches to achieve a good tradeoff between run-time performance and accuracy for dark-pupil infrared illumination. The goal of the algorithm is to extract the location of the pupil center and the corneal reflection so as to relate the vector difference between these measures to coordinates in the scene image. The algorithm begins by locating and removing the corneal reflection from the image. Then the pupil edge points are located using an iterative feature-based technique. An ellipse is fitted to a subset of the detected edge points using the Random Sample Consensus (RANSAC) paradigm [4]. The best fitting parameters from this feature-based approach are then used to initialize a local model-based search for the ellipse parameters that maximize the fit to the image data.

4.1. Noise Reduction

Due to the use of a low-cost head-mounted eye tracker described in Section 2, we need to begin by reducing the noise present in the images. There are two types of noise, shot noise and line noise. We reduce the shot noise by applying a 5×5 Gaussian filter with a standard deviation of 2 pixels. The line noise is spurious and a normalization factor can be applied line by line to shift the mean intensity of the line to the running average derived from previous frames. This factor C for each line l in frame i is

$$C(i,l) = \beta \bar{I}(i,l) + (1-\beta)C(i-1,l) \qquad (1)$$

where $\bar{I}(i,l)$ is the average line intensity and $\beta = 0.2$. Note that this noise reduction technique is optional and can be eliminated when the algorithm is used in combination with an eye tracker capable of capturing less noisy images.

4.2. Corneal reflection detection, localization and removal

The corneal reflection corresponds to one of the brightest regions in the eye image. Thus the corneal reflection can be obtained through thresholding. However, a constant threshold across observers and even within observers is not optimal. Therefore we use an adaptive thresholding technique in each frame to localize the corneal reflection. Note that because the cornea extends approximately to the limbus, we can limit our search for the corneal reflection to a square region of interest with a half width of $h = 150$ pixels (see the Discussion section regarding parameter values). To begin, the maximum threshold is used to produce a binary image in which only values above this threshold are taken as corneal reflection candidates. It is likely that the largest candidate region is attributable to the corneal reflection, as other specular reflections tend to be quite small and located off the cornea as well as near the corner of the image where the eye lids meet. The ratio between the area of the largest candidate and the average area of other regions is calculated as the threshold is lowered. At first, the ratio will increase because the corneal reflection will grow in size faster than other areas. Note that the intensity of the corneal reflection monotonically decreases towards its edges, explaining this growth. A lower threshold will, in general, also induce an increase in false candidates. The ratio will begin to drop as the false candidates become more prominent and the size of the corneal reflection region becomes large. We take the threshold that generates the highest ratio as optimal. The location of the corneal reflection is then given by the geometric center (x_c, y_c) of the largest region in the image using the adaptively determined threshold.

Given its small size, the corneal reflection is approximately a circle in the image. While the approximate size of the corneal reflection can be derived using the thresholded region from the localization step, this region does not typically include the entire profile of the corneal reflection. To determine the full extent of the corneal reflection, we assume that the intensity profile of the corneal reflection follows a bivariate Gaussian distribution. If we find the radius r where the average decline in intensity is maximal and relate it to the radius with maximal decline for a Gaussian (i.e. a radius of one standard deviation), we can take the full extent of the corneal reflection as $2.5r$ to capture 99% of the corneal reflection profile. We find r through a gradient decent search that minimizes

$$\frac{\int I(r+\delta, x_c, y_c, \theta)\, d\theta}{\int I(r-\delta, x_c, y_c, \theta)\, d\theta} \qquad (2)$$

where $\delta = 1$, and $I(r, x, y, \theta)$ is the pixel intensity at angle θ on the contour of a circle defined by the parameters r, x

Input: Eye image with corneal reflection removed, Best guess of pupil center
Output: Set of feature points
Procedure:
Iterate
  Stage 1:
  Follow rays extending from the starting point
  Calculate intensity derivative at each point
  If derivative $>$ threshold then
    Place feature point
    Halt marching along ray
  Stage 2:
  For each feature point detected in Stage 1
    March along rays returning towards the start point
    Calculate intensity derivative at each point
    If derivative $>$ threshold then
      Place feature point
      Halt marching along ray
  Starting point = geometric center of feature points
Until starting point converges

Figure 2: Feature-point detection method

and y. The search is initialized with $r = \sqrt{area/pi}$, where $area$ is the number of pixels in the thresholded region. The search converges rapidly.

