Webcam Eye Tracking for Desktop and Mobile Devices: A Systematic Review

Melanie Heck
University of Stuttgart, Germany
melanie.heck@ipvs.uni-stuttgart.de

Viola Deutscher
University of Mannheim, Germany
viola.deutscher@uni-mannheim.de

Christian Becker
University of Stuttgart, Germany
christian.becker@ipvs.uni-stuttgart.de

Abstract

Building the Internet of Behaviors (IOB) obviously requires capturing human behavior. Sensor input from eye tracking has been widely used for profiling in market research, adaptive user interfaces, and other smart systems, but requires dedicated hardware. The wide spread of webcams in consumer devices like phones, notebooks, and smart TVs has fostered eye tracking with commodity cameras. In this paper, we present a systematic review across the IEEE and ACM databases – complemented by snowballing and input from eye tracking experts at CHI 2021 – to list and characterize publicly available software for webcam eye tracking that estimate the point-of-regard with no additional hardware. Information from articles was supplemented by searching author websites and code repositories, and contacting authors. 16 eye trackers were found that can be used. The restrictions regarding license terms and technical performance are presented, enabling developers to choose an appropriate software for their IoB application.

Keywords: gaze estimation, webcam eye tracking, accuracy, sampling rate, systematic review

1. Introduction

Intelligent applications such as personalized YouTube playlists and product recommendations on Amazon that respond to the users' behavior enhance digital services by connecting to the users on a personal level. Eye tracking has been proven to be useful for determining users' attitude and attention, and has therefore been used in market research, adaptive information systems, and, recently, entertainment. Although prices for dedicated eye tracking devices have dropped, they are still not widespread. On the other hand, webcams have been integrated into many devices such as mobile phones, tablets, and notebooks.

As a result, researchers have developed solutions that use images from a webcam or the front-facing camera of a mobile device and coined them *webcam eye tracking*. The idea of webcam eye tracking is not new, but the accuracy and sampling rate of available solutions has so far not been competitive with even entry-level commercial eye trackers like the Tobii Pro Nano[1]. A reasonable sampling rate is necessary to distinguish between *fixations* at a specific point on the display and the *saccade* transitions between them.

In a previous literature, Hansen and Ji (2010) categorized eye and gaze detection methods and identified research trends. However, the article does not describe or report the performance of particular working solutions. Ferhat and Vilariño (2016) identified more than 60 gaze estimation solutions and critically noted that only a small fraction of the source codes have been made publicly available. The authors list seven open source eye trackers, though some are no longer supported. The retention of the source code not only slows down the development of better eye trackers, but also makes them inaccessible to researcher and practitioners not wanting to engineer their own, but who instead are looking for a working solution as an enabling tool for another application. More recent reviews make clear how much research has since progressed. The systematic review by Shehu et al. (2021) lists 17 open-source eye trackers – though multiple are deprecated or not operational with conventional cameras. Cheng et al. (2021) observed an almost exponentially increasing number of articles presenting new gaze estimation techniques between 2015 and 2021. This resulted in a myriad of feature extraction methods, neural network architectures, and calibra-

[1] Tobii Pro Nano: https://www.tobiipro.com/product-listing/nano/.

URI: https://hdl.handle.net/10125/103459
978-0-9981331-6-4

tion approaches. Since the source code of most reviewed systems is not public, the authors implemented a selection and compared their accuracy on different datasets. To the best of our knowledge, their article is the only existing work that compares the performance of multiple non-commercial eye tracking software and thus gives a benchmark to third-party developers. It focuses on appearance-based gaze estimation with deep learning, which is only a subset of the available techniques. Yet, already within this narrow field, the design options are so numerous and complex that non-experts are unlikely to succeed in implementing the systems based only on the specifications from the research articles.

The *goal* of this systematic review is thus to identify eye tracking software that can be readily used by others. It *targets* developers and researchers who are not necessarily eye tracking experts. Unlike previous reviews, it is not our objective to compare gaze estimation methods and theories. Instead, we create a catalog of 16 working software solutions from which developers and researchers can select the most appropriate option depending on the requirements of their projects.

In behavioral research, a high sampling rate and accuracy are essential for behavioral research to correctly classify eye movement (Gibaldi et al., 2017). The latter is typically conditional on an often lengthy eye tracker calibration. Yet a lengthy calibration limits the usability of eye tracking for HCI applications (Drewes et al., 2019). We thus formulate three research questions:

- **RQ1:** For what runtime environments and terms of use are webcam eye trackers available?
- **RQ2:** How does the model initialization and personal calibration affect usability?
- **RQ3:** What accuracy and sampling rate can application developers expect?

To answer these questions, we conducted a systematic review according to the PRISMA guidelines (Liberati et al., 2009) with a qualitative result analysis.

