Starburst: A robust algorithm for video-based eye tracking

Dongheng Li, Derrick J. Parkhurst

Human Computer Interaction Program
Iowa State University, Ames, Iowa, 50011

Abstract

Knowing the user's point of gaze has significant potential to enhance current human-computer interfaces. The primary obstacle of integrating eye movements into 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 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 shows that the technique can reliably estimate eye position with an accuracy of approximately one degree of visual angle even in the presence of significant image noise.

Key words: Human Computer Interaction, Open Source

1 Introduction

The use of eye tracking has significant potential to enhance the quality of everyday human-computer interfaces [1]. 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 [2]. For example, eye-typing applications allow the user to look at keys on a virtual keyboard to type instead of manually pressing keys as with a traditional keyboard [3]. Similarly, systems have been designed that allow users to control the mouse pointer with their eyes in a way that can support, for example, the drawing of pictures [4]. Active interfaces that allow users with movement disabilities to interact with computers may also be helpful

for healthy users by speeding icon selection in graphical user interfaces [5] or object selection in virtual reality [6]. Passive interfaces, on the other hand, monitor the user's eye movements and use this information to adapt some aspect of the display. For example, in video-transmission and virtual-reality applications, gaze-contingent variable-resolution display techniques present a high level of detail at the point of gaze while sacrificing level of detail in the periphery where the absence of detail is not distracting [7,8].

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 [9] 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 [10,11]. Remote video-based eye-tracking techniques also minimize intrusiveness [12,13], 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 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 mostly associated with custom software implementation, 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 [10,11,14], there is a significant potential that eye tracking will be widely incorporated into the next generation of human-computer interfaces.

2 Problem statement

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 in general are not as accurate or flexible as

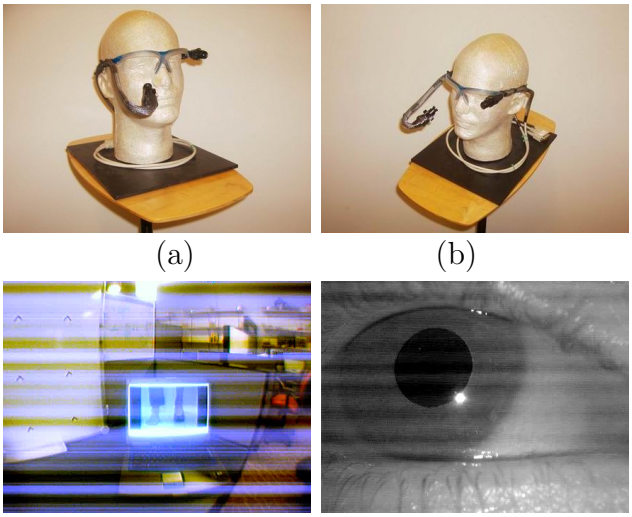


Fig. 1. (a&b) Head-mounted eye tracker (c) Image of a scene obtained by the eye tracker. (d) Image of the user's right eye illuminated with infrared light. Note the clearly defined dark pupil and the specular reflection of the infrared LED. Also note the degree of line noise present in the captured images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

head-mounted systems. In other work [14], we have developed a low-cost head-mounted eye tracker. 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 per frame. In this paper we focus on developing 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 approaches are commonly used in eye tracking, visible and infrared spectrum imaging [15]. 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 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 and testing on infrared spectrum imaging techniques but aim to extend these techniques to visible spectrum imaging as well.

Infrared eye tracking typically utilize either the bright-pupil or dark-pupil technique (however see [13] for the combined use of both bright and dark pupil techniques). The bright-pupil technique illuminates 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. The vector between the pupil center and the corneal reflection center is typically used as the dependent measure rather than the pupil center alone. This is because the vector difference is less sensitive to slippage of the head gear - both the camera and the source move simultaneously. In this paper we focus on algorithm development for dark-pupil techniques however our algorithm could be readily applied to bright-pupil techniques with only slight modification.

3 Related Work

Eye-tracking algorithms typically use two main 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 imagery the dual-threshold technique uses appropriately set intensity thresholds to extract the region corresponding to the pupil and the corneal reflection. The locations of the pupil and corneal reflection are then taken as the geometric center of the identified regions. The intensity gradient can also be used to detect the the pupil contour in infrared spectrum images [16,17] or the limbus in visible spectrum images [18]. An ellipse can then be fit to the feature points using least-squares fitting [19,18,16,17] or the hough transform [20].

