Behav Res
DOI 10.3758/s13428-014-0465-z

ExpertEyes: Open-source, high-definition eyetracking

Francisco J. Parada · Dean Wyatte · Chen Yu · Ruj Akavipat · Brandi Emerick · Thomas Busey

Abstract ExpertEyes is a low-cost, open-source package of hardware and software that is designed to provide portable high-definition eyetracking. The project involves several technological innovations, including portability, high-definition video recording, and multiplatform software support. It was designed for challenging recording environments, and all processing is done offline to allow for optimization of parameter estimation. The pupil and corneal reflection are estimated using a novel forward eye model that simultaneously fits both the pupil and the corneal reflection with full ellipses, addressing a common situation in which the corneal reflection sits at the edge of the pupil and therefore breaks the contour of the ellipse. The accuracy and precision of the system are comparable to or better than what is available in commercial eyetracking systems, with a typical accuracy of less than 0.4° and best accuracy below 0.3°, and with a typical precision (*SD* method) around 0.3° and best precision below 0.2°. Part of the success of the system comes from a high-resolution eye image. The high image quality results from uncasing common digital camcorders and recording directly to SD cards, which avoids the limitations of the analog NTSC format. The software is freely downloadable, and complete hardware plans are available, along with sources for custom parts.

F. J. Parada · C. Yu · B. Emerick · T. Busey (✉)
Department of Psychology, Indiana University, 1101 East 10th Street, Bloomington, IN 47405, USA
e-mail: busey@indiana.edu

D. Wyatte
Department of Psychology, University of Colorado, Boulder, CO, USA

R. Akavipat
Department of Computer Engineering, Mahidol University, Bangkok, Thailand

Keywords Eyetracking · Methodology · Open-source software · Open-source hardware · High-definition

ExpertEyes is an eyetracking package that was developed to perform eyetracking research in remote locations with very demanding testing conditions. For example, our research group collects eye-gaze data from participants in their workplaces and at conferences around the world, and we are therefore often limited by environmental factors such as variable ambient lighting. Although portable commercial systems capable of remote recording are readily available from a variety of manufacturers, they typically perform gaze estimation in real time, a task that is vulnerable to environmental factors that can affect recording accuracy, often without the awareness of the experimenter. Therefore, encoding errors or even data loss can be real concerns when working with real-time eyetracking systems outside of optimal lab conditions. In developing ExpertEyes, we chose an offline approach, in which we record raw video streams from eye and head cameras to a storage device and perform more complex processing of data to deal with demanding testing conditions.

The inspiration for our early efforts came from work by Babcock and Pelz (2004) and Li, Babcock, and Parkhurst (2006), who both presented hardware plans for a portable, head-mounted eyetracker. We adapted many of their hardware ideas and took advantage of improvements in camera technology over recent years to develop our own portable eyetracking system (see Fig. 1).

However, no software was available that met our needs for data analysis, and we therefore developed the ExpertEyes software to process video recorded from the system. At the core of the ExpertEyes software is a novel gaze-estimation algorithm that uses a forward model to perform parameter fitting and tracking of a participant's pupil and corneal reflection (CR) across video frames. Because gaze estimation is

Published online: 17 June 2014

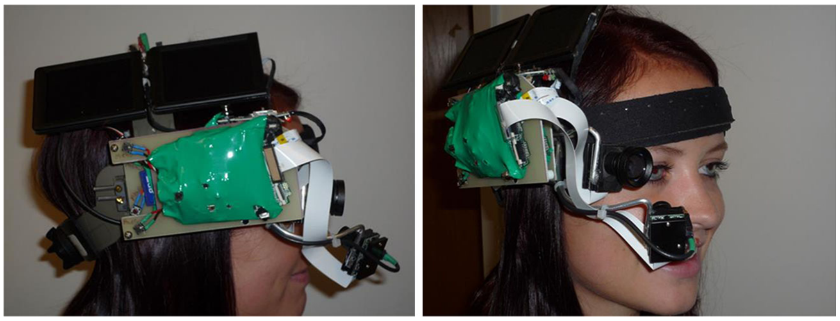


Fig. 1 Two views of the ExpertEyes head-mounted eyetracker: The headgear consists of two uncased camcorders with lens elements that are extended to allow the image sensor to be positioned to record the scene view and the eye view

done offline, we have ample opportunity to refine the model, even for different regions of visual space, resulting in a highly accurate estimate of gaze tailored to each recording session. The software also supports the entire eyetracking data-processing workflow, including data cleaning, trial segmentation, projection of eye gaze into computer monitor coordinate spaces, and export for fixation finding and complex data analysis in MATLAB or other analysis packages.

The ExpertEyes system has five main advantages:

1. High spatial resolution and accuracy: Our system reports comparable and sometimes superior precision in comparison with commercial systems. Additionally, our offline gaze estimation provides users the opportunity to improve the accuracy of each eye model.
2. Higher temporal resolution than NTSC systems: Our HD cameras record at 60 Hz, and the system is hardware independent, enabling it to take advantage of future innovations such as cameras that can record at 120 Hz or even 240 Hz.
3. Portability: The ExpertEyes system can collect data in controlled laboratory settings as well as under natural viewing conditions.
4. Transparency: Many commercial systems rely on a black box of data processing, because they do not publish their algorithms. In contrast, our system allows users to see all the details in the data-processing pipeline and provides users with the option to make sensible decisions on the basis of data quality. The underlying algorithms are also viewable in the source code.
5. Low cost: The suggested system is relatively simple and cost-effective to build, with final costs being less than US$500. Moreover, the ExpertEyes software is platform independent, available for free downloads, and is completely open source.

We envision two potential uses for the ExpertEyes package: (1) Users who want to perform offline analysis of raw eyetracking data, which is often provided by many commercial and open-source eyetracking systems and (2) Users who are willing to build relatively simple head-mounted eyetrackers, such as our own or similar (Babcock & Pelz, 2004), to collect such data. This article covers both scenarios. We begin by describing the complete hardware plans of our particular eyetracking system for users who want to build their own. We then describe the ExpertEyes software, first focusing on the gaze-estimation algorithm, followed by a brief overview describing the various modules and capabilities that the software provides. Step-by-step video and written tutorials are available online at the project's website (http://code.google.com/p/experteyes).

Hardware

The hardware consists of two HD cameras that record an image of the eye relative to the head, and an image of the scene, which positions the head relative to the world. After calibration, we know the position of the eye relative to the world. Figure 1 illustrates two views of the HD eyetracker. (A video description of the hardware can be found on the project's website.)

At the heart of our HD eyetracker are two uncased digital camcorders. (Complete instructional videos and discussion can be found on the project's website.) The advantage of this approach is that it is lightweight, records high-resolution images, and is battery-powered. There are several technical challenges involved with preparing the cameras, which are discussed in the website wiki (https://code.google.com/p/experteyes/). In essence, it is necessary to extend the imaging sensor using a small cable, and make a custom lens holder to hold a standard 12-mm, .5-mm pitch lens.