2. Method

2.1. Search procedure

Information sources: Articles were identified by searching electronic databases and scanning reference lists of included articles. The search was applied to the ACM Guide to Computing Literature (1951 – present) and IEEE Xplore (1929 – present) databases, which we identified as artifact oriented libraries. The following search string was used: ("eye tracking" OR "gaze tracking" OR "gaze estimation") AND ("device camera" OR "built-in camera" OR "webcam" OR "web cam*"). No restrictions on language, publication date, or publication type were set. The last search was run on May 3, 2022. The set of identified eye trackers was extended by consulting with eye tracking experts during the EMICS workshop (Wang et al., 2021) held on May 14, 2021 at the ACM CHI conference. The search strategy was discussed with 15 independent reviewers. Agreement was reached that the relevant keywords were included and most systems that were not found through the database search were unpublished or not listed in the databases. Thus, no modification to the initial search strategy was made.

Selection of relevant systems: Stand-alone implementations (excluding extensions or wrappers for other systems) of software for estimating the point-of-regard (PoR) – i.e., the position on the computer display at which that the gaze is directed – were considered. Software of any release date and status, including published and unpublished solutions, were included if the following criteria were met:

- The PoR is estimated as coordinates of the looked at screen position. Excludes software that estimates the gaze direction or broader gaze region.
- The code or interface for integration into an arbitrary application is publicly available. Excludes methods without published code or API for integration into an arbitrary application.
- Runs on desktop (laptop, PC with monitor) or mobile devices (phone, tablet) without external HW. Excludes head-mounted devices (HMDs) and systems using multiple cameras or special HW.
- Targets healthy adults. Excludes eye trackers for children or users with motor impairments, as those are typically fine-tuned to the disparate oculomotor properties of children (e.g., Cui et al., 2017), prioritize robustness over accuracy (e.g., Karamchandani et al., 2015), or use external HW to provide robustness (e.g., Hughes et al., 2017).

Eligibility of the articles from electronic databases was assessed by two independent reviewers. Disagreements were resolved by collectively scanning and discussing articles in dispute until a consensus was reached. The search provided 1808 articles. The steps taken to identify the final set of eye trackers can be retraced via the replication package[2]. After adjusting for duplicates, 1785 articles remained. Of these, 1669 articles were discarded after reviewing the title and abstract because it appeared that they did not meet the criteria.

The full text of the remaining 116 articles was examined. If no link to the source code was provided, author websites and code repositories related to the authors, system, or article were searched to obtain a comprehen-

[2]Replication package: https://www.ipvs.uni-stuttgart.de/institute/team/Heck-00002/

sive collection of publicly available solutions. 106 articles did not meet the inclusion criteria. Of those, 13 required external hardware to be integrated into HMDs or used multiple cameras, depth sensors, or external light sources. 6 articles proposed functionalities for existing eye trackers that can not be used as stand-alone solutions. In 26 cases, the PoR was not calculated. Instead, the objective was to detect and track the eye, estimate the gaze direction, or measure attention to a broad screen area. For 61 articles, no source code was found, either because the system had not yet been implemented, or because the authors described the implementation without sharing the source code. By comparing author names and source code references, two of the remaining 10 articles were identified as publications of the same system. Since they report complementary performance indicators, both were included in the systematic review.

By checking the references of included articles, 4 more eye trackers were found in unpublished sources and articles not indexed by the searched databases or not containing any of the search terms related to camera.

Additionally, 3 eye trackers that met the criteria for inclusion were suggested at the EMICS workshop 2021 and were added to the systematic review.

2.2. Data extraction

Outcome measures: Key characteristics of the systems and their evaluation were extracted from articles and online sources into a standardized documentation form[3]. Nine authors were contacted for additional information. Six responded and five supplemented details about the evaluation. Two authors were contacted for download credentials of a source code[4] and database[5]. Based on the extracted information, the systems were clustered across the target device (mobile vs. desktop vs. browser) and categorized based on the gaze estimation method as proposed by Ferhat and Vilariño (2016):

1. **Appearance-based**: Uses deep learning to infer the PoR from a vector of image pixel intensities.
2. **Feature-based**: Extracts high-level features (e.g., facial feature points) from the image and feeds them into the gaze estimation model.
3. **Model-based**: Creates a geometric model of the eye and infers the PoR from its parameters.

Deployment options were extracted with regard to the runtime environment and license terms (**RQ1**). If not specified, it was assumed that the eye tracker was designed for the operating system for which deployment instructions were provided. The documentation form was piloted on three systems and, as several online repositories appeared to be no longer maintained, was complemented by the date of the last code update.

To assess whether disruptive online data acquisition limits usability (**RQ2**), information was extracted about model initialization and personal calibration.