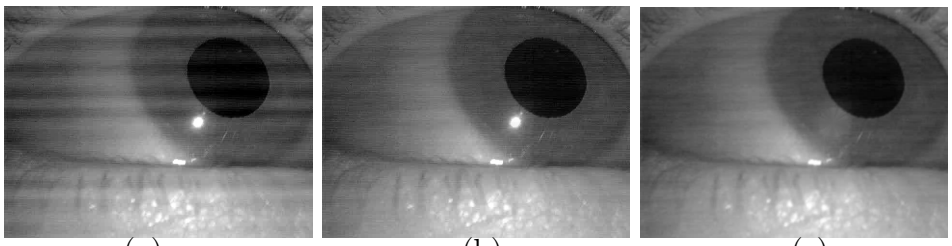
1 **Input**: Eye image, Scene image
2 **Output**: Point of gaze
3 **Procedure:**
4 Detect the corneal reflection
5 Localize the corneal reflection
6 Remove the corneal reflection
7 Iterative detection of candidate feature points
8 Apply RANSAC to find feature point consensus set
9 Determine best-fitting ellipse using consensus set
10 Model-based optimization of ellipse parameters
11 Apply calibration to estimate point of gaze

Fig. 2. Starburst algorithm

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 [21] or ellipse [22] for the limbus or 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 shape of the pupil and its center than a feature-based approach. However, model-based approaches require searching a complex parameter space that can be fraught with local minima and thus cannot be used without a good initial guess of the model parameters. 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 [23] in combination with a model-based approach hold promise for real-time performance (e.g. see [15]).

4 Starburst Algorithm

Presented in this section is a robust eye-tracking algorithm that combines feature-based and model-based approaches to achieve a good trade-off between run-time performance and accuracy for dark-pupil infrared imagery. 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 [24]. The best fitting parameters from this feature-based approach are then used to initialize a local model-based search for the ellipse parameters that maximizes the fit to the image data. The algorithm is shown in Figure 2.



(a) (b) (c)

Fig. 3. (a) The original image. (b) The image with noise reduction. (b) The image with the corneal reflection removed after noise reduction.

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 $5x5$ 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 is

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

where $\bar{I}(i,l)$ is the average line intensity of line l in frame i and $\beta = 0.2$. For $i = 1$, $C(i,l) = \bar{I}(i,l)$. Note that this noise reduction technique is optional when the algorithm is used in combination with an eye tracker capable of capturing images with less noise. The effect of the noise reduction can be seen in Figure 3 (compare a and b).

4.2. Corneal reflection detection, localization and removal

In infrared spectrum eye tracking using the dark-pupil technique, 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 Parameter Analysis 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 any other specular reflections tend to be quite small and located off the cornea (e.g., near the corner of the image

where the eye lids meet). The ratio between the area of the largest candidate region 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.

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 symmetric 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 symmetric bivariate Gaussian (i.e. a radius of one standard deviation), we can take the full extent of the corneal reflection as $2.5r$ to capture 98% of the corneal reflection profile. We find r through a Nelder-Mead Simplex search that minimizes

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

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 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 and on average requires only 2.3 percent of the algorithm's runtime.

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 3 (compare b and c).

4.3. Pupil contour detection

We have developed a novel feature-based method to detect the pupil contour. 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

```
1  Input: Eye image with corneal reflection removed,
2         Best guess of pupil center
3  Output: Set of feature points
4  Procedure:
5  Iterate
6    Stage 1:
7    Follow rays extending from the starting point
8    Calculate intensity derivative at each point
9    If derivative > threshold then
10     Place feature point
11     Halt march along ray
12   Stage 2:
13   For each feature point detected in Stage 1
14     March along rays returning towards the starting point
15     Calculate intensity derivative at each point
16     If derivative > threshold then
17       Place feature point
18       Halt march along ray
19   Starting point = geometric center of feature points
20 Until starting point converges
```

Fig. 4. Feature-detection algorithm

approaches can be computationally wasteful as the pupil contour frequently occupies very little of the image and not all the pupil contour points are necessarily needed for accurate estimation of the pupil contour. 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 5a. For robustness to inaccuracy of the starting point, edges are also detected along a limited number of rays extending from the initial set of detected features returning in the direction of the starting point. These returning rays can be seen in Figure 5b&c. This two-stage detection method takes advantage of the elliptical profile of the pupil contour to preferentially detect features on the pupil contour. The feature-detection algorithm is shown in Figure 4.