The imaging sensors for both cameras are mounted on a small daughter card and communicate over a 30-pin, .5-mm pitch cable. We found that we could extend this cable for 6 to 8 in. to reach the scene and eye cameras, respectively. The disadvantage to this extension is that the cables themselves are somewhat exposed, and care must be taken when handling the eyetracker.

The eye is illuminated with a standard infrared LED, which not only provides a corneal reflection that can be tracked using the ExpertEyes software, but also provides enough light for clean separation between the pupil and iris. A 9-volt battery and a custom circuit board provide power and allow for changes in illumination as required by the setting. (All plans and parts lists are available on the project's website.)

The two uncased cameras are positioned on a fiberglass mount, on headgear taken from a hard hat. The LCD monitors from the uncased cameras are mounted above the cameras to provide previews of the images from both cameras. The cameras are powered by the original battery from each camera, which is mounted on the other side of the head, to provide balance. Additional weight is often required, although once positioned on the head the weight is not usually noticeable by subjects.

The cameras are turned on via a button mounted on the fiberglass mount, and a second button allows access to the advanced menus. Recording is synchronized using an infrared remote supplied by the camera manufacturer that reliably activates both cameras. Synchronization is verified by using a standard camera flash, which will be recorded in 1 frame in both cameras. ExpertEyes gives the option to synchronize both streams of data to any frame; the flash-marked frame can be used for resynchronization, if necessary.

After they are positioned on the head, the cameras must be oriented to capture the eye in various positions, and the scene camera must be aimed at the scene of interest (usually a computer monitor). Both cameras must be focused, and the infrared LED must be adjusted to point directly at the eye. After both cameras are adjusted, recording is initiated, and both cameras are flashed, the subject proceeds with a brief calibration sequence (usually looking at fixed locations presented as dots on a monitor) and then performs approximately 20–30 min of the experiment. We typically record at 1280 × 720 pixels at 60 Hz, although 1920 × 1080 at 30 Hz is also available. At the end of recording we stop both cameras, power them down, and remove the SD cards to transfer the files to ExpertEyes.

Accuracy and precision evaluation

One measure of accuracy in eyetracker measures is defined as the average difference between the estimated gaze location and the real gaze location at a particular point (e.g., each calibration point). Ideally, this difference should be close to zero. In contrast, precision can be thought of as "spatial resolution." This is how reliably a measure of eye position can be reproduced. Ideally, when an eye fixates twice in the same point, an eyetracker should report both locations as being exactly the same. Nevertheless, even when measured under ideal conditions, eyetracking systems report an inevitable offset that affects accuracy and precision. In standard reports, accuracy and precision are measured in degrees of visual angle. Eyetracking systems generally report an offset range that falls between 0.5° and 1° for accuracy and between 0.01° and 1° for precision (Duchowski, 2007). Although most companies report in their manuals accuracy gaze vectors of 0.5°, some work that has been done comparing different eyetracker systems differs from these reports. Results show that, on average, accuracy estimations during "real laboratory" circumstances range from 1.3° to 1.8°, depending on the system (Nevalainen & Sajaniemi, 2004). Differences between reported and actual accuracy happen for a couple of reasons. Eyetracker companies report the system's accuracy measured under ideal conditions. In general, these ideal conditions include subjects with normal vision and with no need for corrective glasses or contact lenses. Moreover, accuracy measurements are usually done right after calibration (Holmqvist, 2011).

Figure 2 illustrates four example calibration screens, illustrating excellent fits (top row), a typical fit (lower left), and a relatively poor fit (lower right). There are several sources of error in an eyetracker system, including errors in estimation of the pupil, corneal reflection, or monitor corners, slippage of the eyetracker on the head, and errors introduced by changes in viewing distance. This last error occurs because the calibration is typically only good for one viewing distance, and any change in depth results in errors that are proportional to the distance between the center of the tracked eye and the center of the scene camera. Positional shifts of the eyetracker on the head can be partially compensated via drift correction, using known gaze points throughout the task (i.e., between trials in which the subject is asked to fixate a known location). Changes in depth are difficult to handle and should be avoided because the calibration is only good for one depth plane, and the accuracy error due to changes in depth is related to the offset between the scene camera and the center of the eye.

For the purposes of comparison with the typical report of accuracy and precision, we chose to estimate the eyetracker accuracy and precision from a group of 26 subjects from an eyetracking study (Kitchell et al., in preparation), which included scene undistortion and monitor corner estimation. This data set was collected by an undergraduate honors student (Lindsey Kitchell) as part of her senior honors thesis, and all data were collected and processed by her. She graciously provided the calibration data from each participant to allow a summary of the accuracy and precision of the high-definition eyetrackers that are part of the suggested hardware of the ExpertEyes system. The gaze data will be published as part of a separate article.

We computed the angular difference between the estimated gaze location and the to-be-tracked location on the monitor (essentially, between dark and light grey dots in Fig. 2, consult online version for color figures) in degrees of visual angle and computed the average accuracy. For accuracy, we calculated the average distance between each calibration gaze point and the ground truth spatial location for that point (i.e., the difference between where the subject was asked to look and where we estimated he or she actually looked). The distributions are plotted in Fig. 3.

There are two methods to compute the precision of the eyetracker, and these serve slightly different purposes. In

