Behavior Research Methods
https://doi.org/10.3758/s13428-019-01305-2

RemoteEye: An open-source high-speed remote eye tracker

Implementation insights of a pupil- and glint-detection algorithm for high-speed remote eye tracking

Benedikt Hosp[1] · Shahram Eivazi[1] · Maximilian Maurer[1] · Wolfgang Fuhl[1] · David Geisler[1] · Enkelejda Kasneci[1]

Abstract

The increasing employment of eye-tracking technology in different application areas and in vision research has led to an increased need to measure fast eye-movement events. Whereas the cost of commercial high-speed eye trackers (above 300 Hz) is usually in the tens of thousands of EUR, to date, only a small number of studies have proposed low-cost solutions. Existing low-cost solutions however, focus solely on lower frame rates (up to 120 Hz) that might suffice for basic eye tracking, leaving a gap when it comes to the investigation of high-speed saccadic eye movements. In this paper, we present and evaluate a system designed to track such high-speed eye movements, achieving operating frequencies well beyond 500 Hz. This includes methods to effectively and robustly detect and track glints and pupils in the context of high-speed remote eye tracking, which, paired with a geometric eye model, achieved an average gaze estimation error below 1 degree and average precision of 0.38 degrees. Moreover, average undetection rate was only 0.33%. At a total investment of less than 600 EUR, the proposed system represents a competitive and suitable alternative to commercial systems at a tiny fraction of the cost, with the additional advantage that it can be freely tuned by investigators to fit their requirements independent of eye-tracker vendors.

Keywords High-speed · Open-source · Eye · Tracking · Vision research · Low-cost · Saccade · Glint · Pupil

Introduction

A quick look around any modern-day eye-tracking lab reveals the presence of low-cost eye trackers for capturing eye movements in a wide variety of studies. For instance, in interactive applications (Hansen, Alapetite, MacKenzie, & Møllenbach, 2014; Kangas et al. 2014; Hansen, Ahmad, & Mardanbegi, 2014; Canare, Chaparro, & He, 2015; Ramos, Hanada, Da Graça, Pimentel, & Teixeira, 2017) and eye movements data analysis experiments (Rodrigue, Son, Giesbrecht, Turk, & Höllerer, 2015; Zhang, Fan, Yuan, & Peng, 2015; Coyne & Sibley 2016; Sari, Santosa, & Wibirama, 2017; Murugaraj & Amudha 2017; Jbara & Feitelson 2017) rely on these devices. The most commonly used technique for building such eye trackers is video oculography (VOG), which includes an image processing algorithm to detect the pupil and/or corneal reflections (named glints) based on an eye's images recorded by a video camera with or without infrared (IR) illumination. While low-cost eye trackers such as the Eye Tribe Tracker, Tobii EyeX, and Pupil Labs are becoming more common in different research fields, several authors have pointed out the restrictions of these eye trackers (Funke et al. 2016; Ooms, Dupont, Lapon, & Popelka, 2015; Santini, Niehorster, & Kasneci, 2019). Results from Funke et al. (2016) showed that—compared to commercial high-cost systems—these

✉ Benedikt Hosp
benedikt.hosp@uni-tuebingen.de

Shahram Eivazi
shahram.eivazi@uni-tuebingen.de

Maximilian Maurer
maximilian.maurer@student.uni-tuebingen.de

Wolfgang Fuhl
wolfgang.fuhl@uni-tuebingen.de

David Geisler
david.geisler@uni-tuebingen.de

Enkelejda Kasneci
enkelejda.kasneci@uni-tuebingen.de

[1] Human Computer Interaction, University of Tübingen, Tübingen, Germany

Published online: 24 March 2020

low-cost eye trackers have moderate accuracies close to 2 degrees (e.g., faceLAB with 1.93 or Smart Eye Pro with 2.4), though they suffer from high rates of data loss (approximately 22% lost data). The lack of high-speed eye trackers is another limitation of further significance. To our knowledge, all existing low-cost eye trackers are built with slow-frame-rate cameras ranging anywhere between 30 and 120 Hz. Low sampling frequencies hinder the applicability of low-cost eye trackers for robust saccade and microsaccade detection.

Moreover, Holmqvist et al. (2011) provided evidence that saccade velocity and acceleration are not measured accurately at lower sampling frequencies. As such, microsaccade studies only use eye trackers with a sampling rate of 200 Hz or more (Holmqvist et al., 2011). Additionally, in sports-related expertise studies, the classification of expertise is usually based on fixations (mostly duration and number of fixations). Also visual search strategies, like scene exploration, are typically interpreted based on fixations (Mann, Williams, Ward, & Janelle, 2007). High sampling rates lead to more accurate detections of fixations compared to low sampling systems, which have more temporal error and cannot accurately detect rapid saccades (Andersson et al., 2010).

Since 2010, high-end commercial eye trackers offer 1000–2000 Hz systems (tower-mounted); however, such solutions usually cost too much for a small research laboratory, limiting the research directions. Tracking eye movements at such a high speed is problematic for a number of reasons. First, and most important, are camera and optical sensor limitations. With higher speed, there is less light reaching the optical sensor as the shutter of the camera closes earlier than with slow speed. A lower exposure time leads to less illuminated images with lower contrast, which makes feature detection harder. Another limitation is the size and weight of high-speed cameras, as it is preventing their usage in head-mounted eye trackers. Furthermore, remote eye-tracking systems require high-resolution images of the face and eyes to accurately estimate the pupil and glint center. Additionally, we should also add the pricing challenge: For instance, a 500-Hz Full-HD camera from OPTRONIS (CR600X2-M-8G-GE-XM) runs currently at the retail cost of about 10,000 euros.

Designing a high-speed eye tracker invites questions concerning the time/accuracy trade-off: e.g., (1) what is an optimal camera resolution where a pupil and glint detection algorithm performs under 2–3 ms, and (2) to what extent are current consumer computer resources (CPU, RAM, Hard drive) capable of real-time eye tracking.

With these concerns in mind, the current work addresses the design of an open-source, high-speed eye tracker—involving hardware and software—in a remote-based setting. As certain challenges are not entirely new in the field of eye tracking and have been solved with other methods, we first review papers that aim to design low-cost high-speed eye trackers and point out their restrictions in chapter "Related work". More important, we propose a novel approach for real-time glint and pupil detection algorithms (chapter "System description"). In chapter "Method" we describe the study we conducted to evaluate the system. The results follow in chapter "Results" and in chapter "Discussion" we discuss the results from the study in reference to the strengths and weaknesses of the system.

Related work

Advances in digital imaging technologies have widely impacted the availability of high-performance digital cameras for researchers in the machine vision domain. This rise was accompanied by a growing body of computer vision studies examining different algorithms for detecting eye features (e.g., pupil and glint) from images captured by cameras (Fuhl, Tonsen, Bulling, & Kasneci, 2016; Morimoto, Koons, Amir, & Flickner, 2000; Ebisawa 1970). While robust and accurate detection of the pupil and glint position is a key feature of an eye tracker, sampling frequency is an important component in multiple areas, such as microsaccade studies (Holmqvist et al., 2011). To date, tower-mounted commercial eye trackers have been developed that support 1000–2000 Hz monocularly or 500 Hz binocularly with average latencies of less than 2 ms (Holmqvist et al., 2011). These solutions, however, remain unaffordable for many researchers due to their pricing options. Aiming to extend the application of high-speed tracking into more appropriate price ranges, there is a small, but growing, body of research that focuses on building such eye trackers (Dera, Boning, Bardins, & Schneider, 2006; Long, Tonguz, & Kiderman, 2007; Schneider et al. 2009; Sogo 2013; Farivar & Michaud-Landry 2016).

Typically, open-source pupil and glint detection algorithms have been developed for head-mounted trackers (Fuhl et al., 2018; Fuhl, Santini, Kasneci, Rosenstiel, & Kasneci, 2017; Li, Babcock, & Parkhurst, 2006; San Agustin et al. 2010; Santini, Fuhl, & Kasneci, 2018; Stengel, Grogorick, Eisemann, Eisemann, & Magnor, 2015; Parada et al. 2015), where a low-cost web or lightweight camera is used to capture eye images with 30–120 fps. From the camera images, these eye trackers then employ pupil-detection algorithms in conjunction with gaze estimation. It is beyond both the scope and purpose of this chapter to offer a complete coverage of eye-tracking algorithms. Rather, our intent here is to highlight works about state-of-the-art pupil-detection methods that report runtimes or have been explicitly designed for high-speed eye tracking.