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 rays, extending radially away from this starting point, are independently evaluated pixel by pixel until a threshold ϕ 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 5a.

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 5b), 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 (e.g., see Figure 5c), this process will generate additional candidate feature points that are not necessarily consistent with any single given ellipse. Thus, this procedure tends to increase 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 returning rays is variable and set to $5\Delta/\phi$. 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 5d is such a case. While the initial set of rays only detects two feature points on the pupil contour, the return rays from these two points detect many more points on the contour (see Figure 5e). The combined set of feature points is shown in Figure 5g 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 5g shows the starting point for the second iteration. The detected feature locations for the second iteration are shown in Figure 5h. Note the absence of a strong bias. Figure 5i 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 one iteration is required. When the initial estimate is not good, typically only a few iterations are required for convergence. The histogram of the iteration count is shown in Figure 6a for

the videos recorded as described in Section 5. Note that 93% of the iteration counts are less than or equal to 5. 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. The tested videos contained over $50,000$ frames and only 1.7 percent of the frames are of this kind. On average, the feature-detection process requires 27 percent of the algorithm's runtime.

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, gross errors made in the feature-detection stage can strongly influence the accuracy of the results. Consider the detected feature points shown in Figure 5j and the resulting best-fit ellipse using the least-squares techniques shown in Figure 5k. 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 [24]. 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 [25]). RANSAC is an effective technique for model fitting in the presence of a large but unknown percentage of outliers in a 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.

In most cases, our two stage feature-detection process results in very few outliers (e.g., see Figure 5l) while in other cases, outliers are much more prevalent (e.g., see Figure 5m). The distribution of outlier percentages for our test videos is shown in Figure 6b. On average, 17 percent of the feature points are outliers. This relatively high amount of outliers is due to the fact that we are using a low-cost eye tracker constructed from off-the-shelf parts that introduces sig-