The primary outcome measures were the reported sampling rate in frames per second (fps) and accuracy (**RQ3**). In settings where the exact distance between the user's eyes and the display are unknown, it is not possible to accurately predict the gaze angle – which is typically used to measure gaze accuracy (Huang et al., 2017). We thus report accuracy as the Euclidean distance in centimeters between the estimated and true PoR. If an article reported only the gaze angle, it was converted into its metric equivalent if the available information allowed it[6]. The results were organized by device type. This distinction was needed because the same metric accuracy on a desktop implies a much higher relative error on the smaller screen of mobile devices. Sampling rates on mobile devices were expected to be lower due to their limited processing power. Since some systems are under continuous development, only the results of their most recent evaluation are reported.

Risk of bias assessment: To ascertain the validity of generalizing the reported performance to other users and contexts, we extracted the number of test subjects and constraints on light, head motion and glasses. Accuracy was considered valid when calculated across all subjects. If technical failures led to the exclusion of samples, we noted their frequency and cause. Validity of the sampling rate was confirmed if the gaze was estimated in real time on the target device. If computation was done offline on another (more powerful) device, we extracted the specifications of the processing component.

3. Software integration (RQ1)

The solutions aim to support multiple devices, but most have been tested on only one (n = 8) or few (n = 1) systems, or on eye tracking databases without ever running on the actual target device (n = 3)[7]. Only software solutions running in the browser (n = 4) have been tested on diverse end user devices. Table 1 lists the eye trackers according to the target devices, with information about supported operating systems.

[3]Documentation form: https://www.ipvs.uni-stuttgart.de/institute/team/Heck-00002/

[4]The password for the NNET source code can be obtained form the author: https://userweb.cs.txstate.edu/~ok11/index.html.

[5]Download of the GazeCapture dataset is made available after registration: https://gazecapture.csail.mit.edu/download.php.

[6]Errors in cm were calculated from the visual angle α as $\epsilon = 2 * D * tan(\alpha/2)$, where D = distance between the eyes and the display.

[7]The accuracy of three additional systems was evaluated on eye tracking databases. However, feasibility tests were performed on the target devices.

Table 1. Summary & terms of use of included eye trackers

	Runtime Environment		Gaze Estimation Model			Release Details		
Software & Author(s)	**Device**	**Software**	**Method**	**Model**	**Training Data**[1]	**License**	**Source Code**	**Updated**
OpenGaze (Zhang et al., 2019)	desktop	Ubuntu	appearance	AlexNet	face images & videos	proprietary research	www.opengaze.org	04-Jun-20
FAZE (S. Park et al., 2019)	desktop	Ubuntu	appearance	MAML	face images	Nvidia	https://github.com/NVlabs/few_shot_gaze	22-Sep-20
GazeRefineNet (S. Park et al., 2020)	desktop	Ubuntu	appearance	ResNet + LSTM	face videos	MIT	https://github.com/swook/EVE	24-May-21
ODABE (Klein Salvalaio & Ramos, 2019)	desktop	*all*	appearance	VGG11	face images	CC0-1.0	https://github.com/brunoklein99/eye-tracking-otl	16-Jul-19
CVC ET (Ferhat et al., 2014)[2]	desktop	Ubuntu, macOS	feature	GP	*Iris segmentation histogram**	GPLv2	https://github.com/tiendan/OpenGazer	10-May-16
Optimeyes (Allen & Jensen, 2014)	desktop	Linux, Windows	feature	LR	*Pupil-center-keypoint vector**	MIT	https://github.com/LukeAllen/optimeyes	25-Mar-21
iTracker (Krafka et al., 2016)	phone/ tablet	*incompatible*	appearance	AlexNet	face images	proprietary research	https://github.com/CSAILVision/GazeCapture	09-Jun-21
GazeEstimator (Jigang et al., 2019)	phone/ tablet	*incompatible*	appearance	ResNet	face images	CC0-1.0	https://github.com/liujigang82/gazeEstimator	26-Feb-19
iMon (Huynh et al., 2022)	phone/ tablet	iOS	appearance	Efficient-NetB3	face images	CC0-1.0	https://github.com/imonimwut/imon	14-Aug-21
GAZEL (J. Park et al., 2021)	phone	Android	appearance	AlexNet + LR	face images	CC0-1.0	https://github.com/joonb14/GAZEL	08-Nov-21
NNET (Holland et al. 2012, Sewell et al. 2010)	tablet	iOS	feature	ANN	*Facial feature points**	GPLv3, commercial	https://userweb.cs.txstate.edu/~ok11/nnet.html	*outdated*
EyeTab (Wood & Bulling, 2014)	tablet	Windows	model	Iris contour	–	MIT	https://github.com/errollw/EyeTab/	07-Apr-14
TurkerGaze (Xu et al., 2015)	*all*	browser	feature	RR + SVR	*Eye appearance feature vector**	MIT	https://github.com/PrincetonVision/TurkerGaze	07-Apr-16
WebGazer (Papoutsaki et al., 2016)	desktop	browser	feature	RR	*Pupil location & eye features**	GPLv3	https://webgazer.cs.brown.edu/	25-Mar-22
RealEye (Lewandowska, 2019)[3]	*all*	browser	feature	RR	*Pupil location & eye features**	commercial	https://www.realeye.io/	continuous support
GazeRecorder (Deja, 2013)	*all*	browser	n/d	n/d	n/d	CC BY-NC, commercial	https://github.com/szydej/GazeRecorder	26-Jun-19

RR: ridge regression; **LR**: linear regression; **GP**: Gaussian Process

[1] Models trained on data marked with asterisk (*) are user-dependent (i.e., they are trained on user data collected at the start of each new session).