To the knowledge of the authors, there is no systematic investigation of the run time, but several papers (Fuhl et al. 2016; Fuhl, Santini, Kübler, & Kasneci, 2016) have measured the performance of various algorithms and compared them with their own pupil-detection method. For example, Fuhl et al. (2016) published their ElSe algorithm—based on ellipse evaluation of a filtered edge image—and achieved a 14.53% improvement on the pupil-detection rate compared to other methods. ElSe was tested on a common desktop computer (an Intel i5-4570, 3.2-GHz CPU with a 60-Hz camera) and achieved a runtime of 7 ms (∼140 Hz). However, there are no runtime measures for the other steps in the eye-tracking pipeline (e.g., glint detection, gaze estimation), which would extend the total runtime. They reported 8 ms for Świrski, Bulling, and Dodgson (2012), and 6 ms for ExCuSe (Fuhl, Kübler, Sippel, Rosenstiel, & Kasneci, 2015) pupil detection methods. In other words, any eye tracker using these algorithms will be able to track eye movements with a sampling frequency of 30–120 Hz.

Another influential technical contribution to our understanding of the eye-tracking processes time-line can be found in the work of Santini, Fuhl, Geisler, and Kasneci (2017). Of significance is their report of software pipeline latency using a Dikablis eye tracker with Windows 8.1 installed on a PC running on an Intel i5-4590, 3.30-GHz CPU, with 8GB RAM. From the image acquisition on the camera, to saving the videos on the hard drive, the whole process took 17.25 ms (with a standard deviation of 2.07 ms). Similarly Kassner, Patera, and Bulling (2014) inquired into the latency of the Pupil Pro Headset revision 022 tracker. They measured the latency based on 1200 successive samples on a Lenovo X201 laptop with an Intel Core i7 620M processor running on Ubuntu 12.04. They observed a total latency of 124 ms (with a standard deviation of 5 ms).

Long et al. (2007) developed a head-mounted eye tracker capable of tracking at 150 Hz. In their system, the processing frame rate was increased by eliminating the bottleneck of transferring and processing the entire image. Their symmetric mass center algorithm was developed over the naive center of mass algorithm. The processing region was reduced using the location and the approximate position of the pupil on a low-resolution image. Dera et al. (2006) designed a large head-mounted eye tracker for 3DOF real-time motion control of a head-mounted camera. They identified the pupil by the brightness gradient between pupil and iris followed by an edge search (Zhu, Moore, & Raphan, 1999). In their implementation, thresholding took 0.3 ms, pupil search 0.7 ms, and edge detection took 1.1 ms. As such, they were able to perform eye tracking in the range of 500–600 Hz (Schneider et al., 2009).

Another way to achieve a high-speed recording system is to put a larger camera with higher frame rate in front of the user (remote eye tracker). However, the image will then include not only the eye region but also other regions including the entire face and other background objects. As such, additional image-processing techniques are required to distinguish the eye region apart from other parts of the image. Ebisawa (1998), for example, suggested using two light sources and the image difference method for pupil detection. The author argued that his method can detect the pupil center every 1/60 s. Following the same technique as Ebisawa (1998) and applying a 3D model-based method for gaze estimation, Hennessey and Lawrence (2009) designed a free head motion remote eye tracker that had an accuracy of ±8 degree of visual angle. They used a monochrome DragonFly Express from the former Point Grey Research with a resolution of 640 × 480 pixels at 200 fps. Of significance here is the Sogo (2013) GazeParser software. The author reported the performance of his system by comparing various sampling rates. The maximum sampling interval reaches 4.0 ms in all measurement conditions. As this value is twice the ideal value, the authors implied that data loss might occur often. In their setup, they use one PC for stimulus presentation and one for eye tracking that are connected over ethernet. With frequencies of higher than 400 fps, the recorder PC could not process camera images in time during recording, as 99.5 % of the samples needed at least 2.15 ms: Which corresponds to ± 470 fps. In a more recent study, Farivar and Michaud-Landry (2016) used the GazeParser to achieve a sampling rate of 450 fps. In the results, they reported 4.99 ms latency when recording images with high speed. They did not investigate on methods of how to synchronize the gaze signal with the stimulus frames and therefore reported a frame rate for the gaze signal of 200 Hz.

Drawing on these insights from the literature, we present a high-speed eye tracker. The focus of our development is to achieve 2 ms or less for feature detection, so that real-time eye tracking at 500 Hz is possible using a common desktop computer. The contribution, specifically, is to share our experience with a real-time glint and pupil detection algorithm. We present a system that is capable of detecting eye features at 570 Hz and above. To show that our system is capable of tracking at such speed, we implemented the full pipeline, including a calibration and gaze estimation through a 3D eye model technique.

System description

Recording eye movement data based on VOG method has relatively straightforward requirements; First, there is a need for one or more cameras to capture eye images. Second, there is a need for an algorithm to detect a few distinguishable features from the eyes images. Especially

Table 1 List of components for building the high-speed eye tracker

Components	Quantity	Description	Price
Eye camera	1	IDS:UI-3130CP	400
Infrared emitter	9	SFH 4557 LED	5
IR filter	1	Optolite IR acrylic	5
Camera lens	1	Computer 12 mm	100
Resistor	3	180Ω	5
Camera tripod	1	any model	20
AC adapter	1	12V, 1A	5

The prices are in euros

for calculating the pupil center, there are different possible features that can be used (e.g., limbus, pupil contour). In this paper, we are detecting the pupil center directly from a region of interest based on the glint detection. In the following subsection, we describe our system.

Hardware design

To reduce the cost of the eye tracker (less than 600 euros), we use only one camera. Moreover, IR emitters (LEDs) are used to create glints and a dark pupil effect in the eye images. Table 1 shows the components necessary to build the eye tracker. The main challenge of building a high-speed eye tracker is to find a suitable camera. Most of the high-speed cameras are expensive or have a very low resolution. We use a CMOS PYTHON 500 by ON Semiconductor with a 1/3.6" global shutter sensor. A USB 3.0 monochrome camera equipped with this sensor is able to work with 575.0 fps at 800 x 600 pixel resolution.

We use SFH 4557 LEDs (840 nm) from OSRAM Opto Semiconductors Inc. for IR illumination, and Optolite Infra Red Acrylic visible filter due to performance/cost ratio (Instrument Plastics Ltd Optolite Infra Red Acrylic, n.d.) to block the wavelengths less than 740 nm. Figure 1 shows the eye tracker close to the computer screen. Based on the Intersil IR safety guide, our LEDs have $0.8W/m^2$ irradiance in total, which is far below the $100W/m^2$ limit (according IEC 62471 (based on CIE S009)).[1]

We developed our eye-tracking system using a PC with Intel I7, 7700k, 4.2 GHz and 16 GB RAM. Although we tested our eye tracker using this computer, our results show that there is no need for that much RAM or CPU power.