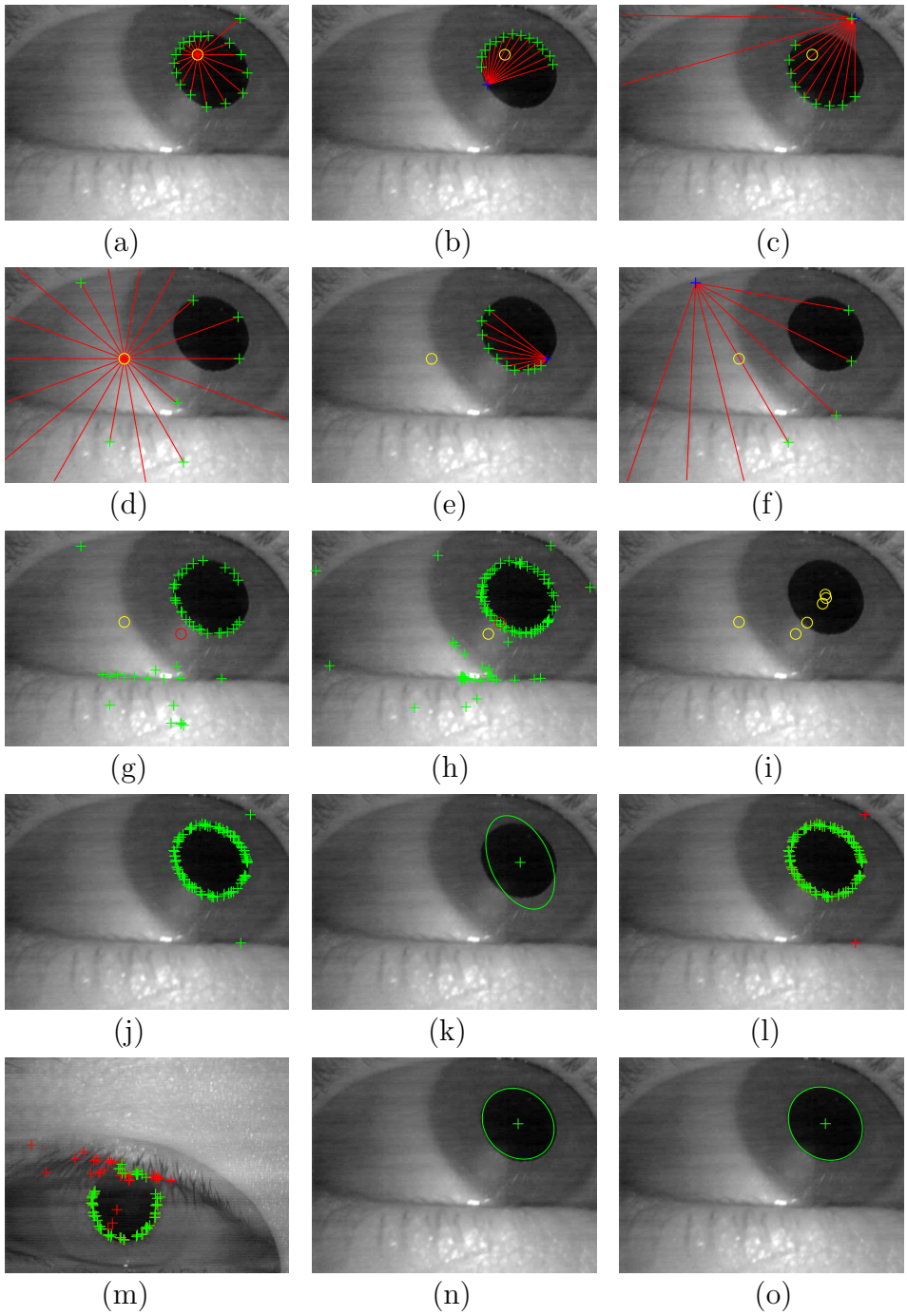


Fig. 5. (a) Pupil edge candidates (crosses) are detected along rays extending from a starting point (yellow circle) inside of the pupil contour. (b) For each candidate, a set of returning rays are generated that further enlarges the set of candidates. (c) Returning rays from a candidate not on the pupil contour (d-f) As in (a-c) with the starting point away from the pupil (g) All candidates and their average location (red circle) are shown. This location seeds the next iteration. (h) Second iteration (i) Convergence (j) Example set of feature points with only 2 outliers. (k) Poorly fit ellipse resulting from least-squares approach. (l) Inliers (green) and outliers (red) differentiated by RANSAC. (m) An example with more outliers. (n) Best-fitting ellipse using only inliers. (o) Best-fitting ellipse using model-based optimization.

nificant image noise into the videos. Given the presence of these outliers, 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 [25] is then used to find the parameters of the ellipse that perfectly fits these five points. The parameters of the ellipse must be real, the ellipse center must be inside of the image, and the major axis must be less than two times the minor axis. Otherwise, five more points are randomly chosen and an new ellipse fit, until these constraints are met. Then, the number of candidate feature points in the data set that agree with this model (i.e. the inliers) are counted. This set is called the consensus set. After the necessary number of iterations, an ellipse is fit to the largest consensus set (e.g., see Figure 5n).

Inliers are those sample points for which the algebraic distance to the ellipse is less than some threshold. 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 [25]. This results in a threshold distance of 1.98 pixels.

Because it is often computationally infeasible to evaluate all possible feature point combinations, the number of random subsets to try must be determined in a way that assures 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 size of the largest consensus set found. Thus R can initially be set very large and then set lower based on Equation 3 as the iteration proceeds. The number of necessary iterations can be further reduced each time that a new largest consensus set is detected, by iteratively re-estimating the model using all the members of the consensus set until the total number of inliers remains constant. The histogram of RANSAC iterations for the tested videos is shown in Figure 6c. Note that the median number of iterations is only 8 and the RANSAC model fitting on average utilizes 5.5 percent of the algorithm's runtime.