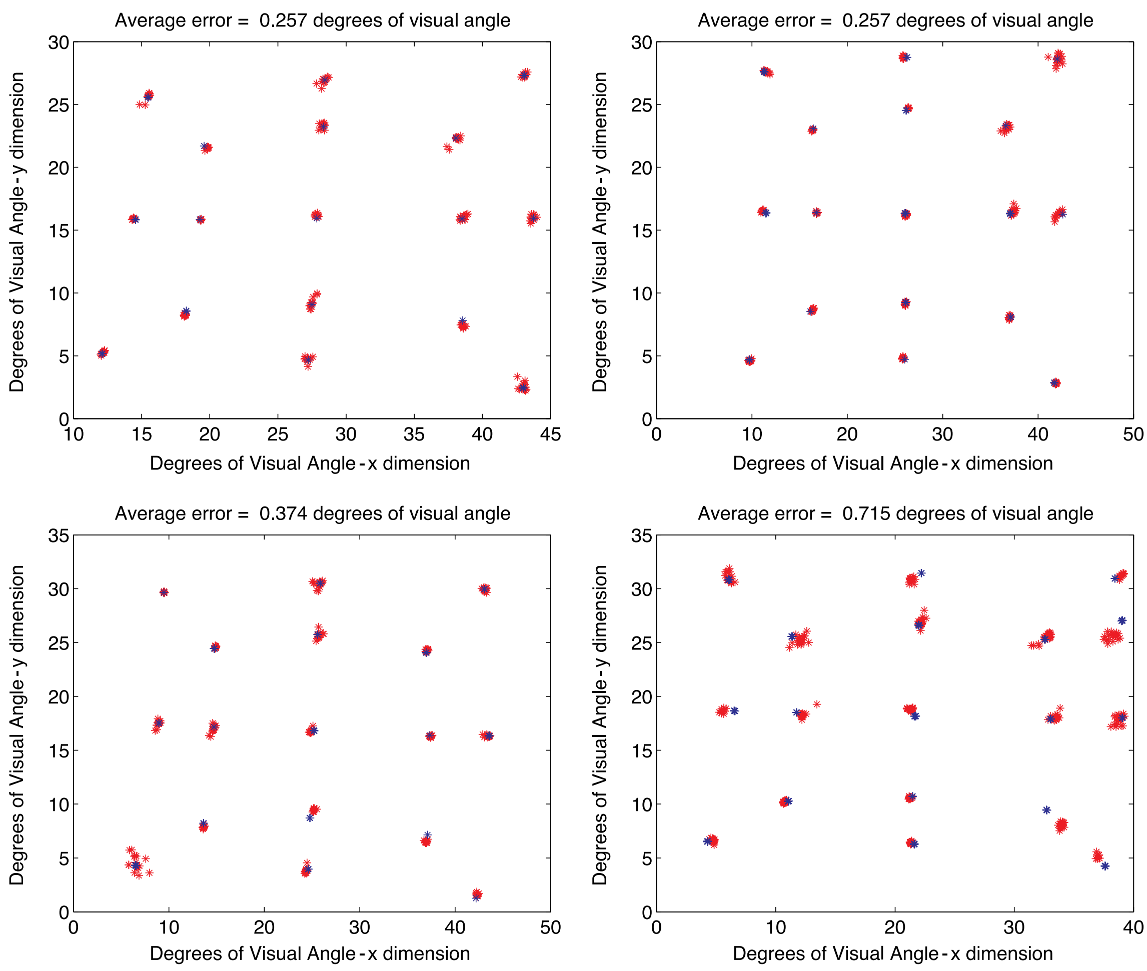


Fig. 2 Four example calibration screens, showing to-be-fixated points in dark gray (blue), and estimated gaze-location points in light gray (red). Overall error is a function of the spread of the light gray points (due mostly to errors in pupil estimation) and systematic error that moves the estimated gaze locations away from the to-be-fixated points (in dark gray). The latter errors result from nonlinearities that go beyond our 3rd-order polynomial mapping function, or may result from a subject not fixating where instructed. Both errors contribute to the accuracies reported in Fig. 3, as does estimation error of the corners. Top row: Highest accuracies, with minimal error distributed for all points (top left), or in the upper-right portion of the display (top right). Bottom row: Fit near the median accuracy on the left, and a relatively poor fit on the right. See online version for color

Fig. 3 we plot precision values computed using the *SD* method, which computes the mean location for a cluster of points and then computes the standard deviation of the distances from all points in a cluster to the centroid. This provides a mean precision of 0.298° and a median precision of 0.270° for our sample. This compares favorably with other eyetracker technologies (Duchowski, 2007). An alternative method of precision is the RMS calculation, which considers sequential data points within a fixation regardless of cluster membership. Here we compute the summed squared deviations from one point to the next, divide by the number of deviations, and take the square root. Using the RMS method, we find a mean precision of 0.329° and a median precision of 0.328° for our sample. Note that this method is sensitive to the sampling rate of the eyetracker, because as more time elapses, the eye has more of an opportunity to make a microsaccade or have the noise properties of the camera change. Therefore, eyetrackers that have a higher sampling rate may produce lower RMS precision values because of their ability to sample within a microsaccade. This value may also limit the ability of the software to separate saccades that are close together in space (<1°) as separate fixations, which should be viewed as a limitation of the ExpertEyes software and hardware.

The ExpertEyes software

ExpertEyes is open-source software written in Java for the processing of eyetracking data, and is freely available for download under the BSD license (http://code.google.com/p/experteyes). In the following subsections, we describe the different functionalities that ExpertEyes offers for the preprocessing of eyetracking data. The descriptions of the software functions go into some detail, which allows the reader to assess the overall

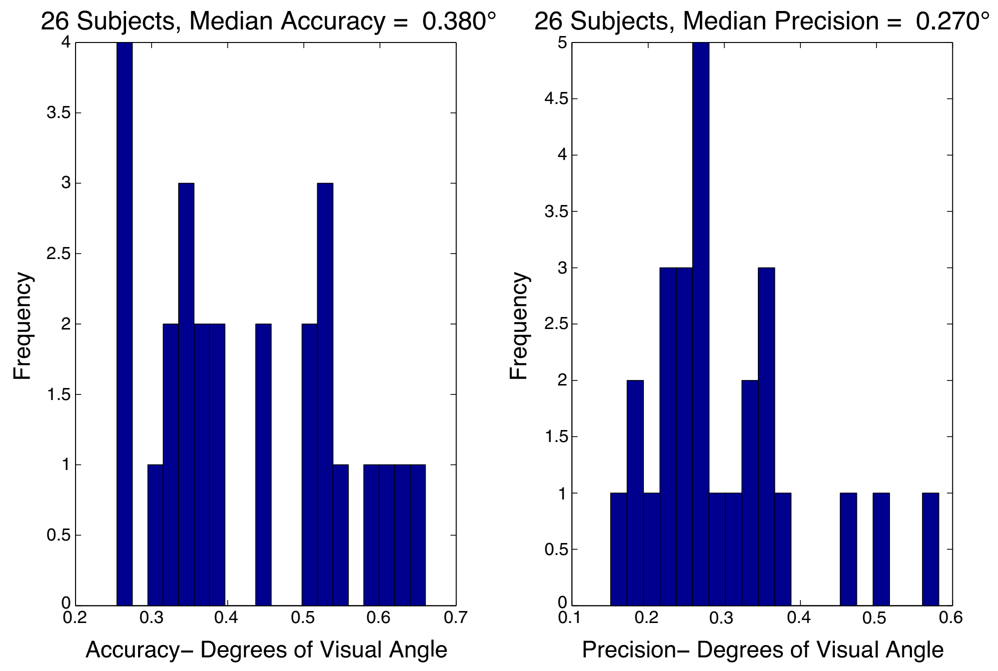


Fig. 3 Distribution of eyetracker accuracy from Kitchell et al., in preparation. The abscissa represents various levels of eyetracker accuracy for different subjects in degrees of visual angle, and the ordinate is the number of subjects falling in each range. The precision is computed using the *SD* method

functionality of the package, but they are not intended to provide complete documentation of the software for users. Video tutorials and a more detailed written description of functions, along with sample data, are available at the project's website.

The ExpertEyes software produces two forms of data. The first is the *head* (or *scene*) camera movie with the eye gaze overlaid on each frame. This is suitable for teaching environments and demonstrations, as well as qualitative data analyses. However, fixation finding and quantitative analyses require additional steps, and the software therefore outputs a text file with the eye-gaze position along with estimates of fiducial features in the scene, such as the four corners of a computer monitor. This text file can then be imported into MATLAB for fixation finding and more complex analyses. The exported data can also be imported to other eyetracking data-processing and visualization software. ExpertEyes also allows the user to select time intervals that correspond with specific events, such as trials. These can be manually selected or imported into the software, using, for example, log files from stimulus-presentation software such as E-Prime or Presentation.