[2] The outcome measures reported in this review are based on the more recent evaluation by (Ferhat et al., 2015).

[3] The outcome measures reported in this review are based on the latest software version and evaluation described by (Wisiecka et al., in press).

3.1. Desktop computers

All 6 desktop eye trackers run on Linux, and 3 provide additional support for Windows, macOS, or both.

OpenGaze (Zhang et al., 2019) implements a programming interface for the appearance-based gaze estimation pipeline MPIIFaceGaze (Zhang et al., 2017) with an AlexNet pre-trained on face images. FAZE (S. Park et al., 2019) uses a Model-Agnostic Meta Learning (MAML) approach to adapt a pre-trained model to the individual user based on very few calibration points. GazeRefineNet (S. Park et al., 2020) combines a ResNet with a LSTM trained on video data to take into account temporal patterns. ODABE (Klein Salvalaio & Ramos, 2019) is a platform agnostic solution that uses online transfer learning to personalize a pre-trained VGG11 model with calibration data collected on mouse clicks.

The CVC Eye Tracker (CVC ET) (Ferhat et al., 2015) builds on the deprecated Opengazer (Zieliński, 2009). Its current version transforms the initially appearance (Ferhat et al., 2014) into a feature-based solution. It trains a Gaussian Process with histogram features of segmented iris pixels collected at calibration. Optimeyes (Allen & Jensen, 2014) applies linear regression to a pupil-center keypoint vector, where the keypoint is the center of the eye bounding boxes.

3.2. Mobile devices

On mobile devices, screen dimensions and distance vary substantially. The three systems supporting both tablets and phones thus train models with images from both devices. The appearance-based iTracker (Krafka et al., 2016) and GazeEstimator (Jigang et al., 2019) implement a pre-trained AlexNet and ResNet, respectively. Yet running such large models on mobile devices

would require substantial modifications. Accuracy evaluations are therefore performed on desktop computers with pre-recorded images. In contrast, iMon (Huynh et al., 2022) leverages Apple's CoreML framework to run a pre-trained EfficientNetB3 on iPhone and iPad. Based on the observation that models initialized with tablet data are more accurate due to being trained on data from larger screens, GAZEL initializes an AlexNet with tablet data. In the online phase, a linear regression uses calibration samples to transfer the model to phones.

NNET (Holland & Komogortsev, 2012; Sewell & Komogortsev, 2010) is a feature-based approach for iPads that trains an Artificial Neural Network (ANN) on iris patches extracted from calibration samples. The software is – as declared by the authors – outdated.

EyeTab (Wood & Bulling, 2014) is the only model-based among the reviewed systems. It creates a model of the iris contours to estimate the PoR on tablets.

3.3. Browser integration

Four systems run in the browser. The feature-based TurkerGaze (Xu et al., 2015) trains a ridge regression model at the start of each session with eye appearance features from calibration samples. While this provides real-time gaze estimation, accurate results require a subsequent offline phase to train a refined Support Vector Regression (SVR). According to the developers, the desktop tested software also runs on mobile devices, but with a presumably lower accuracy due to, e.g., more variable head poses and smaller screens. WebGazer (Papoutsaki et al., 2016) uses more refined pupil location and eye features to train a ridge regression. It only supports desktop computers. RealEye (Lewandowska, 2019) is a commercial solution based on WebGazer. Using a customized face landmark model, it supports both mobile and desktop devices. GazeRecorder (Deja, 2013) is another device-agnostic commercial solution. While usage is free for research purposes, the provider does not disclose any details about the gaze estimation method.

3.4. Terms of use (licenses)

Any software published on GitHub – unless accompanied by a separate license – falls under the permissive Creative Commons Zero License CC0-1.0 (Creative Commons, 2017). It can be copied, modified, and distributed even for commercial purposes (n = 1). The MIT license (Open Source Initative, n.d.) allows to copy, modify, and distribute the software, as long as copies of the source code are accompanied by the license statement (n = 4). Systems under the copyleft GNU General Public License GPLv2 / GPLv3 (FSF, 2007) can be copied, modified, and distributed if derivatives are released under the same license (n = 3).