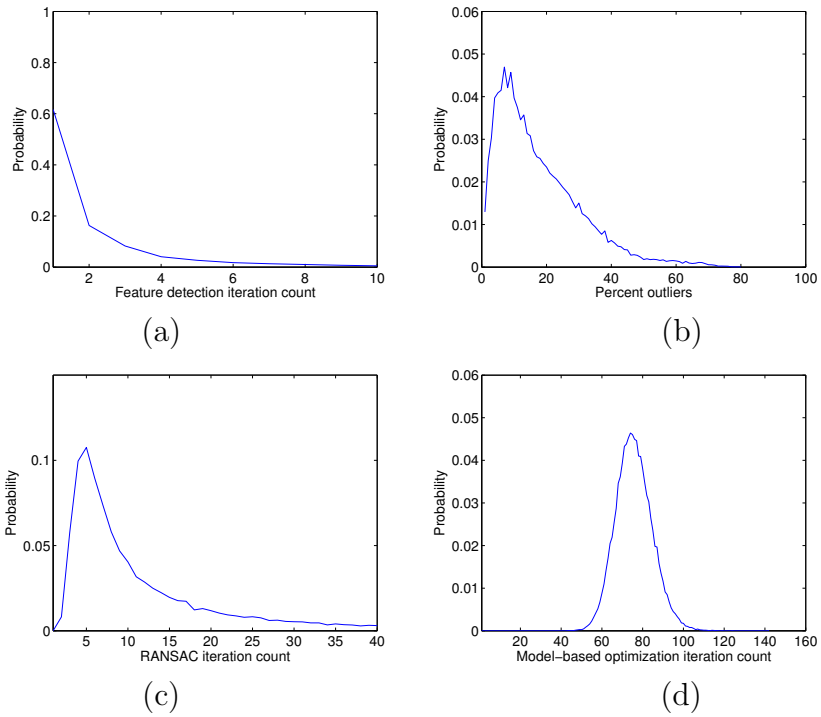


Fig. 6. (a) The histogram of iterations of pupil feature detection. (b) The percentage of outliers in processed videos. (c) The histogram of RANSAC iterations. (d) The histogram of iterations of model-based optimization

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

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

using a Nelder-Mead Simplex search 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 RANSAC. An example of model-based optimization can be seen in Figure 5o. The probability distribution of optimization iterations is shown in Figure 6d. The mean number of iterations is 74 and, on average, model-based optimization requires 17 percent of the algorithm's runtime.

4.6. Calibration

In order to calculate the point of gaze in the scene image, a mapping must be constructed between eye-position coordinates and scene-image coordinates. Either the pupil center or the vector difference between the pupil center and the corneal reflection center can be used. However, as will be shown in the Algorithm Validation Section, the vector difference leads to superior performance because it reduces sensitivity to the slippage of the headgear. The mapping can be initialized by relating known eye positions to known scene locations. The typical procedure in eye-tracking methodology is to measure this relationship through a calibration procedure [26]. During calibration, the user is required to look at a 3×3 grid of scene points for which the positions in the scene image are known. While the user is fixating each scene point $\vec{si} = (x_{si}, y_{si})$, the eye position $\vec{e_i} = (x_{ei}, y_{ei})$ is measured.

The particular mapping used by different eye-tracker manufacturers and different research group varies widely. Therefore, we examined the optimal mapping for our head-mounted system by examining the accuracy of the mappings derived from the first nine-point calibration in our validation study (see Algorithm Validation Section). The mapping that we examined was a first-order linear mapping. For each correspondence between $\vec{s_i}$ and $\vec{e_i}$, two equations are generated that constrain the mapping:

$$x_{si} = a_{x0} + a_{x1}x_{ei} + a_{x2}y_{ei} \tag{5}$$

$$y_{si} = a_{y0} + a_{y1}x_{ei} + a_{y2}y_{ei} \tag{6}$$

where a_{xi} and a_{yi} are undetermined coefficients of the linear mapping. This linear formulation results in six coefficients that need to be determined. Given the nine point correspondences from the calibration and the resulting 18 constraint equations, the coefficients can be solved for in the least-squares sense using SVD [25]. Nonlinear mappings were also be considered using this framework including second-order and third-order polynomial mappings. The second-order mapping included all six additional higher order terms. Because a full third-order mapping would require 20 parameters and the number of unknowns would exceed the number of constraints, cross terms were not considered to allow for a solution.