Data recording is as simple as placing the headset on the subject, turning on both cameras, and starting movie recording using a remote control. We typically record for 20–30 min. After data stream recording, the software offers different functions, which can be grouped into three major processes:

1) Data import and construction of the eye model
2) Synchronization of video streams and calibration
3) Trial/event marking and data export.

The ExpertEyes forward eye model

In eyetracking, it is essential to have an eye model that can effectively take into account the vast differences present with regard to the eye's physiology and dynamics. In order to accomplish this task, a variety of algorithms have been developed over the years to balance the tradeoffs between versatility and computational efficiency, most focusing on tracking of the pupil. Three large groups of methods can be distinguished in the literature: appearance-based, shape-based, and hybrid methods. A complete description and discussion of eye models in eyetracking is outside the scope of this methods article. Nevertheless, in the following paragraphs we will briefly describe these three broad categories. (For a complete discussion of eye models in eyetracking, see Hansen & Ji, 2010.)

Appearance-based methods use the color distribution of the eye image (i.e., differences in photometric display of the pupil, iris, and sclera) to create a direct eye model. The most common approach for appearance-based methods uses an image patch model and a similarity measure in order to perform the eye detection. Importantly, these models can greatly differ, one from the other, depending on the source of information used to create the direct eye model; some approaches rely on the intensity or filtered intensity of the eye images (Grauman, Betke, Gips, & Bradski, 2001; Guestrin & Eizenman, 2006) while some other approaches acquire their information from a lower dimensional subspace (Hillman, Hannah, & Grant, 2003).

Shape-based methods build an eye model based on local features present in the images, such as eye or face fiduciary marks or their edges, corners, and/or contours. Using these features, a geometric eye model is built and a similarity measure is used in order to perform the eye detection. A popular example of shape-based models is the Starburst algorithm (Li, Winfield, & Parkhurst, 2005), which creates an active elliptical-shaped model that allows the use of several features based on gray-level maxima for detecting the iris. Importantly, shape-based models can greatly differ from each other depending on the parameters that allow template deformations included in the geometric eye model. These parameters for rigid and nonrigid deformations generally allow shape-based methods to be flexible enough and handle dynamic changes in the eye such as shape, scale, and rotation. A challenge for this algorithm is determining which points fall on the true edge of the pupil, and which fall on false edges created by the corneal reflection or the eyelid.

An alternative to the Starburst method that addressed this challenge was proposed by Zhu, Moore, and Raphan (1999), who used a pupil boundary detection algorithm. This method uses a pixel-classification approach that finds the derivative of the curvature function along the edge of the pupil to find those points that are likely associated with the true edge and those that are associated with occlusions as a result of artifacts such as eyelids and the CR. Once the true points are estimated, an ellipse may be fit through only the points on the pupil. This approach can handle occlusions of the pupil by the eyelid by as much as 30%–50% of the pupil. This approach appears to be quite accurate at finding the pupil, although it lacks any direct way to track the location of the corneal reflection using the same algorithm. The CR will often serve as a useful reference point to compensate for any shift of the eyetracker on the head that might be incorrectly interpreted as an eye movement (Duchowski, 2007).

Hybrid models offer a combination of the strengths and advantages of appearance- and shape-based models in order to compensate for their respective limitations. Examples of these hybrid models can be found in Ishikawa, Baker, Matthews, and Kanade (2004) and Witzner, Hansen, Nielsen, Johansen, and Stegmann (2002), where shape and appearance methods are combined to form a single generative model. (For a deeper survey on methods, see Duchowski, 2007; Hansen & Ji, 2010; Holmqvist, 2011.)

In developing the eye model for the ExpertEyes software, we found inspiration in the pupil-estimation techniques reviewed here. However, we found that all had difficulty with a common situation in which the corneal reflection (CR) from the eyetracker's infrared LED fell near the contour of the pupil. In addition, we wanted to use the CR, where possible, to protect against shifts of the eyetracker on the head that might translate into positional error in the gaze estimates. To simultaneously estimate both the pupil and the CR while addressing possible edge occlusion of the pupil by the CR, we construct a *forward model* that models the iris, CR, and pupil as independent grayscale regions.

Figure 4 illustrates our forward eye model. We start with a gray background of the approximate luminance of the iris. We first overlay a dark ellipse in the approximate region of the pupil (as determined by a user-specified threshold), and then add a bright ellipse in the approximate region of the CR (again determined by a user-specified threshold). This produces a new image that has the coarse appearance of an eye, which we will call the forward eye. This image has a gray background that is similar to the gray of the iris, a dark ellipse located near the pupil that is the approximate gray value of the pupil, and a light ellipse located near the CR. The grayscale values associated with the pupil and the CR regions are selected by the user via a grayscale sampling tool that samples the eye image in a region within the pupil or CR. Global thresholds for the pupil and the CR determine the initial guesses for the locations of these ellipses, and then region-finding procedures determine the rough starting locations for both elements.

The positions of the pupil and the CR in the forward eye image are refined using a standard gradient descent. The objective function is computed as the Euclidian distance on a pixel-by-pixel basis between the forward eye image and the actual eye image taken from the eye camera. If the pupil ellipse in the forward eye image is in an incorrect location relative to the actual pupil, this creates a larger distance between the two images due to the mismatch in gray values. The gradient descent (simplex) algorithm then adjusts the locations of the two ellipses in subsequent iterations, with the goal of producing a forward eye that has as little error as possible when compared with the actual eye image. Note that although there are features in the real image that are not present in the forward eye image (e.g., the eyelashes), these typically fall in the gray areas of the forward eye image and

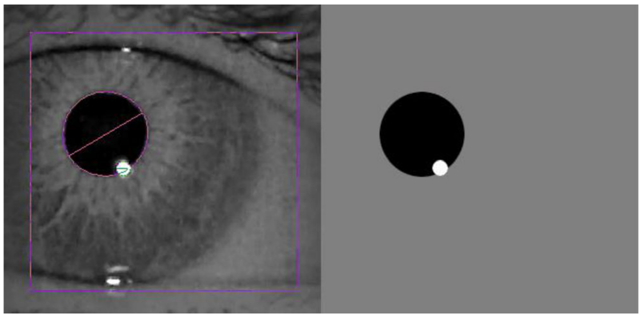


Fig. 4 Forward eye model: The eye image is compared against a forward eye model (right side). This image is constructed using ellipses of the approximate gray level of the pupil and CR, set on top of a gray background that is the approximate gray tone of the iris. The initial locations are found by thresholding the eye image, and then the forward eye model is refined by moving and resizing both ellipses (here shown mostly circular because of the position of the camera for this subject). We refine the locations using gradient descent, which adjusts the locations of the two ellipses, and the objective function is simply the pixel-by-pixel difference between the two images. As a result, it is important to match the gray levels of the artificial pupil and CR to their real image counterparts. In practice, the high-definition eye image is fairly tolerant to mismatches. See online version for color

are not influenced by the choice of parameters that govern the pupil and CR locations. They therefore contribute constant error and have no impact on the eventual eye-model fit.