Research licenses restrict usage to non-commercial purposes (n = 4). Under the Creative Commons Attribution Non-Commercial License CC BY-NC 2.0 (Creative Commons, 2017), GazeRecorder may be copied, modified, and distributed for non-commercial purposes if derivatives reference the source code. Proprietary licenses customize the terms of use. The Nvidia Source Code License permits to use, modify, and distribute the source code, but restricts the release of derivatives to non-commercial purposes under the same license (S. Park et al., 2019). OpenGaze may be used and modified for internal research only and does not permit the distribution of derivatives (Zhang et al., 2019). ITracker additionally requests to be cited if results obtained with the software are published (Krafka et al., 2016).

For more permissive use, two systems additionally offer commercial licenses. Only RealEye has no free version, but requires a commercial license with monthly subscriptions starting at $189 (Lewandowska, 2019).

4. Initialization and calibration of the gaze estimation models (RQ2)

Appearance- and feature-based methods train machine learning models on the pixel intensities of face images or on extracted features, respectively (cf. Table 2).

Table 2. Offline training & personal calibration data

	Software	Offline Training	Personal Calibration
appearance	GazeEstimator	GazeCapture	–
	iTracker	GazeCapture	13 points
	iMon	GazeCapture	20 points
	FAZE	GazeCapture	9 points
	GAZEL	10 tablet users	interaction-based
	ODABE	MPIIGaze	interaction-based
	OpenGaze	MPIIGaze, EYEDIAP	60 points
	GazeRefineNet	EVE	saliency-based
feature	NNET	–	16 points
	Optimeyes	–	interaction-based
	TurkerGaze	–	9 points
	CVC ET	–	15 points
	RealEye	–	40 points
	GazeRecorder*	n/d	10-30 points
	WebGazer	–	interaction-based
model	EyeTab	–	–

* Since no information about offline training has been disclosed, GazeRecorder is subsumed under feature-based methods.

Appearance-based methods: Most models (n = 7) are pre-trained on large eye tracking datasets covering diverse ambient conditions and user appearance. GazeCapture[8] contains images from 1249 iPhone and 225 iPad users. Three models were initialized with the dataset's 1271 training samples (Huynh et al., 2022; Jigang et al., 2019; Krafka et al., 2016). FAZE, although

[8]GazeCapture: https://gazecapture.csail.mit.edu/

being designed for desktops, is also trained on GazeCapture (S. Park et al., 2019). For robustness to within-subject variability, only uses samples with more than 400 frames. J. Park et al. (2021) argue that many GazeCapture images represent nontypical use cases (e.g., arms being raised above the head). They therefore train GAZEL with data collected from ten tablet users under professedly more typical usage scenarios.

Datasets with laptop images include MPIIGaze[9] whose 15 samples were used to train ODABE (Klein Salvalaio & Ramos, 2019) and OpenGaze (Zhang et al., 2019). OpenGaze was additionally trained on the 16 videos from EYEDIAP[10]. GazeRefineNet was released with the EVE dataset[11] consisting of 54 videos (S. Park et al., 2020). The sequential video data of EYEDIAP and EVE can be used to train temporal models.

Feature-based methods: Typically, personalized models are created through online calibration where the users look at a set of points – whose number varies across the eye trackers – on the display (n = 9). By knowing the estimation error at the collected points, a calibration model is trained and applied to all subsequent gaze estimates. In OpenGaze, calibration with 60 points is optional, but its omission increases the error by factor 2.56 (Zhang et al., 2019). Yet, as increasing the number of calibration points beyond four leads to only marginal improvements, a reasonable accuracy may be possible with less samples. Similarly, FAZE benefits the strongest from calibration when performed with nine points (S. Park et al., 2019).

As this time-consuming procedure compromises usability, four eye trackers implement implicit techniques:

- **Interaction correlation:** The model is continuously re-calibrated using click data as calibration points (Allen & Jensen, 2014; Klein Salvalaio & Ramos, 2019; Papoutsaki et al., 2016; J. Park et al., 2021). WebGazer starts gaze estimation only after training with 2-3 calibration points. Optimeyes produces accurate results with 10-20 samples, and GAZEL with 3 points, while achieving only marginally better results with more samples.
- **Saliency correlation:** Assuming that users look at visually striking areas, GazeRefineNet correlates gaze points with visual saliency maps of the screen content (S. Park et al., 2020).

Seven appearance-based methods also calibrate the pre-trained models to fit them to each user and context. While the model-based EyeTab is not calibrated, Wood and Bulling (2014) ascribe its high error to the omission.

[9]MPIIGaze: https://www.mpi-inf.mpg.de/mpiigaze
[10]EYEDIAP: https://www.idiap.ch/en/dataset/eyediap
[11]EVE: https://ait.ethz.ch/projects/2020/EVE/

5. Performance outcomes (RQ3)

The primary objective of all included eye trackers was to maximize accuracy. Sampling rate was frequently reported (n = 11), but never the primary goal. Because the experimental conditions, subjects, and HW varied substantially, we describe the evaluations on a qualitative level, rather than performing a meta-analysis.