Radial interpolation is then used to remove the corneal reflection. First, the central pixel of the identified corneal reflection region is set to the average of the intensities along the contour of the region. Then for each pixel between the center and the contour, the pixel intensity is determined via linear interpolation. An example of this process can be seen in Figure 5 (compare a and b).

4.3. Pupil contour detection

We have developed a novel feature-based method to detect the pupil contour. The pseudo code that describes the algorithm is shown in Figure 2. While other feature-based approaches apply edge detection to the entire eye image or to a region of interest around the estimated pupil location, these approaches can be computationally wasteful as the pupil contour frequently occupies very little of the image. We, instead, detect edges along a limited number of rays that extend from a central best guess of the pupil center. These rays can be seen in Figure 3a. This method takes advantage of the high-contrast elliptical profile of the pupil contour present in images taken with infrared illumination using the dark-pupil technique.

For each frame, a location is chosen that represents the best guess of the pupil center in the frame. For the first frame this can be manually determined or taken as the center of the image. For subsequent frames, the location of the pupil center from the previous frame is used. Next, the derivatives Δ along $N = 18$ rays, extending radially away from this starting point, are independently evaluated pixel by pixel until a threshold $\phi = 20$ is exceeded. Given that we are using the dark-pupil technique, only positive derivatives (increasing intensity as the ray extends) are considered. When this threshold is exceeded, a feature point is defined at that location and the processing along the ray is halted. If the ray extends to the border of the image, no feature point is defined. An example set of candidate feature points is shown in Figure 3a.

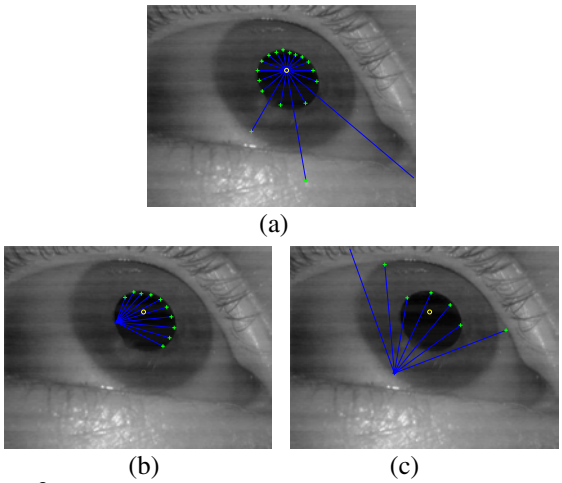

Figure 3: Feature detection. (a) Pupil contour edge candidates are detected along the length of a series of rays extending from a best guess of the pupil center. Pupil contour candidates are marked using crosses. Note that two contour candidates are incorrect - one ray reaches the border and does not generate a candidate. (b) For each pupil contour candidate another set of a rays are generated that create a second set of pupil contour candidates (c) pupil contour candidates not on the pupil contour can lead to additional feature points not on the contour however these are typically not consistent with any single ellipse.

For each of the candidate feature points, the above described feature-detection process is repeated. However, rays are limited to $\gamma = \pm 50$ degrees around the ray that originally generated the feature point. The motivation for limiting the return rays in this way is that if the candidate feature point is indeed on the pupil contour (as shown in Figure 3b), the returning rays will generate additional feature points on the opposite side of the pupil such that they are all consistent with a single ellipse (i.e. the pupil contour). However, if the candidate is not on the pupil (for example see Figure 3c), this process will generate additional candidate feature points that are not necessarily consistent with any single ellipse. Thus, this procedure tends to increase the ratio of the number of feature points on the pupil contour over the number of feature points not on the pupil contour. Given that feature points defined by a large Δ are more likely to be located on the pupil contour (as this is the strongest image contour), the number of rays returned is set to $5\phi/\Delta$. Note that the minimum number of rays is 5 because by definition a feature point is determined by $\Delta >= \phi$.