Another non-linear method that we considered was to use a homographic mapping. We generate the mapping H, 3×3 matrix that has eight degrees of freedom, between the scene point $\vec{s} = (x_s, y_s, 1)$ and the pupil-CR vector $\vec{e} = (x_e, y_e, 1)$. 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 [25]. Finally the null space of the constraint matrix can be determined

Calibration Method	Error (degrees)
Linear	0.77
2nd-order polynomial	0.57
3rd-order polynomial	0.64
Homographic	0.58

Table 1

through SVD to provide H. SVD produces the mapping H that minimizes the algebraic error. Once the mapping is determined, the user's point of gaze in the scene for any frame can be established as $\vec{s} = H\vec{e}$.

The average errors obtained are shown in Table1. All mappings provide reasonable accuracy however, the second-order mapping and homographic mappings result in the best performance. The lack of cross-terms hurts the third-order mapping. However, we expect that the third-order mapping would result in an accuracy comparable to the second-order mapping, if sufficient correspondences were available to include the cross-terms. Overall, we conclude that the choice of mapping makes little difference but that a non-linear model should be preferred. However, a more comprehensive investigation that examines the ability of these mappings to extrapolate outside of the nine-point calibration grid would be valuable, but is outside the scope of this paper.

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 three users (the two authors and a research assistant) 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, with eye movement predictions, is available for viewing at http://hcvl.hci.iastate.edu.

Shown in Table 2 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. Note that the first viewing of the grid is used to generate the mapping for all predictions. Five sets of results are shown in Table 2, each specifying the error in degrees of visual angle for each calibration and both types of lens. The first set of result shows the poor accuracy of the dual-threshold algorithm. Most commercial eye trackers use this technique. This poor performance of this algorithm with our noisy low-cost hardware originally motivated the development of the Starburst algorithm. The second set of results were obtained with the dual-threshold algorithm run on the noise reduced images generated by the algorithm in Section 4.1. The results are improved but still significantly affected by the residual noise. The third set of results were obtained with the Starburst algorithm and show an average accuracy of one degree of visual angle, a ten-fold improvement in performance compared to the simple dual-threshold algorithm. Notably, the error of the wide FOV camera is slightly higher than that of the narrow FOV camera due to presence of radial distortion. If additional accuracy is desired when using a wide FOV camera, the distortion can be easily be digitally removed once the camera has been calibrated.

The fourth set of results is generated by the Starburst algorithm without the final model-based optimization step. Comparing these results to the full algorithm, there is an approximately 7 percent improvement in accuracy attributable to the model-based optimization. While the model-based optimization takes approximately 17 percent of the algorithm's total runtime, our measures of accuracy only quantify the improvement attributable to the coordinates of the ellipse center. Based on visual inspection, the optimization also improves the other parameters of the ellipse, which may be useful in other applications.

The final set of results show the performance of the Starburst algorithm when only the pupil center is used as a measure of eye position. Performance on the initial calibration is actually slightly superior than when the vector difference between the corneal reflection center and the pupil center is used. This effect can be attributed to the greater variability of the vector difference. Notably, however, the accuracy is dramatically reduced for subsequent calibrations. The error tends to increase after each calibration and is likely due to slippage of the eye tracker on the participant's head.

Dual-Threshold	1st	2nd	3rd
Narrow FOV	10.059	11.196	9.605
Wide FOV	7.670	7.901	9.542
DT (noise reduced)	1st	2nd	3rd
Narrow FOV	1.838	3.062	4.221
Wide FOV	3.272	3.083	4.039
Starburst	1st	2nd	3rd
Narrow FOV	0.596	1.025	1.049
Wide FOV	0.678	1.113	1.366
RANSAC only	1st	2nd	3rd
Narrow FOV	0.616	1.006	1.151
Wide FOV	0.970	1.215	1.299
Pupil Center only	1st	2nd	3rd
Narrow FOV	0.349	6.520	10.264
Wide FOV	0.542	7.376	12.357