5.1. Accuracy

Accuracy is typically assessed by asking test subjects to look at points on the display and calculating the Euclidean distance between the estimated PoR and the visual stimulus. Six eye trackers were evaluated on eye tracking datasets. In all other cases, data from a user study was processed offline. The study details – including constraints on the setup that may bias the generalization of the outcome – are presented in Table 3.

We should note that all performance outcomes are reported by the authors and may not be obtained in different settings. For example, when tested with GAZEL's setup, GazeEstimator had an error of 3.86 cm (over three times higher than in the original study) and the 1.9 cm error achieved with iTracker was almost 1.5 times higher (Jigang et al., 2019). In two evaluations, some samples were excluded due to unsuccessful eye tracking (Wood & Bulling, 2014) or technical issues (Papoutsaki et al., 2016). As accuracy was calculated without these samples, we suspect a positive bias in the reported results.

Desktop trackers: On average, feature-based – including software running in the browser, which were all evaluated on desktop computers – outperformed appearance-based methods. With an error of 4.06 cm, WebGazer is a notable outlier (Papoutsaki et al., 2016). However, the value was obtained with the original system which has since been substantially modified. The current version may perform significantly better, but, according to the authors, has not been systematically tested. The other feature-based solutions for which metric accuracy is available predict the PoR within 1.04-2.93 cm of its true coordinates, which beats most appearance-based methods with 2.75-3.77 cm accuracy.

All feature-based eye trackers were evaluated with fixed head position and, exempting CVC ET (Ferhat et al., 2015), require controlled light. While no effect measures for ambient conditions are reported, the developers of RealEye and WebGazer agree that head motion and poor light substantially lower accuracy. Optimeyes even requires retraining if users move their head (Allen & Jensen, 2014). Ferhat et al. (2015) mention the option to make the CVC ET resilient to head motion by implementing a head movement correction. But since no

Table 3. Reported accuracy & evaluation settings

	Software	Accuracy		Hardware		Setup				Comment
		in degrees (pixels)	in cm[1]	Screen Distance[2]	Screen Size	# Test Users	Constraints H	L	G	
appearance	**Desktop trackers**									
	OpenGaze	3.2° (–)	*3.5 cm*	*62.5 cm*	14”	n=20			n=4	- Good light improves accuracy to 2.8° - Glasses reduce accuracy to 4.5°
	FAZE	3.14° (–)	*3.29 cm*	*60 cm*	13.3”-15.4”	n=15			n=5	- Evaluated on MPIIGaze - Issues with glasses and East Asians
	GazeRefineNet	2.49° (95 px)	2.75 cm	65 cm	25”	n=10		✘	n=7	- Evaluated on EVE test set
	ODABE	– (–)	*3.77 cm*	n/d	*14”*	n=1			n=0	- Accuracy reported as normalized value (0.15 on a $\sqrt{2}$ diagonal screen)
feature	CVC Eye Tracker	2.23° (–)	*2.33 cm*	*60 cm*	13”	n=6	✘		n/d	
	Optimeyes	2.8° (79 px)	*(2.93 cm)*[3]	n/d	n/d	n=1	✘	✘	✘	
	Browser trackers									
	WebGazer									- Accuracy with 50 calibration points
	online study	– (175 px)	n/d	n/d	n/d	n=82	✘	✘	n=25	- Exclusion of 7 samples (6 online,
	on-site study	4.17° (130 px)	4.06 cm	59 cm	24”	n=5	✘	✘	n=0	1 on-site): technical issues
	TurkerGaze	1.06° (–)	*1.04 cm*	*56 cm*	18”	n=3	✘	✘	✘	- Accuracy computed as median error after offline SVR model refinement
	RealEye	– (45 px)	n/d	n/d	27”	n=83	✘	✘	n/d	- Robust to glasses if pupils stay visible
	GazeRecorder	1.43° (–)	*1.75cm*	*70 cm*	22”	n=30		✘	n=0	- Chin rest improves accuracy to 1.3° - Robust to glasses (except anti glare)
	Mobile trackers									
	NNET	4.42° (204 px)	*4.49 cm*	*58.42 cm*	9.1”	n=9		✘	n=6	
model	EyeTab	6.88° (–)	2.58 cm	20 cm	11”	n=8			n/d	- Exclusion of 3 samples: 2 unsuccessful gaze estimations, 1 unsuccessful eye detection
appearance	iTracker *(phone)*	– (–)	1.34 cm	varying[4]	3.5”-5.5”	n=121			n=25	- Evaluated on GazeCapture test set
	iTracker *(tablet)*	– (–)	2.12 cm	varying[4]	7.9”-12.9”	n=29			n=5	
	GazeEstimator *(phone)*	– (–)	1.14 cm	varying[4]	3.5”-5.5’	n=121			n=25	- Evaluated on GazeCapture test set
	GazeEstimator *(tablet)*	– (–)	1.92 cm	varying[4]	7.9”-12.9”	n=29			n=5	
	iMon *(phone)*	– (–)	1.11 cm	varying[4]	3.5”-5.5”	n=121			n=25	- Evaluated on GazeCapture test set
	iMon *(tablet)*	– (–)	1.59 cm	varying[4]	7.9”-12.9”	n=29			n=5	- Significant negative effect of glasses
	GAZEL	– (–)	1.49 cm	40-50 cm	6.2”	n=10			n=1	- Issues with glasses and female users