Software design

The whole system is written in C/C++ and uses OpenCV, Qt and boost libraries. On startup, a user interface is shown where calibration, gaze estimation, image or video upload, and playback can be chosen. The user interface is built for easy usage of the different processes (Fig. 2). All data are saved inside a user folder. The preferred sequence of using the system is as follows:

- **Show face**: At first the user should have a look at the head position. The "show face" function opens the camera and shows the users face with the pupil center and glints detections mapped on the image (example see Fig. 5). In this view, the user can already get an impression of the detection quality and whether the head should be adjusted for better position and focus in the camera.
- **Calibration:** After adjusting the head position, a calibration procedure is run in order to allow the system to adjust the default values of the 3D eye model to the user. This step is also necessary for each user to train our pupil detection algorithm to learn the most probable pupil sizes (explained in detail in chapter "Pupil detection") and afterwards the gaze estimation algorithm can calculate the visual/optical axis based on the estimates of the pupil and cornea diameter. This step takes around 10-30 s.
- Then, the user can have a look at the calculated values that are plotted onto the calibration pattern. As the gaze estimation highly depends on the results of the calibration, a shift or similar artifacts can be seen in this screen. After calibration, the user has several options.
- **Gaze estimation evaluation:** An optional gaze estimation evaluation procedure for calculating the accuracy and precision on unseen data can be chosen.
- **Live Gaze:** If the user wants to experience how well the tracker works, he can start a live gaze overlay where he can see his estimated gaze on the screen.
- **Stimulus playback:** If the user has uploaded an image or video as stimulus, then he is able to playback this stimulus while recording gaze in real time. As stimuli can have different speeds, we synchronize the gaze and stimulus by saving the timestamps of the gaze signal when it is captured and the stimulus frame, when it is presented on the screen. Afterwards, mapping takes the timestamp of the gaze signal and matches the timestamp of the corresponding stimulus frame. The resulting video has the speed of the gaze signal (575 Hz). For example, if the stimulus video has 30 Hz, each stimulus frame timestamp corresponds to about 19 gaze timestamps. The resulting video contains about 19 times the same stimulus frame but with a different gaze signal mapped on it. The raw gaze data with the timestamps and the stimulus with the mapped gaze on it are placed inside the user folder.

[1] https://www.intersil.com/content/dam/Intersil/documents/an17/an1737.pdf

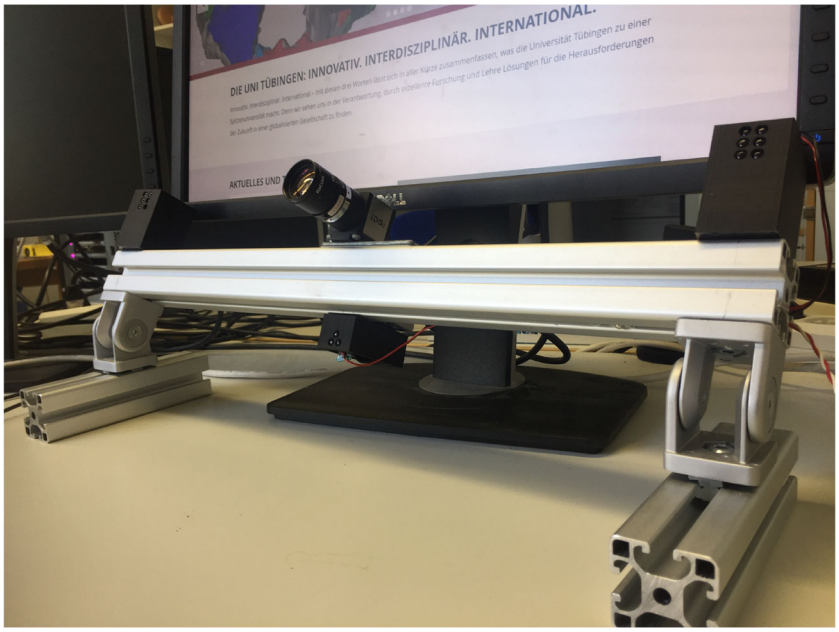

Fig. 1 We use a self-built aluminium stand to keep the eye tracker close to the screen and the angle of the tracker adjustable. The IR illuminators are placed inside 3D printed black boxes in the center under the camera and at both horizontal ends. The distance between camera and left and right side LEDs is 25 cm. The distance from the bottom LEDs to the camera is 8 cm

In general, our system consists of two main threads. One handles image capturing and feature detection from both eyes and the second thread handles the display of the stimulus and the timed saving of the detected features according to the stimulus. The main components of the first thread are: (1) image capturing, (2) glint detection, and (3) pupil detection. The second thread shows the stimulus and grabs the saved features from the first thread. Either the calibration, the evaluation, live gaze overlay, or the video playback procedure run in the second thread. These procedures only differ in the usage of the captured data. During calibration, the detected values are used to estimate the visual and optical axis of the users eyes, and during gaze estimation and video playback, the values are used to estimate the gaze based on the calibration model. In all procedures, the values are saved inside the user folder (Fig. 3).

Image capture

The uEye camera software from the IDS-Imaging company provides a SDK to set camera parameters. For example, the manual settings in our system are: 575 FPS, 1.6 ms exposure time, and image brightness. The camera gain value is another important parameter, as a high gain value would lead to a very noisy image. With higher frame rates, the exposure time gets shorter and therefore less light is reaching the sensor. For detection, the image must be bright enough to distinguish the different features. As an exposure time of 1.6 ms with the default gain value did not provide enough brightness for a robust feature detection, the gain value of the sensor needs to be adjusted too. A higher gain value leads to a brighter image, but also increases the noise in the image, which can disturb the detection of edges (e.g., glint contours). Setting these values is a trade-off between brightness and noisiness. Thus, these values (gamma 220, gain 3) are adjusted to get an image that is bright enough to distinguish the features but low-noise enough to not disturb edges. Next, the frame capture is started and sends the data to our software. When a frame is processed within the camera memory, the captured data are saved in the working memory (RAM) of the PC where they are accessible for the second thread.

Glint detection

When the camera frame is ready, the image is convolved by a Gaussian kernel for denoising. On the denoised image, a 2D-Laplacian filter (Laplacian of Gaussians) is applied to improve the pixel intensity. The size of the kernel for both filters highly depends on the size of the bright points that should be detected and is a trade-off between speed and quality of results. We choose the smallest possible kernel size to achieve the fastest detection. In conjunction with optimizing the brightness of the IR emitters by controlling their voltage and the angle of the IR emitters we can use a 3×3 kernel to detect the glints in the images. As the feature improvement also strengthens other less bright white blobs, the image is thresholded. The thresholding value can

Fig. 2 The user interface offers several functions. Next to the main features like calibration, accuracy evaluation, live gaze and video-based gaze capture, the interface shows important timings for glint and pupil detection

be adjusted in the interface if needed. We found that the differences in brightness between glints and other bright points is big enough to use a fixed thresholding value of 176 on images with brightness normalized to a range of 0 to 255. The thresholding removes less bright white blobs that can disturb the detection of the contours of the brightest blobs—assumed to be the correct glints—or lead to false detections in the next step. After the contour detection on the thresholded image, the centers of the minimum area rectangles of the found contours are passed as glint candidates to the next step. For each glint candidate, the brightness and distance to the surrounding pixels is checked. These values depend on the size of the glints and on the distance between the glints. We ignore reflections—assumed they lie on the sclera—when the mean brightness of the surrounding pixels is above a certain limit. In our system, we designed the locations of the IR emitters that the glints often appear at the border of the iris when a user is at a 60–70-cm distance from the camera. A glint candidate whose surrounding mean brightness is too high is treated

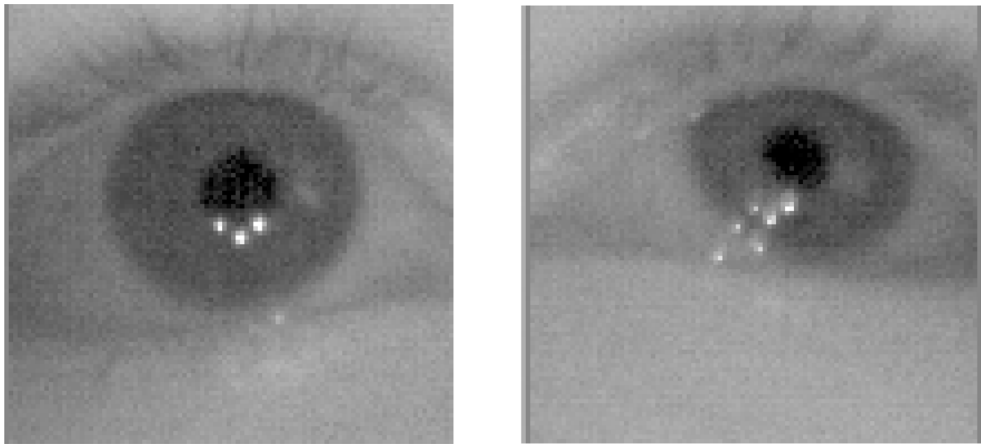

Fig. 3 On the left side image **a**) shows three reflections with only one possible geometry. On the right side image **b**) shows multiple reflections. Only one combination for the searched geometry is entirely positioned on the iris, so we can ignore the remaining as their surrounding brightness is higher

as a reflection of another environmental light source. One problem here is to choose the correct value for the threshold. Low limit values eliminate glints that are on the border between iris and sclera, whereas high limit values create large number of false glints on the sclera area.