Table 2

6 Parameter Analysis

To determine the robustness of the Starburst algorithm, a number of analyses were conducted. In the algorithm, the pupil center of previous frame is used as the best guess for the current frame, which can be a poor guess during eye movements. The goal of the first analysis was to examine the sensitivity of the algorithm to the quality of the initial best guess of the pupil center used to seed the feature-detection process. This analysis included the nine images from the first calibration in each of the six videos in which a user is fixating on a calibration point. The ground truth ellipse parameters in each eye image were determined by using a starting point set at approximately the center of the pupil as determined by observation. Then, 36 starting points equi-distant from the true pupil center were selected in 10 degree angular steps and used to initialize the algorithm. Performance at a range of distances was examined and is shown in Figure 7a. Performance is given as the probability of error, where the result is classified as an error if the distance between calculated pupil center and the ground truth was greater than 4 pixels, or the major or minor axis radius was different than 4 pixels from the ground truth. Even with such stringent criteria, the average error stays under 15 percent for starting

points misplaced by one-half of the image width. With the 30 hertz frame-rate camera that we use, the distance that the eye moves between frames is typically less than 100 pixels from the pupil center and thus error rates due to poor starting points should be less than 2 percent.

The robustness of our algorithm to its free parameters was also examined. The first free parameter examined was the threshold Δ controlling feature detection. All 6 videos were processed with a variety of thresholds. The accuracy of the algorithm in predicting the point of gaze in the scene for the first calibration grid was examined and is shown in Figure 7b separately for each video. With a threshold in the range of 15 to 20, the error is approximately one degree for all videos. The degree to which the algorithm requires tuning of the threshold depends on the quality of the recorded video and the degree of image contrast available. Image quality will vary with illumination intensity and the reflectance of the user's iris, which can vary from individual to individual. This variability can be seen in the different traces in Figure 7b.

The effects of manipulating the number of rays and the step size (in pixels) along the rays used in calculating the derivative was also examined. Figure 7c shows the eye tracking error as a function of the number of rays used in the first stage of the feature-detection process. It can be seen that beyond 8 rays, the benefits in terms of accuracy will be probably be outweighed by the additional computational cost. Figure 7d shows the eye tracking error as a function of the step size along the rays. For either small or large step sizes, performance is poor. For very small step sizes, it is difficult to exceed the threshold for feature detection and for large step sizes, accuracy in localizing the feature is sacrificed. A intermediate step size of 8 pixels results in the best trade-off between speed and accuracy of computation.

7 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 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 ellipse fit. We conducted a validation study which indicates that the algorithm out performs standard algorithms on video obtained from a low-cost head-mounted eye tracker. The algorithm is also robust to variation in its parameters, but this robustness is directly linked to the quality of the video obtained from the eye tracker.

A number of improvements could be made to our current implementation.

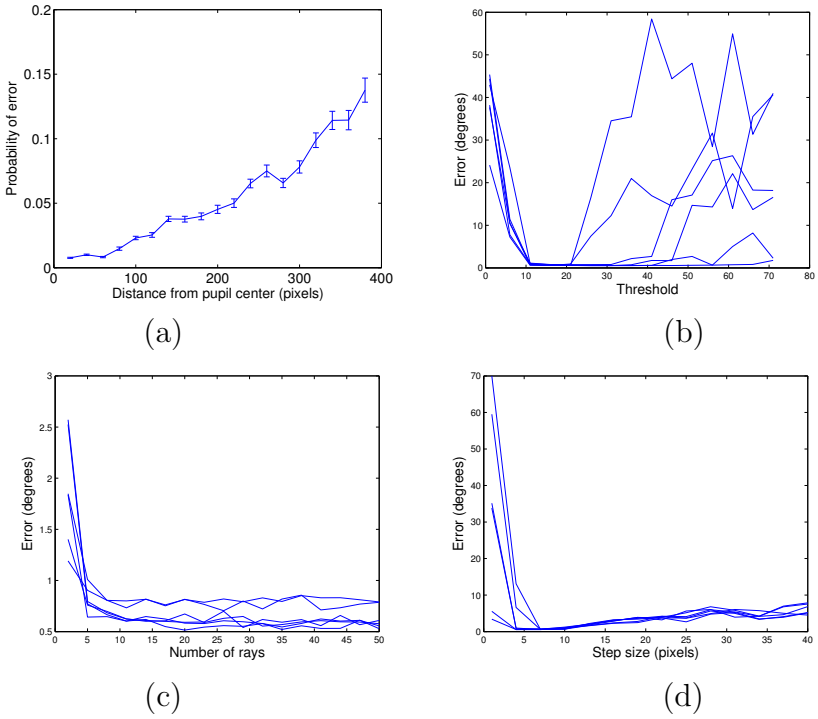


Fig. 7. The parameters analysis. (a) The probability of error over the distance between starting point and true pupil center. (b) The visual error when changing the threshold of pupil feature detection. (c) The visual error when changing the step size of pupil feature detection. (d) The visual error when changing the number of rays to detect features.