The two-stage feature-detection process improves the robustness of the method to poor initial guesses for the starting point. This is a problem when an eye movement is

made as the eye can rapidly change positions from frame to frame. This is especially true for images obtained at low frame rates. For example, shown in Figure 4a is such a case. While the initial set of rays only detects three feature points on the pupil contour, the return rays from these three points detect many more points on the contour (see Figure 4b). The combined set of feature points is shown in Figure 4d and the number of points on the contour well exceed those off of the contour. However, the feature points are biased to the side of the pupil contour nearest the initialization point. Although another iteration of the ray process would minimize this bias, the computational burden grows exponentially with each iteration and thus would be an inefficient strategy.

At this point an ellipse could be fitted to the candidate points, however, the bias would induce a significant error into the fit. To eliminate this bias, the above described two-stage feature-detection process is iterated. For each iteration after the first, the average location of all the candidate feature points from the last iteration is taken as the next starting location. The red circle in Figure 4d shows the starting point for the second iteration. The detected feature locations for the second iteration are shown in Figure 4e. Note the absence of a strong bias. Figure 4f shows how the central locations rapidly converge to the actual pupil center. The iteration is halted when the center of the detected feature points changes less than $d = 10$ pixels. When the initial guess is a good estimate of the pupil center, for example during eye fixations which occupy the majority of the frames, only a single iteration is required. When the initial estimate is not good, typically only a few iterations (< 5) are required for convergence. If convergence is not reached within $i = 10$ iterations, as occurs sometimes during a blink when no pupil is visible, the algorithm halts and begins processing the next frame.

4.4. Ellipse fitting

Given a set of candidate feature points, the next step of the algorithm is to find the best fitting ellipse. While other algorithms commonly use least-squares fitting of an ellipse to all the feature points (e.g. see [20]), gross errors made in the feature detection stage can strongly influence the accuracy of the results. Consider the detected feature points shown in Figure 5c and the resulting best-fit ellipse using the least-squares techniques shown in Figure 5d. Notice that a few feature points not on the pupil contour dramatically reduces the quality of the fit to an unacceptable level.

To address this issue, we apply the Random Sample Consensus (RANSAC) paradigm for model fitting [4]. To our knowledge, ours is the first application of RANSAC in the context of eye tracking, however RANSAC is frequently applied to other computer-vision problems (e.g., see [7]). RANSAC is an effective technique for model fitting in the presence of a large but unknown percentage of outliers in a

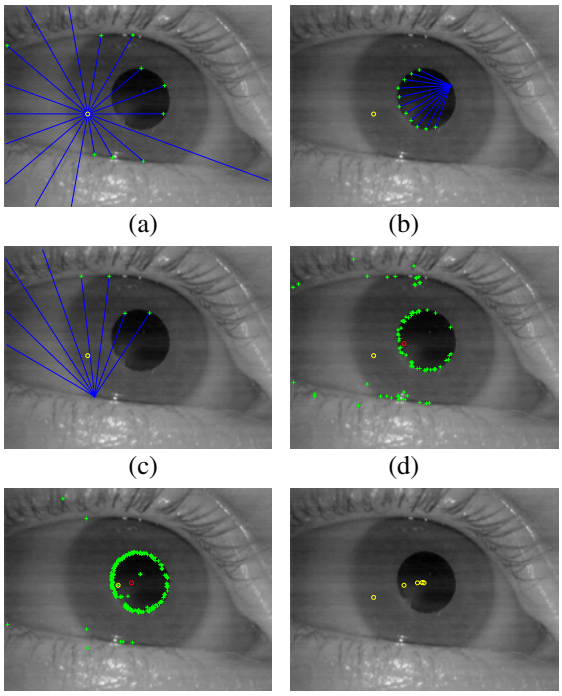

Figure 4: Feature detection. (a) The original start point (yellow circle) shoots rays (blue) to generate candidate pupil points (green crosses). (b&c) The candidate pupil points shoot rays back towards the start point to detect more candidate pupil points. (d) All the candidate pupil points are shown. The average of these locations is shown as a red circle. This location seeds the next iteration. (e) The results of the second iteration. (f) The starting locations from all iterations show a rapid convergence.