In addition to the above procedure, we check the geometric relations of the glints that fit to the LEDs. We know the physical distances between the LEDs: Where two are on the same horizontal axis and one is in the center of them shifted downwards. Thus, we can search for a similar geometry inside the eye image. For example, when two glints lie on the same horizontal axis, $\pm$ 4 pixels buffer vertically, we assume, these are two of the three glints we are searching for. We know that the third glint must lie horizontally in the center of these two (or slightly shifted to one side, depending on the corneal curvature) and is vertically shifted downwards. The buffer and distances between the glints in pixels are empirically defined and fit to the possible sizes of the glints in the whole range where the camera is in focus.

In the case where more than one glint combination is found that fits to our geometry, the combination with the lowest surrounding mean brightness is taken. Due to our LED design, one of the combinations must lie entirely on the pupil or iris. In all other combinations at least one glint must lie on the sclera or on the border to the iris and was not ignored in the steps before. Combinations with such glint have therefore a higher surrounding mean brightness. Figure 3 shows two cases. Image a) shows three reflections that are entirely positioned either on the pupil or on the iris. Their background is dark. We detect these three reflections as our glints as they fit to the geometry of the LEDs.

In Fig. 3 image b) there are more than three reflections and therefore more than one possible combination for the geometry that fits to the LED geometry. As only one combination is positioned entirely on dark background (iris) we can ignore other combinations, because our algorithm detects that the remaining combinations have at least one reflection on the brighter background of the sclera or near the border to it. The surrounding brightness is much higher than for the combination lying on the iris. Once we have a glint combination for each eye, we define a region of interest around each eye and pass a much smaller image (100x100 pixel) to the pupil detection algorithm. These glint combinations are calculated for each incoming frame. Figure 4 shows a summary of the glint detection algorithm. Figure 5 shows a sample image with the detected glints and pupil center. An image of 100x100 pixel compared to the full image size is a good ratio, because the eye is fully covered for the full range of where the camera stays in focus. This image size seems to be typical for other eye trackers such as The Eye Tribe in which the eye image is 100 $\times$ 90 pixel based on our tests.

Pupil detection

The pupil detection algorithm is an implementation of the BORE algorithm from Fuhl, Santini, and Kasneci (2017), which is based on the oriented edge optimization formulation from Fuhl et al. (2017). Oriented edge

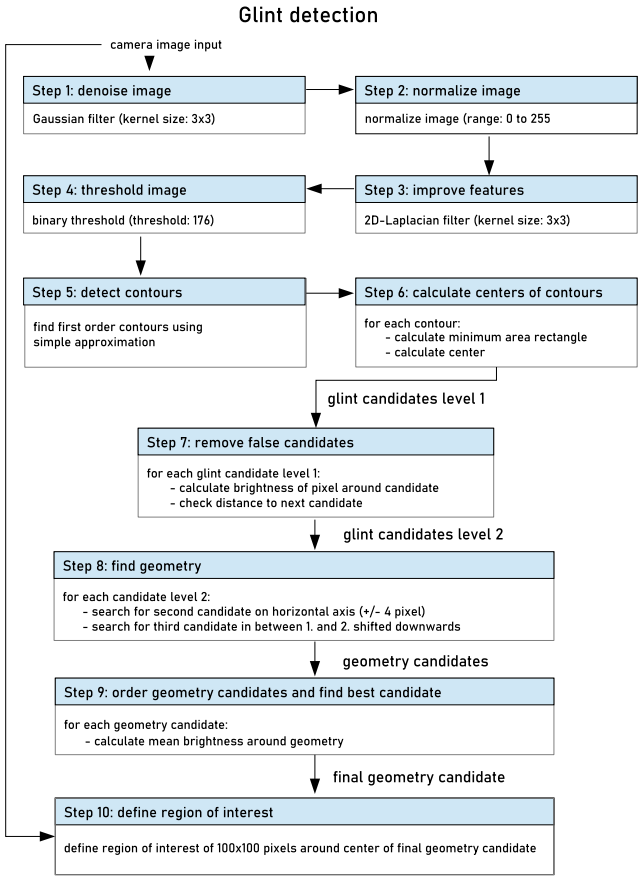

Fig. 4 Summary of the glint detection algorithm showing the single steps

optimization gives a rating for each pixel based on its surrounding edges. In Fuhl et al. (2017), polynomials are used as the pattern of the edges. Each edge with the corresponding orientation to the current image position and the gradient pattern (correct edge), corresponds to a positive evaluation. In contrast, wrongly oriented edges do not count into the evaluation. BORE (Fuhl et al., 2017) is the reformulation of the optimization to circles and ellipses. In the case of circles, different radii (r) are considered for each image position. The current image position corresponds to the center of the circle (p). To scan the oriented edges along a circle, different angles (a) with a fixed increment are considered.

$$argmax_p \int_{r=r_{min}}^{r_{max}} \int_{a=0}^{2\pi} L(\Delta C(p,r,a))\, dr\, da, \tag{1}$$

Equation 1 describes the approach formally. r_{min} and r_{max} are the circle radii. $\Delta C()$ is the oriented edge weight and is either ignored or added to the current evaluation via the evaluation function (L, Eq. 2).

$$L = \begin{cases} inner < outer & 1 \\ else & 0 \end{cases} \tag{2}$$

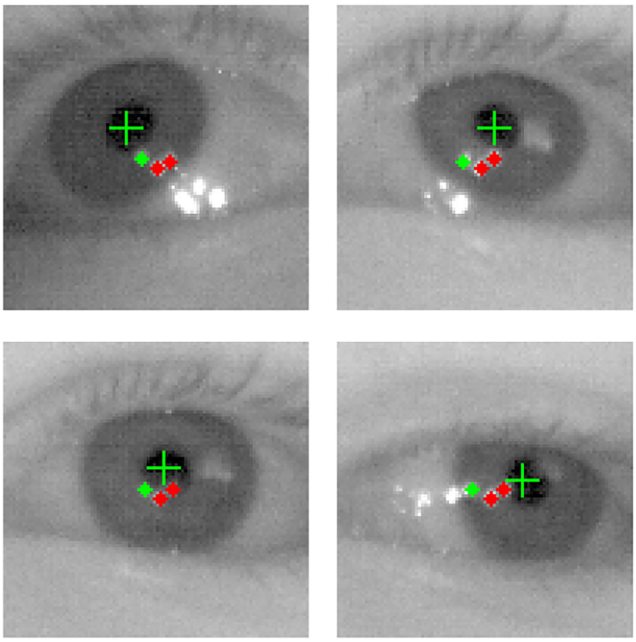

Fig. 5 Sample eye images (with eye looking in different directions) cropped to 100 × 100 pixel. The *green cross* shows the detected pupil center. The *green dot* marks the first detected glint and the *red dots* the other two of the glint combination

In Eq. 2, the inner pixel value is compared with the outer pixel value. After each image position has been evaluated, the maximum is selected as the pupil center. This position corresponds to one pixel in the image which is too inaccurate for low resolution images of the eye, which typically occur in remote eye tracking. To overcome this restriction, the authors Fuhl et al. (2017) provided an extended version with an ellipse fit. The ellipse fit selects the oriented edges which are positively evaluated for this center and calculates an ellipse over these points. As this procedure is computationally very expensive, the authors Fuhl et al. (2017) presented an unsupervised boosting approach for automated person specific training of the algorithm.

To train the algorithm, BORE is given video sequences captured during calibration (without any annotation). After calibration, BORE calculates the pupil centers for all oriented gradients occurring in the training images. Afterwards, all gradients are evaluated by their contribution to the detections. Unimportant gradients are removed from the algorithm sequence based on predefined run-time restrictions. This means that BORE can be adjusted to be faster or more accurate. The algorithm can be used for frame rates up to 1000. Further details regarding the algorithm, can be found in Fuhl et al. (2017).