The fact that the pupil is not a perfect ellipse when the CR falls on the edge (as illustrated in Fig. 4) is of no consequence, because we fit the pupil and CR locations simultaneously. Therefore the forward eye model compensates for the fact that the CR hides portions of the pupil by the fact that the forward eye image also hides portions of the pupil behind the CR in the model image. Therefore, as some commercial eyetracking systems do, we avoid the problem that the pupil is not an ellipse when the CR partially overlaps the edge of the pupil. Processing is done offline, which gives us ample opportunity to refine this model, even for different regions of visual space, to address a full range of issues that might affect gaze- estimation accuracy.

As with all models, there are some situations in which the procedures have difficulties. First, because the procedures are moderately sensitive to illumination, care should be taken to ensure that the light levels do not change dramatically throughout the experiment. Second, the IR illumination should be adjusted to cast an even pattern across the eye (although different parameter settings can compensate for this, to some degree). Finally, the position of the eye camera should be adjusted so that the corneal reflection remains on the corneal bulge and does not smear off into the sclera (which is true for any system that relies on the CR for drift correction).

When taking these precautions into account, we found the ExpertEyes forward eye model approach to be robust in a variety of testing conditions. Although the approach requires additional computing power, offline processing, and some initial hand-tuning by a human operator, the results are comparable and in some cases exceed those of commercial eyetrackers. We feel that having the human actively involved in the eyetracking analysis allows researchers to easily see how well the model is fitting, as well as improve the model fit through better parameter selection. In short, this approach not only allows the researcher to keep in touch with the quality of the eye-model fit, it requires active participation in the analysis. We believe that this reduces the possibility that the researcher will simply take the results of the eyetracker output on faith, without asking how often the eyetracker lost calibration or whether some visual locations have higher quality estimates than others.

The eye model will provide information about the position of the eye relative to the head. However, gaze tracking often requires information about the position of the eye relative to the world, effectively cancelling out head movements. This is done using a second camera that captures the position of the head relative to the world. Calibration procedures, described in a later section, combine the information from both cameras to produce an estimate of the gaze relative to the world that is independent of head movements.

Eye-model fitting

Here we describe the different steps that produce an eye-model fit, which will illustrate the functionality of the approach.

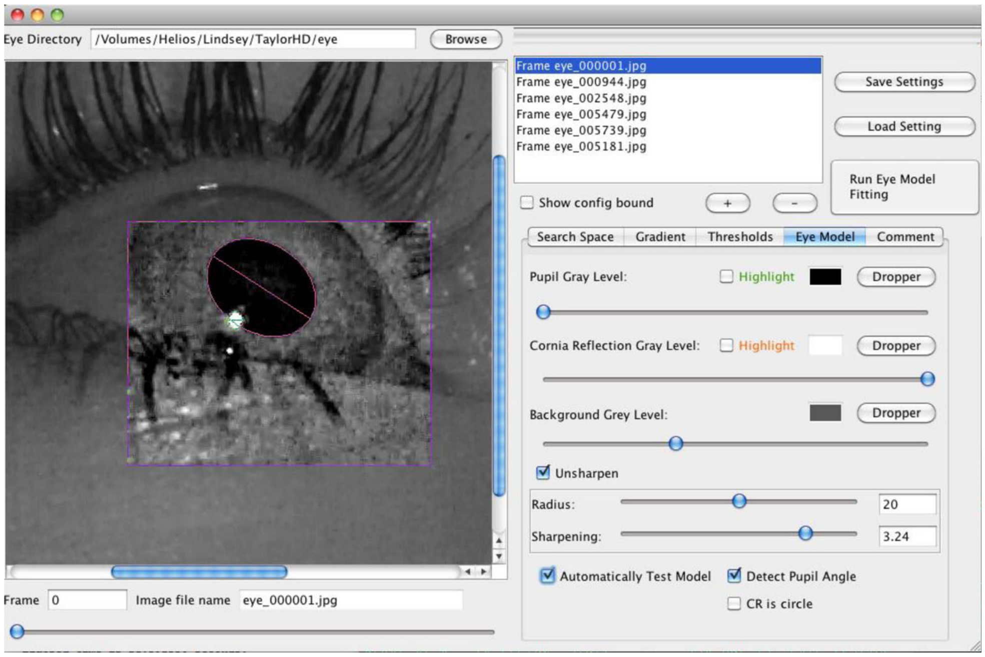
Under the *Tools* menu, the eye-model fitting module can be accessed, as shown in Fig. 5: Here, ExpertEyes provides the tools for finding suitable parameters to estimate the pupil and CR locations in each frame. This process is divided into two steps. The first one is oriented to prepare the data for the eye-model fitting algorithm, while the second focuses on finding the parameters that will build the model itself.

In order to optimize the estimation of pupil and CR positions, the Search Space tool allows users to select the area within the eye frames where the pupil and CR are located. This task is facilitated by the calculation of minimum and maximum pixel values across all frames, which illustrates the extent of travel of the pupil and eliminates issues in which the reflection off the skin may lead to erroneous CR estimation.

Some eye images may suffer from uneven illumination from a poorly aimed IR LED. The Gradient tool offers an optional gradient correction, to be applied in the event of recordings whose lack of illumination interferes with finding suitable parameters for the eye model. Finally, the Thresholds tool provides two sliders, which allow the user to adjust the limits of what pixels will constitute the pupil and the CR. At this point, the user can estimate the pupil locations (Fig. 5) in order to facilitate further parameter search.

After the search space has been properly adjusted, the appropriate thresholds have been selected for the pupil and the CR, and their locations have been estimated, the Eye Model tool provides three sliders to help the user find befitting model parameters. Each slider sets pixel gray levels that will be representative for the pupil, CR, and iris (background) at each particular frame. Likewise, the three dropper buttons allow users to directly click on the eye image, automatically setting the corresponding gray level to the selected pixels. Users can test current model parameters by checking the *Automatically Test Model* box. When necessary, checking the *Detect Pupil Angle* and *CR Is Circle* boxes can add additional corrections to the model parameters.

Additional model-fitting improvements can be achieved by checking the *Unsharpen* box, which will apply a standard two-setting unsharp mask. In order to create the unsharp mask, ExpertEyes uses a blurred version of the eye image to create a high-pass filter; the *Radius* slider controls the amount of blur of this image. A higher radius value will enhance larger-scale details; therefore, radius level will have an impact on the size of the edges to be enhanced. Finally, the *Sharpening* slider controls the amount of contrast and brightness that is added at the edges of the image. Hence, the implemented *Unsharpen* tool allows users to emphasize texture, correct for lens blurring, and improve overall eye image quality, thereby augmenting the



Eye Directory /Volumes/Helios/Lindsey/TaylorHD/eye

Frame eye_000001.jpg
Frame eye_000944.jpg
Frame eye_002548.jpg
Frame eye_005479.jpg
Frame eye_005739.jpg
Frame eye_005181.jpg

☐ Show config bound

Pupil Gray Level: ☐ Highlight ■

Cornia Reflection Gray Level: ☐ Highlight □

Background Grey Level: ■

☑ Unsharpen

Radius: 20

Sharpening: 3.24

☑ Automatically Test Model ☑ Detect Pupil Angle

☐ CR is circle

Frame 0 Image file name eye_000001.jpg

Fig. 5 Fitting a forward eye model to the pupil and corneal reflection. The purple rectangle illustrates the search space for the two features, which is also enhanced via an unsharpen mask. The model then iteratively fits the forward eye model to estimate the location of the pupil (purple ellipse) and the corneal reflection (gray ellipse - see online version for color)

accuracy of the forward eye-model solution. In practice, we have found that a moderate amount of unsharpening almost always improves the accuracy of the eye-model fit.