measurement sample. An inlier is a sample in the data attributable to the mechanism being modeled whereas an outlier is a sample generated through error and is attributable to another mechanism not under consideration. In our application, inliers are all of those detected feature points that correspond to the pupil contour and outliers are feature points that correspond to other contours, such as that between the eye lid and the eye. Least-squares methods use all available data to fit a model because it is assumed that all of the samples are inliers and that any error is attributable exclusively to measurement error. On the other hand, RANSAC admits the possibility of outliers and only uses a subset of the data to fit the model. In detail, RANSAC is an iterative procedure that selects many small but random subsets of the data, uses each subset to fit a model, and finds the model that has the most agreement with the data set as a whole. The subset of data consistent with this model is the consensus set.

In some cases, our two stage feature-detection process results in very few outliers (e.g., see Figure 5e) while in other cases, outliers are much more prevalent (e.g., see Figure 5f). Therefore it is important that we use the RANSAC paradigm to find the ellipse that best fits the pupil contour. The following procedure is repeated R times. First, five

samples are randomly chosen from the detected feature set given that this is the minimum sample size required to determine all the parameters of an ellipse. Singular Value Decomposition (SVD) on the conic constraint matrix generated with normalized feature-point coordinates [7] is used to find the parameters of the ellipse that perfectly fit these five points.

If the parameters of the ellipse are imaginary, the ellipse center is outside of the image, or the major axis is greater than two times the minor axis, five different points are randomly chosen until this is no longer the case. Then, the number of candidate feature points in the data set that agree with this model (i.e. the inliers) are counted. Inliers are those sample points for which the algebraic distance to the ellipse is less than some threshold t. This threshold is derived from a probabilistic model of the error expected based on the nature of our feature detector. It is assumed that the average error variance of our feature detector is approximately one pixel and that this error is distributed as a Gaussian with zero mean. Thus to obtain a 95% probability that a sample is correctly classified as an inlier, the threshold should be derived from a χ^2 distribution with one degree of freedom [7]. This results in a threshold distance of 1.98 pixels. After R repetitions, the model with the largest consensus set is used. Because it is often computationally infeasible to evaluate all possible feature point combinations, the number of random subsets to try must be determined such that it is assured that at least one of the randomly selected subsets contains only inliers. This can be guaranteed with probability $p = 0.99$, if

$$R = \frac{\log(1-p)}{\log(1-w^5)} \tag{3}$$

where w is the proportion of inliers in the sample. Although w is not known *a priori*, its lower bound is given by the maximum number of inliers found for any model in the iteration and thus R can initially be set very large and lowered using Equation 3 as the iteration proceeds. After the necessary number of iterations, an ellipse is fit to the largest consensus set (e.g., see Figure 5g).

4.5. Model-based optimization

Although the accuracy of the RANSAC fit may be sufficient for many eye tracking applications, the result of ellipse fitting can be improved through a model-based optimization that does not rely on feature detection. To find the parameters a, b, x, y, α of the best fitting ellipse, we minimize

$$-\frac{\int I(a+\delta, b+\delta, \alpha, x, y, \theta)d\theta}{\int I(a-\delta, b-\delta, \alpha, x, y, \theta)d\theta} \tag{4}$$

where $\delta = 1$ and $I(a, b, \alpha, x, y, \theta)$ is the pixel intensity at angle θ on the contour of an ellipse defined by the parameters a,b,x,y and α. The search is initialized with the best-fitting ellipse parameters as determined by the RANSAC fit.