Gaze estimation

During calibration, the user is presented different screens containing calibration targets. First, the full nine-point

calibration grid is displayed for 1 s. It is followed by a blank white screen and the calibration targets one after another (from top left to bottom right). Presentation time per target is 1500 ms. We discard the first and last 500 ms per target. This is done because we consider the first 500 ms as necessary to maintain a stable fixation on the target. Furthermore, we observed that the users' focus typically decreases after 1000 ms, to explore the space around the calibration point. In order to avoid outliers and decreasing precision in the calibration, we decided to discard that data. We determined these times based on our observation of the data during development of the system.

During the central 500 ms, we capture the position of the calibration target, the detected eye features (shown in Fig. 5), as well as cropped image regions of the eye. The image data produce considerable amount of data that need to be held in memory. Thus, we implemented a temporary buffer to reserve the required memory before recording. However, it needs to be copied to a global queue that is accessible to different threads after each calibration target. This copy and reset of the temporary buffer happens during the last 500 ms of stimulus presentation.

In the next step, we use the cropped eye images to fine-tune the pupil-detection algorithm. The pupil detection learns which pupil sizes are likely to occur during recording. This training optimizes the processing speed later. The pupil center and glint locations for each calibration point are then detected and passed on to fitting of a 3D eye model. All calibration targets, the location of the detected features, and the percentage of images when the tracker could not detect the necessary features are reported to a file.

For gaze estimation, converting image feature locations into a 3D gaze ray, we use the 3D eye model described by Guestrin and Eizenman (2006). Guestrin specifies models for any number of light sources and cameras. We employ the version with three light sources and one camera as a good trade-off between accuracy and cost-efficiency. This way, we do not require to synchronize multiple high-speed cameras. One disadvantage of this configuration of the model is that, with one camera, the model is not very robust to head movements and the optimization needs considerable computation time.

The fitting process is split into three parts: At first, the optical axis of the eye is reconstructed. Accurate measurement of the physical distances between camera, LEDs, and screen are required for this, as all coordinates need to be transformed to a common world coordinate system. The cornea center is then calculated as an intersection of planes spanned by the camera location, the light source location, and the location of the reflection at the cornea surface. This problem can be solved, assuming a spherical cornea and utilizing the unprojection of the glint image feature on the camera sensor towards 3D. Afterwards, the pupil location can be determined in a similar way and the optical axis is described as the vector between cornea center and pupil center.

In the second part, the deviation between optical and visual axis needs to be determined (illustrated in Fig. 6). This deviation differs between humans and a calibration is needed to estimate the offset. The visual axis is defined by the nodal point of the eye and the center of the fovea (region of highest acuity on the retina). Typically,

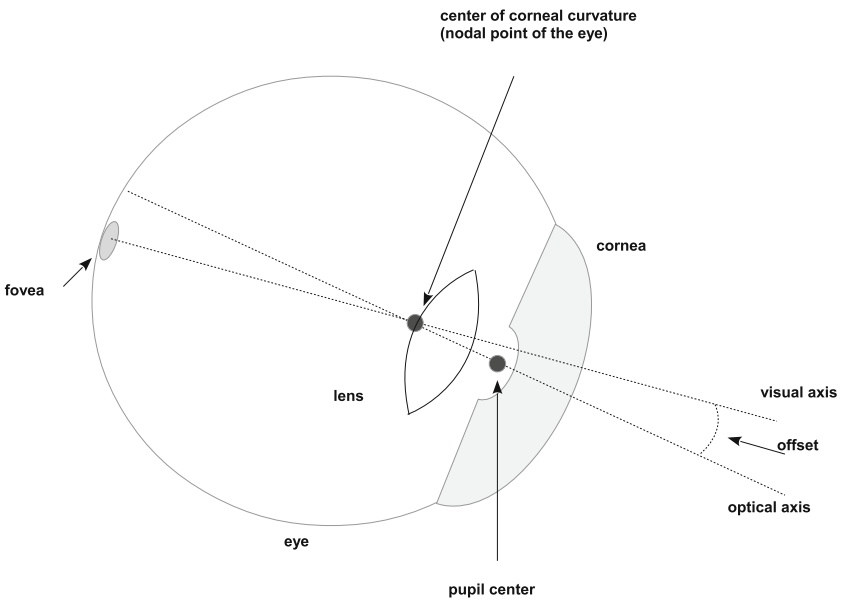

Fig. 6 Illustration of the 3D eye model. The optical axis of the eye is defined by two points, the pupil center and center of corneal curvature/nodal point of the eye. The visual axis by the fovea and the center of corneal curvature/nodal point of the eye. During calibration, these axes need to be reconstructed in order to estimate the gaze

this offset is about 1.5° below, and between 4 and 5° temporally to the intersection point of the optical axis on the retina. The whole model is described as an optimization problem where, depending on the camera and light setup, a differential system needs to be solved for up to 13 unknown variables simultaneously. The amount of unknown variables and therefore the model complexity shrinks with more light sources and especially more cameras. As soon as the visual axis is reconstructed, the model can be used to estimate the gaze (third part).

After the calibration is calculated, we re-run the calibration calculations to remove possible outliers. Therefore, we apply a filter that removes all measurements that exceed the Euclidean distance to their respective calibration target by more than the average half plus/minus the standard deviation.

Signal quality

As humans are usually moving slightly without a chin rest and the resolution of the camera does not allow higher-quality images, leading to a larger scene coverage by each pixel, algorithms for detection can be noisy, as they can only detect pixel-wise. With much more and smaller pixels, this noise can be minimized as small light changes are spread on smaller areas. This is problematic when finding contours in an image. As light changes on pixels at the border of a contour can lead to a fluctuating contour. Another way is to filter the signal. We filter the pupil center signal by applying a moving-average filter over 20 consecutive samples (windows size: 20 samples). We apply this filter to take the positions of all 20 samples of the current window into account to correct the position of the current sample. This way, we can remove noise in the signal. This filter does not remove any sample, but rather optimizes the position of the current sample by smoothing it with the positions of the previous samples. When the algorithm receives a new pupil center signal, we test if the new signal is in the range of one standard deviation of the current window. If so, the window moves one sample further by removing the oldest sample. We add the new value (PC_M) and calculate the mean value of the current window by solving Eq. 3 and use that as the new pupil center signal (PC_{New}). If the new signal is outside of the range of one standard deviation of the window, we assume a movement of the pupil signal was not noise, but rather a real movement and take the new value as a new sample (empty values $PC_0, \cdots, PC_M$).

$$PC_{New} = \frac{1}{n}\sum_{i=0}^{n-1} PC_{M-i} \tag{3}$$

As the pupil signal can change rapidly and a filtering of the signal must be fast and should not falsify saccade detections in later steps, we chose this moving-average filter for the pupil center signal. For the glints we use a Kalman filter (Bishop, Welch, & et al. 2001) to minimize the noise. The Kalman filter can be problematic for the pupil signal, as pupil movements can be very fast and create far jumps in the image. A Kalman filter would smooth these jumps out

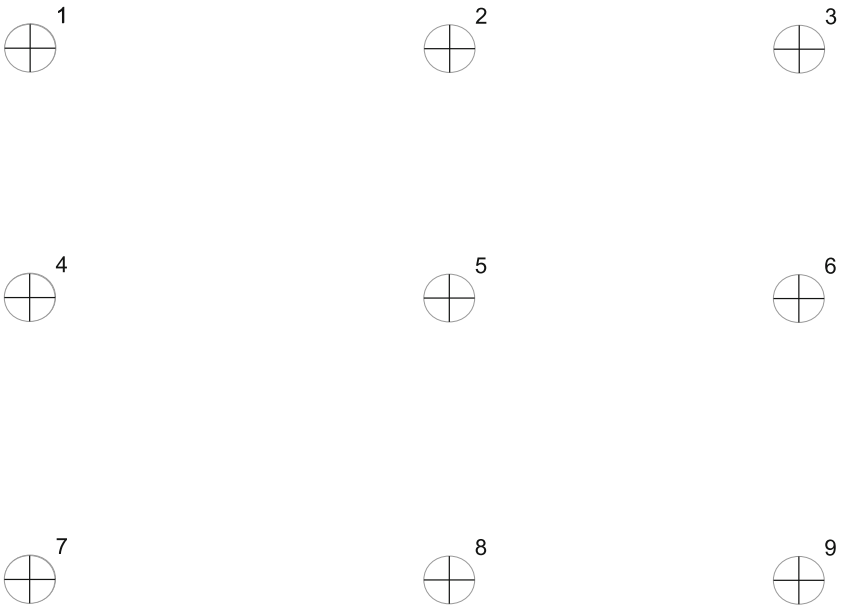

Fig. 7 We used a nine-point calibration pattern for calibration as well as for estimation and showed the pattern in full screen. In both procedures, we showed each point sequentially