Eye-model fit for the entire recording session

We typically use between three and seven sets of parameters to account for the different gaze directions. After an appropriate set of parameters has been found, the ExpertEyes program uses multithreaded processes to estimate the pupil and CR locations for each frame in the experiment. This can take 2–4 hr of processing time on a modern multicore computer. The locations of the pupil and CR are then loaded into the main ExpertEyes database for subsequent processing.

Camera calibration and image undistorting

Because of the open-source nature of the ExpertEyes recommended hardware, the project uses off-the-shelf parts to build the eyetracker hardware. As described above, this has several advantages, but some disadvantages as well. One of the main and most important disadvantages users might encounter is image distortion affecting the scene frames (e.g., radial distortions) as a result of possible camera manufacturing imperfections. A second disadvantage might come from the fact that these cameras are built for commercial and not scientific use; therefore users might also want to establish the relation between frame pixels and real-world dimensions in order to do accurate real-world calculations. Calibrating the cameras solves the two aforementioned problems. Note that for users who simply want to view a movie of the scene camera with a gaze location overlay, camera calibration and scene undistortion are unnecessary. Moreover, even simpler optics can be used, such as the ones present in common webcams.

Rather than building camera calibration and undistorting routines into ExpertEyes, we have relied on an existing toolbox provided by Bouguet (http://www.vision.caltech.edu/bouguetj/calib_doc/). Camera calibration procedures provide a set of output parameters (e.g., focal length and displacement of lens along axes), which can be used later for image undistorting. A common calibration procedure is carried out using a checkerboard, where several pictures/movie frames are taken with the camera posing the checkerboard in different orientations. Several open-source toolboxes are available for computing the intrinsic, extrinsic, and distortion parameters needed to generate a homography matrix that allows making the relation between image coordinates and real-world coordinates. Additionally, distortion parameters can be used to undistort scene images. Sample camera calibration and undistortion routines can be found on the project's website (https://code.google.com/p/experteyes).

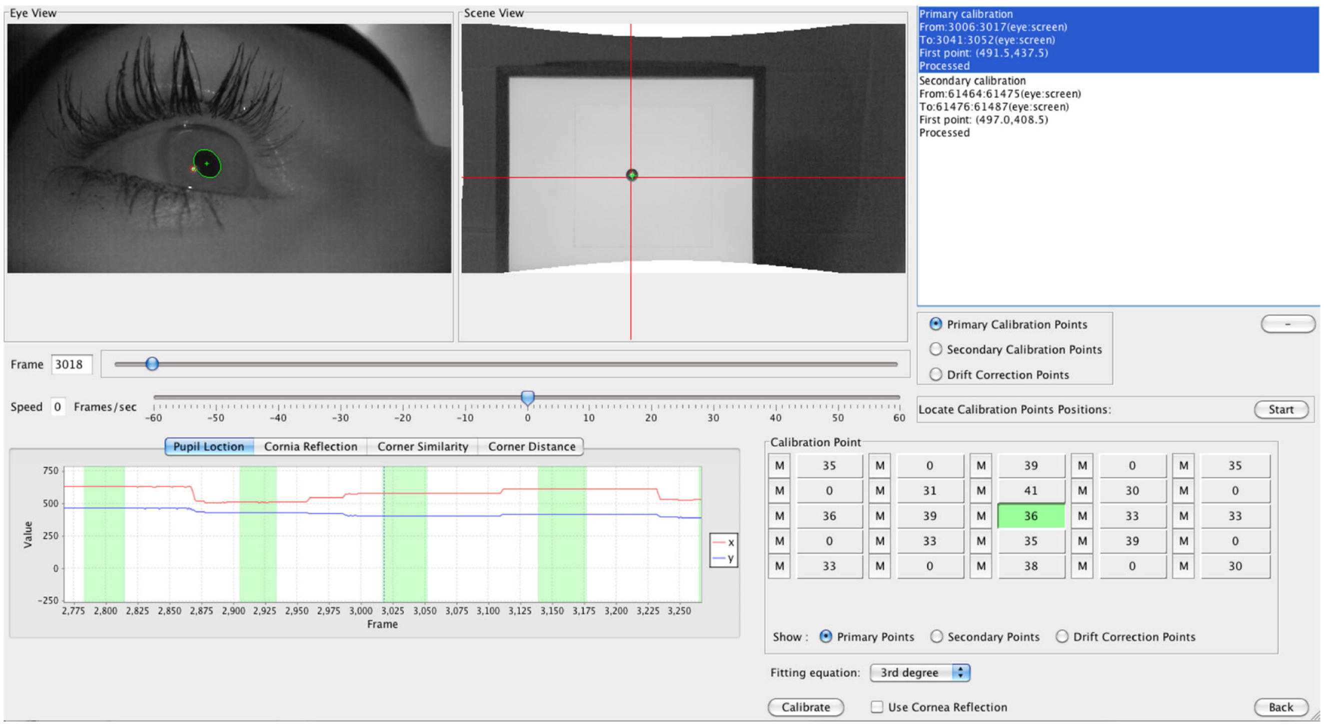
Eye-gaze calibration and drift correction

The eye camera captures the position of the eye relative to the head, whereas the scene camera captures the position of the head relative to the world (usually a computer monitor). However, the researcher is usually interested in the relation between the eye and the world. To link the two cameras, we use a very common calibration procedure in eyetracking research. We ask the subject to move his or her gaze to specific locations on a computer monitor or other flat plane. These are often denoted by dots on the monitor or LEDs that can be activated.

To define the relation between the locations in the scene camera and a position of the eye in the eye camera, we ask the user to specify time regions using the Calibrate tool (Fig. 6). This selects frames with known positions in the scene image (e.g., a set of bull's-eyes) to associate with specific eye-gaze positions. We assume that the subject looked at the location as asked. However, rather than linking a single pair of points, we define a range of 20–60 consecutive frames that are all associated with one location on the monitor. After this range is defined, the user selects the feature that the subject was instructed to fixate (e.g., a bull's-eye) and the ExpertEyes software uses Gabor jet matching to find this feature in subsequent video frames (Wiskott, Fellous, Kruger, & vonderMalsburg, 1997). This has the advantage of reducing the noise in the estimates (because any one frame might have some noise associated with it, but a cloud of points tends to average out this noise). It also provides a grouping method to determine the precision of the eyetracker, independent of accuracy (Fig. 3).