Model-based - Pupil Center	1st	2nd	3rd
Narrow FOV	0.507	6.572	10.362
Wide FOV	0.591	7.527	12.316
Model-based Vector Difference	1st	2nd	3rd
Narrow FOV	0.471	0.981	1.204
Wide FOV	0.515	1.203	1.565

Table 1: Accuracy results of the validation study

4.6. Homographic mapping and calibration

In order to calculate the point of gaze of the user in the scene image, a mapping between locations in the scene image and an eye-position measure (e.g., the vector difference between the pupil center and the corneal reflection) must be determined. The typical procedure in eye-tracking methodology is to measure this relationship through a calibration procedure [17]. During calibration, the user is required to look at a number of scene points for which the positions in the scene image are known. While the user is fixating each scene point $\vec{s} = (x_s, y_s, 1)$, the eye position $\vec{e} = (x_e, y_e, 1)$ is measured (note the homogeneous coordinates). We generate the mapping between the two sets of points using a linear homographic mapping. This mapping H is a 3×3 matrix and has eight degrees of freedom. To determine the entries of H, a constraint matrix is generated using measured point correspondences. Each correspondence generates two constraints and thus four correspondences are sufficient to solve for H up to scale [7]. The null space of the constraint matrix can be determined through SVD, and provides H. Once this mapping is determined the user's point of gaze in the scene for any frame can be established as $\vec{s} = H\vec{e}$. Note that we use a 3×3 grid of calibration points distributed uniformly in the scene image to assure an accurate prediction of eye movements. In this case, there are more constraints than unknowns and SVD produces the mapping H that minimizes the algebraic error distance.

5. Algorithm Validation

An eye-tracking evaluation was conducted in order to validate the performance of the algorithm. Video was recorded from the head-mounted eye tracker described in Section 2 while each of the three authors viewed two movie trailers presented on a laptop computer. Prior to and after the viewing of each trailer, the user placed their head in a chin rest and fixated a series of nine calibration marks on a white board positioned approximately 60 cm away. The evaluation was conducted twice for each user. During the second evaluation, the narrow field of field lens (56^o Field of View (FOV)) used on the scene camera was replaced with a wide field of view lens (111^o FOV, and significant radial distortion) to evaluate the decrease in eye-tracking quality attributable to the non-linear distortion of the lens. The video captured during the evaluation is available for viewing at http://hcvl.hci.iastate.edu/openEyes.

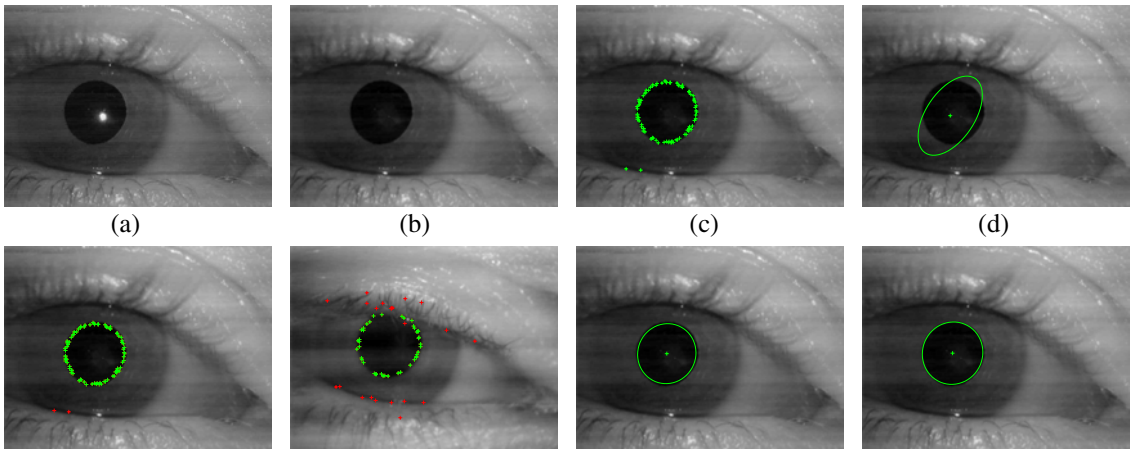

Figure 5: (a) The original image with noise reduction. (b) The image with the corneal reflection removed. (c) Candidate feature points. (d) Ellipse fitted using least-square method. (e) The inliers (green crosses) and outliers (red crosses) differentiated by RANSAC. (f) Another example with many more outliers. (g) Best-fit ellipse using only inliers. (h) Best-fit ellipse using model-based minimization. For illustration purposes, in this figure we use $N = 12$.