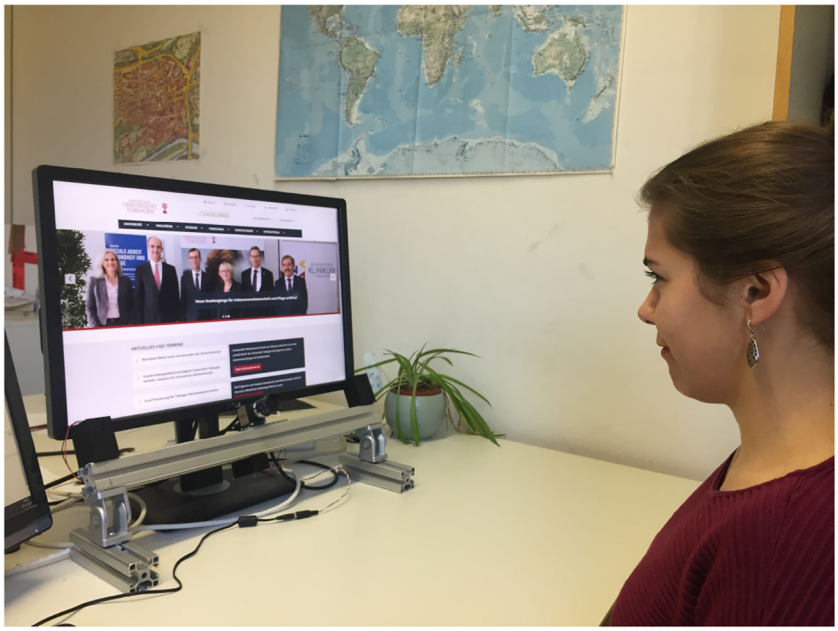

Fig. 8 We used a aluminium profile stand so the eye tracker was positioned close to the screen. We placed it at a distance of 70 cm to the user to provide a common viewing position for the participants

and create a delay, which is not acceptable as saccades can not be detected correctly anymore. As the glints do not move very far on the cornea, the Kalman filter is optimal for stabilizing the position of the glints. We use this filter as slight movements of the participant already introduce noise in the glint signal. Due to the small resolution of the camera image, one pixel covers a larger area and slight movements can lead to flickering of the pixels at the boundaries of the glints. In our tests, the Kalman filter reduced this noise much better than a moving-average filter. We configured the Kalman filter with noise variances of smaller than one pixel in horizontal and vertical axis.

Method

We conducted a study to evaluate the accuracy and precision of the eye tracker. Our intention here was to focus on pupil and glint detection accuracy. We report accuracy and precision based on new unseen data, as reporting the calculated values from the calibration would only show how well the model fits.

For calibration and estimation, the same nine-point calibration pattern (Fig. 7) was shown to the user. The calibration points were 1 degree in size and distanced 12 degrees on the horizontal axis and 7 degrees on the vertical axis. To measure the system performance, we measured pixel location variations (standard deviation) for the glint and pupil center. Moreover, the gaze accuracy was defined as the average distance between the real target position and the estimated gaze position. The precision was defined as standard deviation of the estimated gaze position samples.

Apparatus

Figure 8 shows the experimental setup of our hardware and Fig. 5 shows a sample frame of eyes being tracked by our software. In this step, participants where asked to sit in front of the screen with a distance of about 70 cm. The resolution of the screen was 1920 × 1200 with dimensions of 52 × 33 cm. As we wanted to have as few restrictions as possible, we did not use a chin rest (allowing small head movements). We choose a standard distance of 70 cm (Holmqvist et al., 2011). The connection between camera and PC over USB 3 was stable with no failed transfers, a frame delay time of 0.7 ms and a bandwidth usage of about 220 MB/s (max. interface bandwidth is 420 MB/s).

Participants and procedure

Nineteen participants (18 male and one female) with average age of 29 years (SD = 9.4) were recruited for the study. All participants were students of the computer science department or part of the staff and could work on computer without glasses.[2] Participants had various

[2]We tested our system with glasses, but as glints extremely grow on glasses—cover big parts of the eye—we could not detect the glints/pupil points reliably. Further investigation is needed in future work.

ethnicity namely, Caucasian coming from Europe or North America, Middle-East Asian (Arabic), and Far-East Asian. All participants had either black, brown, brown-green, gray-blue, or blue eyes.

After signing a consent form, participants were asked to watch the nine-point calibration pattern two times with a maximum 30-s break between for calculating the eye model. We used the first set of the data for the calibration and second set for the accuracy and precision report. This way, we avoid testing on the optimized data set from the calibration values, which is often reported from commercial eye trackers (e.g., Tobii Studio software). This value is often low, because the same data used for calibration are optimized and used again for testing.

Results

The current system uses 40 % of all CPUs during training and 13 % while capturing and only 1.0% of RAM. When the system is running only for calibration and gaze estimation, the working memory was less than 170 megabytes. The usage of the working memory during video playback highly depends on the size and length of the video.

Runtime measurements

Table 2 shows that the system on average can compute 723 frames per second. As processing of one frame takes 1.38 ms on average. In extreme cases, when the eye features are hard to detect, the system can still handle 668 frames per second. The frames where our algorithms did not detect eye features—when certain interim variables are not filled with proper data—were skipped (labeled as undetected eye). The glint detection for example will skip the frame if the contour list is empty.

We also report the number of times our algorithms were not able to detect glints or pupils. Over all participants the right eye was detected in 100% of the frames and the left eye in 99.35%.

Table 2 Speed measurements in milliseconds. The total time includes time to detect glint, detect pupil center, and presenting a stimulus (calibration point). We measured the different run times separately and calculated the values over 8500 example random data

Algorithm	Runtime	Std. Dev.	FPS Avg.
Glint	0.46	0.02	2169
Pupil	0.78	0.058	1277
System	1.38	0.115	723

Quality of results

Table 3 shows the precision of the glint and pupil detections based on the standard deviation of the pixel locations. The pupil deviates 0.68 pixel horizontally and 0.26 pixel vertically. The glint values have deviation of 0.67 pixel horizontally and a deviation of 0.29 pixel vertically. In both eyes, the deviation is 0.67 pixel on horizontal and 0.27 pixel on vertical axis. For measuring the pixel deviation, we used 4500 sample detections on different calibration points and different participants.

Outliers in the detection of the features indicate an uncertainty of the algorithm. The origin of these uncertainties is, for example, the position of the glints on the cornea. Different positions, especially when they are near the border to the sclera, lead to variations in size and sometimes even in shape. As the shape changes, surrounding pixels can gain brightness and therefore go over the threshold limit. Here, the center of the glint can slightly move into the direction of the newly added bright pixel. As the quality of detection has an high impact on the gaze estimation, a stable and accurate detection is important for high precision. With a higher-resolution image, the impact of a changing pixel location can be reduced.