This procedure associates each estimated u–v pupil location in the eye image with a specific *x*–*y* location in the scene camera. We use a 2nd- or 3rd-order polynomial in both spatial dimensions to provide the best interpolated relation between the pupil and scene camera locations. We use QR Factorization to find parameters that minimize the least-squares deviations between the projected gaze locations (as estimated from the eye position) and the ground truth locations (as determined by the location of the bull's-eye in the scene camera view). These parameters are weights on both the *x* and *y* dimensions, as well as the *x*-squared and *y*-squared terms and the *x*y* interaction term. This allows for affine transformations as well as nonlinear distortions due to the interaction term. A 3rd-order polynomial is also an option, which also adds parameters for the *x*x*y*, *x*y*y*, *x*x*x*, and *y*y*y* terms. This will provide a better fit at the risk of poor extrapolation at the edges if the calibration points do not span the entire space of interest. A tool in the calibration screen allows for visualization of the amount of distortion introduced by each polynomial fit. Note that whereas the 3rd-order polynomial contains nine parameters, a typical 17-point calibration screen will produce 34 pairs of data points, so even the 3rd-order polynomial is nowhere near saturated. This ensures that the model accurately captures the relation between the eye position and locations in the scene image without overfitting, which could lead to poor interpolation or extrapolation.

These calibration procedures can be done at the start (i.e., primary calibration) and at the end of the recording (i.e., secondary calibration). The primary calibration is then used to estimate the amount of drift that is observed in the secondary calibration.



Primary calibration
From:3006:3017(eye:screen)
To:3041:3052(eye:screen)
First point: (491.5,437.5)
Processed

Secondary calibration
From:61464:61475(eye:screen)
To:61476:61487(eye:screen)
First point: (497.0,408.5)
Processed

[✓ Primary Calibration Points] Secondary Calibration Points Drift Correction Points

Locate Calibration Points Positions:

Calibration Point

M	35	M	0	M	39	M	0	M	35
M	0	M	31	M	41	M	30	M	0
M	36	M	39	M	36	M	33	M	33
M	0	M	33	M	35	M	39	M	0
M	33	M	0	M	38	M	0	M	30

Show : [✓ Primary Points] Secondary Points Drift Correction Points

Fitting equation: 3rd degree

Use Cornea Reflection

Fig. 6 Gaze calibration screen: This allows the user to select regions that correspond to different fixated locations, which ultimately assigns a relation between a location of the pupil in the eye camera and a location in the scene image. The scene view has been undistorted prior to process, which gives the white bows on the top and bottom of the image

The user has the option to include the corneal reflection in this interpolation, which may introduce some noise but compensates for shifts in the eyetracker on the head. Small shifts in the position of the eye camera will be interpreted as large shifts in gaze, but if the input to the interpolation routine is instead the vector between the pupil and the CR, such shifts are minimized. The tradeoff is that sometimes the estimate of the CR is noisier than the pupil, producing better accuracy (by reducing drift errors) but poorer precision. A compromise introduced into the software allows for drift-correction trials throughout the experiment that compensate for affine drift. Like some commercial systems, ExpertEyes allows drift correction to be applied easily by associating a known point in the scene image with eye-gaze positions in between the primary and secondary calibrations (e.g., after each block of trials). This provides an *x*–*y* offset for the calibration algorithm for subsequent frames. A final set of calibration trials at the end of the experiment helps demonstrate that the drift correction is accurate.

Monitor corner tracking

When collecting eyetracker data while participants interact with a computer monitor, it is essential to know which pixels from the recorded scene correspond to the monitor. It is because of this that ExpertEyes has a corner-finder tool that extracts Gabor features (implementing Gabor jets) at multiple scales/orientations in order to find monitor corners. This tool can be accessed using the Clean Data button in the main ExpertEyes window. Gabor jets are commonly used in automated object-recognition research (see Wiskott, Fellous, Kruger, & vonderMalsburg, 1997). Importantly, the user manually provides corner hints as well as the section of the recording where corners will be located, in order to reduce the space where Gabor jets are applied, thus optimizing accuracy and computation times.

Gaze-data exporting and fixation-finding in MATLAB

ExpertEyes allows users to export gaze data into a text file. Detailed description of ExpertEyes output files can be found in the supplementary material on the Google code site (https://code.google.com/p/experteyes). The output text file contains all the information necessary to perform various data analyses, including automated fixation finding. We have developed a custom automated fixation-finding algorithm using eye movements as the main fixation-parsing factor. This algorithm has been successfully applied to eyetracking data in previous reports (Busey, Yu, Wyatte, & Vanderkolk, 2013; Busey et al., 2011). Our approach sets a user-defined, data-driven velocity threshold in order to segment continuous gaze time-series into clusters indicating drastic changes in eye locations. Our velocity threshold relies on the magnitude of velocity at each point in the gaze time-series and can be complemented with a second threshold for minimum fixation-duration, thus preventing spurious brief fixations from populating the output. When necessary, a running median filter over the gaze time-series is applied in order to reduce noise in gaze estimation. Sample code for applying our automated fixation-finding approach can be found on the project's website.

The output of the ExpertEyes code includes the raw estimate of the pupil location in the eye camera, the projection of this into the scene camera image, and if monitor coordinates are specified, also the location of the eye gaze relative to the monitor coordinates.

Discussion

ExpertEyes is an inexpensive and open-source eyetracking package for research and education. As are many commercial and noncommercial portable eyetracker systems available in the market, it is useable in environments in which subjects may be hard to obtain (e.g., ExpertEyes has been used to collect data from fingerprint examiners, Busey et al., 2011, and domestic dogs, Rossi et al., 2013). It is relevant to emphasize that ExpertEyes is an offline-based eyetracker system and cannot be used for online eye-movement analyses. Similarly to other offline-based eyetracker systems, our software's offline data processing allows for recovery of data that might otherwise be lost in real-time systems.

It is important to mention that ExpertEyes is not intended to replace commercial eyetracker systems or to be used as a remote assistive technology (i.e., computer input, environmental control, etc.). Many open-source and/or low-cost eyetracker systems have been designed as remote assistive technologies (e.g., from the list of open-source and low-cost eyetracker systems that can be found in the archived communication by gaze interaction [COGAIN] website (active from 2004 to 2009), most of the projects listed are intended for environment control and communication for people with disabilities).

Importantly, ExpertEyes offers most of the capabilities of other packages, including drift correction, data cleaning, trial and event marking, and data export. It is worth mentioning that ExpertEyes is cross-platform, in contrast to other open-source software, which might be exclusive for Microsoft Windows (e.g., CARPE), Mac OS X (e.g., RIT code), or require specific software that is not open source, such as MATLAB (e.g., openEyes). However, ExpertEyes is not useful for real-time or gaze-contingent designs, and the data processing is relatively time-intensive. The temporal resolution is limited to the used hardware (currently 60 Hz), which rules out tasks in which rapid eye kinematics are important.