Shown in Table 1 are the accuracy estimates derived from the 1st, 2nd and 3rd viewing of the calibration grid separately. Accuracy is measured as the distance between the estimated point of gaze and the actual location of the calibration marks in the scene image averaged over all nine calibration points. The validation results are shown separately for when the pupil center is used and when the vector difference between the pupil center and the corneal reflection center is used to predict the point of gaze. Note that the first viewing of the grid is used to generate the homographic mapping for all predictions.

The first result to notice is that the error is significantly lower when the vector difference between the pupil center and the corneal reflection center is used instead of the pupil center alone. The error tends to increase after each calibration for the point of gaze estimates based on the pupil center alone. This is not surprising and is attributable to slippage of the headgear. The second result to notice is that the pattern of error is similar for both the wide and narrow field of view scene lenses suggesting that the non-linearity has little significance. If desired, this error can be corrected by removing the radial distortion in each frame using standard image-processing techniques.

6. Discussion

We developed a hybrid algorithm for eye tracking that combines feature-based and model-based approaches. Both the corneal reflection and the pupil are located through adaptive feature-based techniques. Then the RANSAC paradigm is applied to maximize the accuracy of ellipse fitting in the presence of gross feature-detection errors. Finally, a model-based approach is applied to further refine the fit. We conducted a validation study which indicates that the algorithm performs well on video obtained from a low-cost head-mounted eye tracker.

We are still in the process of exploring the robustness of our algorithm to variations of its free parameters. For example, the feature detection threshold that we used in our validation study was determined by hand but appears relatively robust, and need not be set differently for different users. However, this threshold will likely need to be tuned for a given eye tracker. In the feature detection process, again we did not explore the effects of manipulating the number of rays and thus it may be that more rays may increase robustness at a minimal cost to runtime performance. We are currently exploring the influence of these parameters on the quality of the point of gaze estimates.

A number of obvious improvements could be made to our current implementation. For example, instead of removing the corneal reflection from the image, which can be quite time consuming, the corneal reflection region could simply be ignored in the other steps of the algorithm. There is also room for additional improvement given that our current algorithm essentially processes images independently (with the exception that the estimate of the pupil center from the previous frame is used as the best guess of the pupil center in the current frame). For example, we are exploring the improvement obtainable through prediction of the pupil center using a Kalman filter. However, the potential benefit of this technique for our hardware is difficult to estimate given the low frame rates and the high velocity of eye movements. We are also exploring automatic calibration. Currently the calibration requires manual input to indicate the location of calibration points in the scene image, which can become tiresome. We expect automatic detection of calibration crosses in the scene image will be possible using image processing techniques.

Our research is aimed at developing reliable eye-tracking algorithms that can run on general-purpose hard-

ware and that can be widely employed in everyday human-computer interfaces. Given that the lack of freely available eye-tracking software has been one obstacle in achieving this goal, we are making the implementation of our algorithm available in an open-source software package under the GNU public license (GPL). This software can be downloaded from our website at http://hcvl.hci.iastate.edu/openEyes. We expect that given the combination of open-source eye-tracking software with low-cost eye-tracking systems built from off-the-shelf components [1, 15, 18], interface designers will be able to explore the potential of eye movements for improving interfaces and that this will lead to an increased role for eye tracking in the next generation of human-computer interfaces.

7. Acknowledgments

We would like thank Jason Babcock who aided in the construction of the head-mounted eye tracker as well as Applied Science Laboratories for supporting this work.