H: fixed head position; **L**: controlled ambient light; **G**: glasses (numbers, if given, indicate the number of study subjects wearing glasses)
[1] Values printed in *italics* were converted from angular degrees based on the formula in Section 2.2.
[2] Distance values are reported by the authors and are either fixed (if using a head rest) or averaged values across all subjects.
[3] As screen distance is not reported, conversion is performed with D = 60 cm (based on the most frequent distance in other desktop settings).
[4] A Neural Network layer infers the distance from the location and size of the head in each frame.

such extension has yet been integrated in any feature-based tracker, it is difficult to estimate its complexity and success. In contrast, most appearance-based trackers were evaluated with diverse head positions and lighting. Only GazeRefineNet was tested while blocking direct and bright light (S. Park et al., 2020). The reported accuracy can thus be expected to be obtained under challenging ambient conditions. Yet OpenGaze achieves a 12.5% higher accuracy under controlled light (Zhang et al., 2019), suggesting that appearance-based methods are indeed affected by light. GazeRecorder (of unknown gaze estimation method) permits free head motion, but a chin rest improves its accuracy by 9% (Deja, 2013).

Mobile trackers: Eye trackers for mobile environments are typically designed for robustness to ambient conditions and user appearance. With the exception of NNET (Holland & Komogortsev, 2012), they were evaluated in perfectly unconstrained settings. The only feature-based – and outdated (cf. Section3.2) – mobile tracker NNET retains an estimation error of 4.49 cm on tablets even in controlled light. With an accuracy of 1.11 - 1.49 cm on phones and 1.59 - 2.12 cm on tablets, appearance-based solutions outperform the model-based EyeTab (accuracy = 2.58 cm on tablets). While the absolute error of most mobile trackers is lower than on desktops, their performance in relation to the screen size is in fact worse. With an error of 9.5% of the diagonal screen extension, GAZEL reports the best performance on phones (J. Park et al., 2021). On desktops, the error of TurkerGaze gets as low as 2.3% (Xu et al., 2015).

While the evaluations, excepting TurkerGaze and Optimeyes, did not exclude subjects with glasses, four authors report a negative effect (Huynh et al., 2022; J. Park et al., 2021; S. Park et al., 2019; Zhang et al., 2019). According to the developers, GazeRecorder and RealEye work with glasses if the eyes remain clearly

visible. TurkerGaze, in contrast, cannot cope with the light reflections and altered eye appearance caused by glasses. Outcomes from evaluations on a random sample including subjects with glasses thus reflect the average expected accuracy, though individual performance may differ (Holland & Komogortsev, 2012; Huynh et al., 2022; Jigang et al., 2019; Krafka et al., 2016; Papoutsaki et al., 2016; J. Park et al., 2021; S. Park et al., 2020; S. Park et al., 2019; Zhang et al., 2019). Systems evaluated on a small sample with no subject wearing glasses may perform worse on a population scope (Deja, 2013; Klein Salvalaio & Ramos, 2019).

5.2. Sampling rate

Since the Neural Networks of appearance-based eye trackers are computationally expensive, the limiting component is typically the device's processing power. Reported sampling rates range from 8 fps on a Pixel 3 XL phone (GAZEL, J. Park et al., 2021) to 60 fps on an iPhone 12 Pro (iMon, Huynh et al., 2022). Since iMon leverages Apple's Neural Engine, older devices with no Bionic graphics card may incur much higher latencies.

The sampling rate of lightweight eye trackers such as RealEye is conditional on the frame rate of the camera. Still, for optimal performance at 30-60 fps, the company's website specifies minimal HW requirements.

TurkerGaze and GazeRefineNet were evaluated offline and report a sampling rate equal to the frame rate of the camera. If computation times are high, the real-time sampling rate may thus be lower.

Given the diversity of the evaluation HW, an objective comparison is not possible. Table 4 thus presents the reported sampling rates on the specified HW.