Accuracy and precision

For the purpose of comparison with the typical report of accuracy and precision, we calculated the average values over all participants for accuracy and precision with new data. We measured the accuracy based on the Euclidean distance from the sample estimated gaze points to the original calibration point on the screen transformed to visual angle. For precision, we used the standard deviation (SD) as the metric. For calculating the accuracy for a single calibration point, we first calculated the Euclidean distances for each of the sampled gaze points $P_1(x_1, y_1) \cdots P_n(x_n, y_n)$ to the corresponding calibration point $C(x_c, y_c)$ shown in Eq. 4. As we report our accuracy in visual angle, we transformed the Euclidean distances to visual angle by solving Eq. 5. To

Table 3 The deviation of the detections in pixel for vertical axis is less then half of the deviation in horizontal axis. For both pupil and for glint the deviation is smaller than 1 pixel

Feature	Horizontal (pixel)	Vertical (pixel)
Pupil	0.68	0.26
Glints	0.67	0.29
Total	0.67	0.27

transform the results of Eq. 5 from radians to degree we used Eq. 6. For calculating the accuracy, we took the average of all distances in visual angle of the sampled gaze points to the calibration point by solving Eq. 7.

$$dist(C, P) = \sqrt{(x_P - x_C)^2 + (y_P - y_C)^2} \tag{4}$$

$$V_{rad}(C, P) = 2 \cdot atan\left(\frac{dist(C, P)}{2 \cdot d}\right) \tag{5}$$

$$V_{deg}(C, P) = \left(\frac{V_{rad}(C, P)}{2 \cdot \pi}\right) \cdot 360 \tag{6}$$

$$Accuracy_C = \frac{1}{n} \cdot \sum_{i=1}^{n} (V_{deg}(C, P_i)) \tag{7}$$

$$Precision_C(SD) = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (P_i - \bar{P})^2} \tag{8}$$

We calculated the standard deviation of all sample points n for one calibration point C by solving Eqs. 4, 5, 6 and 8 with $P = P_i(x_i, y_i)$ and $\bar{P} = P_{i+1}(x_{i+1}, y_{i+1})$.

The accuracy of the gaze estimation with the same data we captured during calibration is typically about 0.5 degrees. Although, eye-tracking companies tend to use the optimized gaze data from calibration (e.g., Tobii studio software) and the corresponding accuracy values to report their tracker performance, we reduce the effect of optimal fitted data by not using them for accuracy and precision reports. Rather, we tested the calibration model on new unseen data. We show the gaze estimation errors in Table 4 in degree of visual angle. On average, over all points and all participants, the system has an accuracy of 0.98 degrees. The precision (standard deviation) is 0.38 degrees. The best accuracy for a single participant was at 0.5 degrees. The worst accuracy for a single participant was 1.4 degrees. The best precision (standard deviation) we achieved was at 0.26 degrees. Worst precision peaked at 0.7 degrees.

Table 4 Best, average, and worst cases of accuracy and precision of gaze estimation (in degrees) based on all participant data. The average is based on the Euclidean distance to the target points. We calculated the precision as standard deviation (SD)

Metric	Best	Avg.	Worst
Accuracy	0.51	0.98	1.4
Precision	0.26	0.38	0.70

Discussion

Our work demonstrates a real-time high-speed eye tracker in a remote-based setup. Additionally, it detects glint, pupil center, and later estimate gaze direction within 2 ms. As the delay of 0.7 ms caused by the camera capturing the image was constant, we could include that value for our gaze synchronization. This perspective on eye-tracking systems stands in stark contrast to the common low-cost head-mounted eye trackers that do not need to optimize algorithms for extreme fast detection as both resolution and speed of cameras are low. Our focus here has been a pupil and glint detection algorithm in a high-speed and low-cost system. As such, we used a single camera solution. To optimize the feature detection quality, we plan to use higher-resolution images as well as a multiple-camera setup. Another improvement can be achieved by adding a second camera with the pixel shift minimized by synchronizing the detections of both cameras using an epipolar geometrical method.

For pupil detection, we proposed an unsupervised boosting technique to select only the most important positions on the circle outline for computation to achieve an extremely fast (0.78 ms) detection. As we see from the evolution, our glint detection performed with only 0.48 pixels detection error. Thus, we achieved significant results in speed of less than 2 ms with only about 0.33% of data loss. To the knowledge of the authors, this is the first time such a high-speed real-time remote eye-tracking system with low data loss is offered. For example, the study by Hiroyuki (Sogo, 2013) shows that EyeLink (a commercial eye tracker) has 6.3% (SD=1.3%) of undetected gaze data. This value is much higher for other low-cost eye trackers (e.g., 15.5% (SD=9.8%) for the Hiroyuki (Sogo, 2013) tracker).

For the pupil detection, we had only 0.47 pixels error on average. Although this value is very low, it caused errors in the gaze estimation and seemed unavoidable. We believe our pupil and glint detection algorithm would perform better on higher-resolution camera images (for example, in our setup after detecting the glints, the eye region was only 100×100 pixels). If getting an expensive camera is not possible, we then suggest to scale up the region around the pupil center and recalculate it to achieve higher sub-pixel accuracy.

One limitation of our system is the gaze estimation technique in combination with the camera setup. We assume that (1) our gaze estimation error is affected by head movements as we did not use a chin rest to provide an environment with as few restrictions as possible and (2) that the optimization process of the 3D eye-model parameters is prone to errors as optimizing 13 unknown variables simultaneously is a highly complex task. With the chosen resolution, even small changes in the shape

(contour) of the features seem to have a big impact on the precision as a lower resolution means a larger scene coverage for each pixel and therefore a higher sensitivity to light changes. These limitations can be overcome by using multiple cameras with higher resolution.

We plan to use a two camera-based 3D calibration technique (Hansen & Ji, 2010) for our system in the future. We also did not use an artificial eye to report the accuracy, which is common for commercial eye trackers. The real accuracy usually varies between 0.3 and 2 degrees (Holmqvist et al., 2011) like the Tobii T60XL has an accuracy of 1.27 (Morgante, Zolfaghari, & Johnson, 2012) but there are also a lot of trackers with higher values (e.g., Tobii X2-30 with 2.46 degree, (Clemotte et al., 2014) or Smart Eye Pro with 2.4 degrees (Funke et al., 2016). Based on Hansen and Ji (2010), an accuracy of 2 degrees is typical for remote eye trackers. Nevertheless, a 3D model-based calibration technique with multiple cameras, like Guestrin and Eizenman (2006), is more suitable.

Another aspect to consider is related to the stimuli and gaze synchronization method. Stimulus synchronization is critical in high-speed eye-tracking systems. Choosing the right camera could be a solution to this challenge, as often (including our system) industrial cameras allow time stamping for external events directly into the camera. In the future, we also plan to include this feature in our system.

The source code of the system is available under https://github.com/hospbene/RemoteEye. The study was not formally preregistered and the data have not been made available on a permanent archive, but requests can be sent via e-mail to the lead author at benedikt.hosp@uni-tuebingen.de.

References

Andersson, R., Nyström, M., & Holmqvist, K. (2010). Sampling frequency and eye-tracking measures: How speed affects durations, latencies, and more. *Journal of Eye Movement Research*, *3*(3), 1–12.

Bishop, G., Welch, G., & et al. (2001). An introduction to the Kalman filter. *Proc of SIGGRAPH, Course*, *8*(27599–3175), 59.

Canare, D., Chaparro, B., & He, J. (2015). A comparison of gaze-based and gesture-based input for a point-and-click task. In *International conference on universal access in human-computer interaction*, (pp. 15–24): Springer.

Clemotte, A., Velasco, M., Torricelli, D., Raya, R., & Ceres, R. (2014). Accuracy and precision of the Tobii x2-30 eye-tracking under non ideal conditions. *Eye*, *16*(3), 2.

Coyne, J., & Sibley, C. (2016). Investigating the use of two low-cost eye tracking systems for detecting pupillary response to changes in mental workload. In *Proceedings of the Human Factors and Ergonomics Society annual meeting*, (Vol. 60, pp. 37–41). Los Angeles: SAGE Publications Sage CA.

Dera, T., Boning, G., Bardins, S., & Schneider, E. (2006). Low-latency video tracking of horizontal, vertical, and torsional eye movements as a basis for 3D of realtime motion control of a head-mounted camera. In *IEEE international conference on systems, man and cybernetics, 2006. SMC'06*, (Vol. 6, pp. 5191–5196): IEEE.

Ebisawa, Y. (1970). Unconstrained pupil detection technique using two light sources and the image difference method. *WIT Transactions on Information and Communication Technologies*, *15*, 11.

Ebisawa, Y. (1998). Improved video-based eye-gaze detection method. *IEEE Transactions on Instrumentation and Measurement*, *47*(4), 948–955.

Farivar, R., & Michaud-Landry, D. (2016). Construction and operation of a high-speed, high-precision eye tracker for tight stimulus synchronization and real-time gaze monitoring in human and animal subjects. *Frontiers in systems neuroscience*, *10*, 73.