References

[1] J. Babcock and J. Pelz, "Building a lightweight eyetracking headgear," in *ACM Eye tracking research and applications symposium*, San Antonio, TX, USA, March 2004, pp. 109–114.

[2] P. Burt and E. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics*, vol. 2, no. 4, pp. 217–236, 1983.

[3] J. Daugman, "High confidence visual recognition of persons by a test of statistical independence," *IEEE Transactions on Pattern Analysis and Machine Intellegence*, vol. 15, no. 11, pp. 1148–1161, 1993.

[4] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[5] D. Hansen and A. Pece, "Eye tracking in the wild," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 155–181, 2005.

[6] A. Haro, M. Flickner, and I. Essa, "Detecting and tracking eyes by using their physiological properties, dynamics, and appearance," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2000, pp. 163–168.

[7] R. Hartley and A. Zisszerman, *Multiple view geometry in computer vision*. Cambridge, UK: Cambridge University Press, 2000.

[8] R. Jacob, "The use of eye movements in human-computer interaction techniques: what you look at is what you get," *ACM Transactions on Information Systems*, vol. 9, no. 2, pp. 152–169, 1991.

[9] P. Majaranta and K. Raiha, "Twenty years of eye typing: systems and design issues," in *ACM Eye tracking research and applications symposium*, New Orleans, Louisiana, USA, March 2002, pp. 15–22.

[10] C. Morimoto, A. Amir, and M. Flickner, "Detecting eye position and gaze from a single camera and 2 light sources," in *Proceedings. 16th International Conference on Pattern Recognition*, 2002, pp. 314–317.

[11] K. Nishino and S. Nayar, "Eyes for relighting," *ACM SIGGRAPH 2004*, vol. 23, no. 3, pp. 704–711, 2004.

[12] T. Ohno, N. Mukawa, and A. Yoshikawa, "Freegaze: a gaze tracking system for everyday gaze interaction," in *Eye tracking research and applications symposium*, March 2002, pp. 15–22.

[13] D. Parkhurst and E. Niebur, "Variable resolution displays: a theoretical, practical and behavioral evaluation," *Human Factors*, vol. 44, no. 4, pp. 611–29, 2002.

[14] D. Parkhurst and E. Niebur, "A feasibility test for perceptually adaptive level of detail rendering on desktop systems," in *ACM Applied Perception in Graphics and Visualization Symposium*, New York, NY, USA, November 2004, pp. 105–109.

[15] J. Pelz, R. Canosa, J. Babcock, D. Kucharczyk, A. Silver, and D. Konno, "Portable eyetracking: A study of natural eye movements," in *Proceedings of the SPIE, Human Vision and Electronic Imaging*, San Jose, CA, USA, 2000, pp. 566–582.

[16] L. Sibert and R. Jacob, "Evaluation of eye gaze interaction," in *SIGCHI conference on Human factors in computing systems*, The Hague, The Netherlands, April 2000, pp. 281–288.

[17] D. Stampe, "Heuristic filtering and reliable calibration methods for video-based pupil-tracking systems," *Behavior Research Methods, Instruments, and Computers*, vol. 25, no. 2, pp. 137–142, 1993.

[18] D. Winfield, D. Li, J. Babcock, and D. Parkhurst, "Towards an open-hardware open-software toolkit for robust low-cost eye tracking in hci applications," in *Iowa State University Human Computer Interaction Technical Report ISU-HCI-2005-04*, 2005.

[19] L. Young and D. Sheena, "Survey of eye movement recording methods," *Behavior Research Methods and Instrumentation*, vol. 7, pp. 397–429, 1975.

[20] D. Zhu, S. Moore, and T. Raphan, "Robust pupil center detection using a curvature algorithm," *Computer Methods and Programs in Biomedicine*, vol. 59, no. 3, pp. 145–157, 1999.

[21] J. Zhu and J. Yang, "Subpixel eye gaze tracking," in *IEEE Conference on Automatic Face and Gesture Recognition*, May 2002, pp. 124–129.