Our system has been shown to be successful in naturalistic conditions in adult humans and nonhumans. For example, Rossi and collaborators (Rossi, Parada, & Allen, 2010; Rossi et al., 2013) successfully combined the Babcock and Pelz (2004) with the ExpertEyes-suggested head-mounted hardware for pet dogs. Relying on the flexibility and robustness of the ExpertEyes software, they successfully collected eyetracking data from pet dogs while performing a cuing task in free-moving conditions.

Thus, we have seen that independently of eye physiology (e.g., eye color) or vision-correction procedures (e.g., contact lenses) ExpertEyes is capable of successfully fitting the eye model. Different shapes of eyelids and participants wearing glasses can be successfully measured by simply varying the angle of the eye camera and/or position of the IR illumination source, allowing our system to handle a wide variety of subjects.

Like most eyetracking systems, ExpertEyes has problems fitting the eye model to extreme gaze angles: Designing experiments in which participants do not have to deviate their gaze too much from the direction of their head usually solves this problem. Nevertheless, in situations in which participants can move their heads, the probability of encountering extreme gaze angles is low.

At present, most eyetracking systems available on the market are monocular. Likewise, our suggested hardware follows this design. In eyetracking literature, it is widely assumed that eyes move in temporal synchrony, however, this assumption has been shown not to be true in every case. In certain tasks, participants do show binocular disparity (Cornell, MacDougall, Predebon, & Curthoys, 2003; Liversedge, White, Findlay, & Rayner, 2006). Again, the nature of the research would define the need for binocular or monocular eyetracking. Most research agendas can use the benefits of monocular eyetracking (e.g., less expensive monetarily and computationally). In contrast, researchers testing developmental, special, or clinical populations might need to use binocular tracking systems (Holmqvist, 2011). As of now, ExpertEyes does not support binocular eyetracking. One of the big advantages of binocular eyetracking is the robustness against the parallax error (i.e., nonlinearities) caused by disparate distance between calibration points and objects in the scene. This error is potentially encountered by all monocular eyetracking systems. In order to reduce the parallax error as much as possible, ExpertEyes' suggested hardware locates the scene camera as close to the eye as possible (Fig. 1).

Most eyetracking systems output data samples using the coordinate system (x,y) of the stimuli and/or using the coordinate system of the scene video camera. ExpertEyes software exports data samples using both coordinate systems. If the location of corners has been provided, the exported data will be projected using monitor coordinates. Otherwise the exported data uses scene video coordinates. Exported movies with overlaid gaze locations follow this rule as well.

In conclusion, ExpertEyes is a hardware-independent open-source eyetracking software package that produces quite accurate gaze estimation in comparison with commercial eyetracking, at the cost of offline processing. As an open-source platform, it is transparent but does require some technical skills on the part of the experimenter. The flexibility of the package provides a large number of user-customizable software and hardware parameters that considerably widen the range of participants and experimental designs that researchers can use without compromising robustness.

Author Note T.B. was supported by National Institute of Justice Grant Nos. 2005-MU-BX-K076 and 2009-DN-BX-K226. The authors thank Lindsey Kitchell, Dr. Alejandra Rossi, Daniel Smedema, and Dr. Nicholas Port.

References

Babcock, J. S., & Pelz, J. B. (2004). *Building a lightweight eyetracking headgear*. San Antonio, TX: Paper presented at the Eye Tracking Research & Applications Symposium.

Busey, T., Yu, C., Wyatte, D., & Vanderkolk, J. (2013). Temporal sequences quantify the contributions of individual fixations in complex perceptual matching tasks. *Cognitive Science, 37,* 731–756.

Busey, T., Yu, C., Wyatte, D., Vanderkolk, J. R., Parada, F. J., & Akavipat, R. (2011). Consistency and variability among latent print examiners as revealed by eye tracking methodologies. *Journal of Forensic Identification, 61,* 60–91.

Cornell, E. D., MacDougall, H. G., Predebon, J., & Curthoys, I. S. (2003). Errors of binocular fixation are common in normal subjects during natural conditions. *Optometry and Vision Science, 80,* 764–771. doi: 10.1097/00006324-200311000-00014

Duchowski, A. T. (2007). *Eye tracking methodology: Theory and practice* (2nd ed.). London: Springer.

Grauman, K., Betke, M., Gips, J., & Bradski, G. R. (2001). *Communication via eye blinks-detection and duration analysis in real time*. Kauai, HI: Paper presented at the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

Guestrin, E. D., & Eizenman, M. (2006). General theory of remote gaze estimation using the pupil center and corneal reflections. *IEEE Transactions on Biomedical Engineering, 53,* 1124–1133.

Hansen, D. W., & Ji, Q. (2010). In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32,* 478–500.

Hillman, P., Hannah, J., & Grant, P. (2003). *Global fitting of a facial model to facial features for model-based video coding*. Rome, Italy: Paper presented at the International Symposium on Image and Signal Processing and Analysis.

Holmqvist, K. (2011). *Eye tracking: A comprehensive guide to methods and measures*. Oxford, New York: Oxford University Press.

Ishikawa, T., Baker, S., Matthews, I., & Kanade, T. (2004). *Passive driver gaze tracking with active appearance models*. Nagoya, Aichi, Japan: Paper presented at the World Congress on Intelligent Transportation Systems.

Li, D., Babcock, J., & Parkhurst, D. J. (2006). *OpenEyes: A low-cost head-mounted eye-tracking solution*. San Diego, CA: Paper presented at the Symposium on Eye Tracking Research & Applications.

Li, D., Winfield, D., & Parkhurst, D. J. (2005). *Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches*. San Diego, CA: Paper presented at the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

Liversedge, S. P., White, S. J., Findlay, J. M., & Rayner, K. (2006). Binocular coordination of eye movements during reading. *Vision Research, 46,* 2363–2374. doi:10.1016/J.Visres.2006.01.013

Nevalainen, S., & Sajaniemi, J. (2004). *Comparison of three eye tracking devices in psychology of programming research*. Carlow, Ireland: Paper presented at the 16th Annual Workshop of the Psychology of Programming Interest Group.

Rossi, A., Parada, F. J., & Allen, C. (2010). *DogCam: A way to measure visual attention in dogs*. Eindhoven, The Netherlands: Paper presented at the Measuring Behavior conference.

Rossi, A., Parada, F. J., Haga, Z., Grooteboer, C., Lenoir, M., Wels, H., … Allen, C. (2013). Eye-tracking the gaze of dogs and humans in a pointing gesture study. *Journal of Veterinary Behavior: Clinical Applications and Research, 8*(4). doi:10.1016/j.jveb.2013.04.049

Wiskott, L., Fellous, J. M., Kruger, N., & vonderMalsburg, C. (1997). Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19,* 775–779. doi:10.1109/34.598235

Witzner, D., Hansen, J. P., Nielsen, M., Johansen, A. S., & Stegmann, M. B. (2002). *Eye typing using Markov and active appearance models*. Orlando, FL: Paper presented at the Sixth IEEE Workshop on Applications of Computer Vision.

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