Fuhl, W., Geisler, D., Santini, T., Appel, T., Rosenstiel, W., & Kasneci, E. (2018). CBF: Circular binary features for robust and real-time pupil center detection. In *Proceedings of the 2018 ACM symposium on eye tracking research & applications* (p. 8). ACM.

Fuhl, W., Kübler, T., Sippel, K., Rosenstiel, W., & Kasneci, E. (2015). Excuse: Robust pupil detection in real-world scenarios. In *International conference on computer analysis of images and patterns*, (pp. 39–51): Springer.

Fuhl, W., Santini, T., & Kasneci, E. (2017). Fast and robust eyelid outline and aperture detection in real-world scenarios. In *2017 IEEE Winter conference on applications of computer vision (WACV)*. https://doi.org/10.1109/WACV.2017.126, (pp. 1089–1097).

Fuhl, W., Santini, T., Kasneci, G., Rosenstiel, W., & Kasneci, E. (2017). PupilNet v2. 0: convolutional neural networks for CPU based real time robust pupil detection. arXiv:1711.00112.

Fuhl, W., Santini, T. C., Kübler, T., & Kasneci, E. (2016). Else: Ellipse selection for robust pupil detection in real-world environments. In *Proceedings of the ninth biennial ACM symposium on eye tracking research & applications*, (pp. 123–130): ACM.

Fuhl, W., Tonsen, M., Bulling, A., & Kasneci, E. (2016). Pupil detection in the wild: An evaluation of the state of the art in mobile head-mounted eye tracking. *Machine Vision and Applications*, *27*, 1275–1288.

Funke, G., Greenlee, E., Carter, M., Dukes, A., Brown, R., & Menke, L. (2016). Which eye tracker is right for your research? performance evaluation of several cost variant eye trackers. In *Proceedings of the Human Factors and Ergonomics Society annual meeting*, (Vol. 60, pp. 1240–1244). Los Angeles: SAGE Publications Sage CA.

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*(6), 1124–1133.

Hansen, J. P., Ahmad, Z., & Mardanbegi, D. (2014). Gaze interactive building instructions. In *Interaction design and children*.

Hansen, J. P., Alapetite, A., MacKenzie, I. S., & Møllenbach, E. (2014). The use of gaze to control drones. In *Proceedings of the symposium on eye tracking research and applications*, (pp. 27–34): ACM.

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*(3), 478–500.

Hennessey, C., & Lawrence, P. (2009). Noncontact binocular eye-gaze tracking for point-of-gaze estimation in three dimensions. *IEEE Transactions on Biomedical Engineering*, *56*(3), 790–799.

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., & Van de Weijer, J. (2011). Eye tracking: A comprehensive guide to methods and measures. OUP Oxford.

Jbara, A., & Feitelson, D. G. (2017). How programmers read regular code: A controlled experiment using eye tracking. *Empirical Software Engineering*, *22*(3), 1440–1477.

Kangas, J., Akkil, D., Rantala, J., Isokoski, P., Majaranta, P., & Raisamo, R. (2014). Using gaze gestures with haptic feedback on

glasses. In *Proceedings of the 8th Nordic conference on human-computer interaction: fun, fast, foundational*, (pp. 1047–1050): ACM.

Kassner, M., Patera, W., & Bulling, A. (2014). Pupil: an open-source platform for pervasive eye tracking and mobile gaze-based interaction. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: adjunct publication*, (pp. 1151–1160): ACM.

Li, D., Babcock, J., & Parkhurst, D. J. (2006). Openeyes: A low-cost head-mounted eye-tracking solution. In *Proceedings of the 2006 symposium on eye tracking research & applications*, (pp. 95–100): ACM.

Long, X., Tonguz, O. K., & Kiderman, A. (2007). A high-speed eye tracking system with robust pupil center estimation algorithm. In *Engineering in medicine and biology society, 2007. EMBS 2007. 29th annual international conference of the IEEE*, (pp. 3331–3334): IEEE.

Mann, D. T., Williams, A. M., Ward, P., & Janelle, C. M. (2007). Perceptual-cognitive expertise in sport: A meta-analysis. *Journal of Sport and Exercise Psychology*, *29*(4), 457–478.

Morgante, J. D., Zolfaghari, R., & Johnson, S. P. (2012). A critical test of temporal and spatial accuracy of the Tobii t60xl eye tracker. *Infancy*, *17*(1), 9–32.

Morimoto, C. H., Koons, D., Amir, A., & Flickner, M. (2000). Pupil detection and tracking using multiple light sources. *Image and Vision Computing*, *18*(4), 331–335.

Murugaraj, B., & Amudha, J. (2017). Performance assessment framework for computational models of visual attention. In *The international symposium on intelligent systems technologies and applications*, (pp. 345–355): Springer.

Ooms, K., Dupont, L., Lapon, L., & Popelka, S. (2015). Accuracy and precision of fixation locations recorded with the low-cost eye tribe tracker in different experimental setups. *Journal of Eye Movement Research*, *8*(1), 1–24.

Parada, F. J., Wyatte, D., Yu, C., Akavipat, R., Emerick, B., & Busey, T. (2015). Experteyes: Open-source, high-definition eyetracking. *Behavior Research Methods*, *47*(1), 73–84.

Ramos, G., Hanada, R., Da Graça, C., Pimentel, M., & Teixeira, C. A. (2017). A word-prediction eye-typing approach for Brazilian Portuguese entries using geometric movements. In *Proceedings of the 35th ACM international conference on the design of communication*, (p. 35): ACM.

Rodrigue, M., Son, J., Giesbrecht, B., Turk, M., & Höllerer, T. (2015). Spatio-temporal detection of divided attention in reading applications using EEG and eye tracking. In *Proceedings of the 20th international conference on intelligent user interfaces*, (pp. 121–125): ACM.

San Agustin, J., Skovsgaard, H., Mollenbach, E., Barret, M., Tall, M., Hansen, D. W., & Hansen, J. P. (2010). Evaluation of a low-cost open-source gaze tracker. In *Proceedings of the 2010 symposium on eye-tracking research & applications*, (pp. 77–80): ACM.

Santini, T., Fuhl, W., Geisler, D., & Kasneci, E. (2017). EyeRecToo: open-source software for real-time pervasive head-mounted eye tracking. In *VISIGRAPP (6: VISAPP)* (pp. 96–101).

Santini, T., Fuhl, W., & Kasneci, E. (2018). PuRe: Robust pupil detection for real-time pervasive eye tracking. *Computer Vision and Image Understanding*, *170*, 40–50.

Santini, T., Niehorster, D. C., & Kasneci, E. (2019). Get a grip: slippage-robust and glint-free gaze estimation for real-time pervasive head-mounted eye tracking. In *Proceedings of the 11th ACM symposium on eye tracking research & applications* (p. 17). ACM.

Sari, F. N., Santosa, P. I., & Wibirama, S. (2017). Comparison expert and novice scan behavior for using e-learning. In *Second international workshop on pattern recognition*. International society for optics and photonics, (Vol. 10443, p. 104430e).

Schneider, E., Villgrattner, T., Vockeroth, J., Bartl, K., Kohlbecher, S., Bardins, S., ..., Brandt, T. (2009). Eyeseecam: An eye movement-driven head camera for the examination of natural visual exploration. *Annals of the New York Academy of Sciences*, *1164*(1), 461–467.

Sogo, H. (2013). Gazeparser: An open-source and multiplatform library for low-cost eye tracking and analysis. *Behavior Research Methods*, *45*(3), 684–695.

Stengel, M., Grogorick, S., Eisemann, M., Eisemann, E., & Magnor, M. A. (2015). An affordable solution for binocular eye tracking and calibration in head-mounted displays. In *Proceedings of the 23rd ACM international conference on multimedia*, (pp. 15–24): ACM.

Świrski, L., Bulling, A., & Dodgson, N. (2012). Robust real-time pupil tracking in highly off-axis images. In *Proceedings of the symposium on eye tracking research and applications*, (pp. 173–176): ACM.

Zhang, X. B., Fan, C. T., Yuan, S. M., & Peng, Z. Y. (2015). An advertisement video analysis system based on eye-tracking. In *2015 IEEE international conference on smart city/socialcom/sustaincom (smartcity)*, (pp. 494–499): IEEE.

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