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.

One issue with using low-cost low frame rate eye-tracking hardware is that the accuracy of point of gaze estimates during eye movements can be quite poor. This loss of accuracy is due to the motion blur induced by the long CCD integration times associated with the low frame rates. Fortunately, eye movements are very rapid lasting on the order of 10 milliseconds while fixations

are much longer (hundreds of milliseconds). Thus only a small percentage of the captured images show the eye in motion and for many of these frames, the motion blur is small enough that an accuracy estimate of the point of gaze can still be obtained. The use of readily available, yet more expensive, cameras capable of flexible integration times, higher sensitivity and higher frame rates would eliminate this problem.

Our research is aimed at developing reliable eye-tracking algorithms that can run on general-purpose hardware 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. This implementation is written in Matlab and operates at approximately one frame per second. We are actively working on releasing a C++ implementation that runs in real-time needed for human-computer interaction applications. We expect that given the combination of open-source eye-tracking software with low-cost eye-tracking systems built from off-the-shelf components [11,10,14], 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.

8 Acknowledgments

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

References

[1] A. Duchowski, A breadth-first survey of eye-tracking applications, Behavior Research Methods Instruments and Computers 34 (4) (2002) 455–470.

[2] 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 9 (2) (1991) 152–169.

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

[4] A. J. Hornof, A. Cavender, R. Hoselton, Eyedraw: A system for drawing pictures with eye movements, in: ACM SIGACCESS Conference on Computers and Accessibility, Atlanta, Georgia, 2004, pp. 86–93.

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

[6] V. Tanriverdi, R. Jacob, Interacting with eye movements in virtual environments, in: Proceedings of the SIGCHI conference on Human factors in computing systems, 2000, pp. 265–272.

[7] D. Parkhurst, E. Niebur, Variable resolution displays: a theoretical, practical and behavioral evaluation, Human Factors 44 (4) (2002) 611–29.

[8] D. Parkhurst, 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, 2004, pp. 105–109.

[9] L. Young, D. Sheena, Survey of eye movement recording methods, Behavior Research Methods and Instrumentation 7 (1975) 397–429.

[10] J. Pelz, R. Canosa, J. Babcock, D. Kucharczyk, A. Silver, 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.

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

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

[13] C. Morimoto, A. Amir, 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.

[14] D. Winfield, D. Li, 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.

[15] D. Hansen, A. Pece, Eye tracking in the wild, Computer Vision and Image Understanding 98 (1) (2005) 155–181.

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

[17] X. Brolly, J. Mulligan, Implicit calibration of a remote gaze tracker, in: IEEE Conference on CVPR Workshop on Object Tracking Beyond the Visible Spectrum, 2004.

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

[19] D. Zhu, S. Moore, T. Raphan, Robust pupil center detection using a curvature algorithm, Computer Methods and Programs in Biomedicine 59 (3) (1999) 145–157.

[20] Y. Matsumoto, A. Zelinsky, An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement, in: IEEE Conference on Automatic Face and Gesture Recognition, Grenoble, France, 2000, pp. 499–504.

[21] J. Daugman, High confidence visual recognition of persons by a test of statistical independence, IEEE Transactions on Pattern Analysis and Machine Intellegence 15 (11) (1993) 1148–1161.

[22] K. Nishino, S. Nayar, Eyes for relighting, ACM SIGGRAPH 2004 23 (3) (2004) 704–711.

[23] P. Burt, E. Adelson, A multiresolution spline with application to image mosaics, ACM Transactions on Graphics 2 (4) (1983) 217–236.

[24] M. Fischler, R. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[25] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge University Press, Cambridge, UK, 2000.

[26] D. Stampe, Heuristic filtering and reliable calibration methods for video-based pupil-tracking systems, Behavior Research Methods, Instruments, and Computers 25 (2) (1993) 137–142.