Table 4. Reported sampling rates

	Software	Frequency	Processing Component
appearance	**Desktop trackers**		
	OpenGaze	13 fps	PC (3.5 GHz CPU, GeForce GTX TITAN Black GPU, 6GB RAM)
	FAZE	15 fps	mobile GTX 1080 TI GPU
	GazeRefineNet	*(10 fps)*	*(offline)*[1]
feature	CVC ET	30 fps	n/d
	Browser trackers		
	TurkerGaze	*(30 fps)*	*(offline)*[1]
	RealEye	30 fps	n/d
	Mobile trackers		
	NNET	0.7 fps	iPad 2 (1 GHz dual-core CPU)
model	EyeTab	12 fps	Microsoft Surface Pro 2 (quad-core 2 GHz CPU, 8 GB RAM)
appearance	iTracker	*(10-15 fps)*	*(iPhone 6)*[2]
	iMon	60 fps	iPhone 12 Pro (Apple GPU)
	GAZEL	8 fps / 12 fps / 18 fps	Pixel 3 XL / Galaxy S9 Plus / Galaxy S20 Plus

[1] Gaze estimation is not performed on the target device.
[2] Expected (not yet implemented) on iPhone 6 with modified model.

6. Discussion and conclusion

We identified 16 eye tracking software solutions for mobile and desktop devices. Research licenses restrict the use of four systems to non-commercial purposes, and one is charged on a monthly subscription (**RQ1**).

We extracted the expected performance of the eye trackers as reported by the authors (**RQ3**). Some reported an accuracy that would be sufficient for most applications (Duchowski, 2002), but evaluations on different devices and with different users tended to produce substantially less optimistic results (e.g., Jigang et al., 2019). Sampling rates – if reported at all – could not match the performance of commercial eye tracking HW. With a frequency of 30 fps – which can be obtained from both feature- and appearance-based eye trackers as long as the client hardware is sufficiently powerful – it is possible to extract fixations and saccades. In contrast, the detection of microsaccades, which play an important role in emotional and cognitive state prediction, require a minimum sampling rate of 300 Hz (Duchowski, 2018).

One major challenge is to collect appropriate user data for model initialization and personal calibration (**RQ2**). Appearance-based eye trackers trained on a diverse dataset are generally most capable of coping with challenging light and head motion. However, two systems performed poorly for users that are underrepresented in the training datasets. FAZE was less accurate for Asians (S. Park et al., 2019), and GAZEL for females (J. Park et al., 2021). Glasses remain a challenge for most systems. Training the models – i.e., Neural Networks of appearance-based or eye detection engines of feature-based systems – with a large number of eyeglass wearers can partly alleviate the issue, but only if the eye is not (partly) occluded by the glasses or light reflections. Yet accurate gaze estimation still requires online refinement through personal calibration.

6.1. Limitations

Study and outcome level: The reviewed articles report performance selectively in inconsistent units, and the multiplicity of evaluation setups make an objective comparison unfeasible. We were only partly able to resolve the issue of selective reporting, because not all authors provided the missing information. Our synthesis is therefore of pure qualitative nature. For a quantitative comparison, all eye trackers should be evaluated with the same devices, users, and ambient conditions. Cheng et al. (2021) have started to implement a small sample of appearance-based eye trackers to compare their accuracy on multiple eye tracking datasets. For the evaluation of the sampling rate, the software additionally has

to run on the actual target device. Since the eye trackers were designed for different runtime environments, a comprehensive evaluation is not a trivial undertaking.

Review level: We found a high number of papers describing a gaze estimation method that made no mention of the source code being publicly available. In some cases, we found the code by searching the authors' websites and online repositories. But since multiple authors have left their research institutions, it is possible that we missed a system that is in fact publicly available.

6.2. Implications for practice

We developed a catalog of publicly available eye tracking software to help third party application developers identify working solutions that meets their runtime environment and performance requirements and is licensed for the intended purpose. In the process of our review, we examined the source codes and consulted authors whenever we suspected that a software was no longer supported. We could thus filter out open source eye trackers listed in previous reviews (e.g., Cheng et al., 2021; Ferhat & Vilariño, 2016; Shehu et al., 2021) that are no longer supported, or require external hardware or third-party software. We further observed that some of the mobile trackers (i.e., iTracker and GazeEstimator) have never actually been implemented on the target system. Rather, they have been designed anticipating the advent of even more powerful HW. Excluding these two solutions, iMon promises the best accuracy performance for iOS, and GAZEL for Android devices. The browser based solutions TurkerGaze, RealEye, and GazeRecorder may also be an option, but accuracy is only reported for desktop environments. All three software solutions outperform desktop trackers in terms of accuracy, and achieve a comparable or better sampling rate. If no online analysis is required, TurkerGaze promises highly accurate gaze estimates. Projects that require real-time evaluations may instead revert to GazeRecorder which promises comparable real-time results. In terms of usability, the interaction-based calibration in GAZEL, ODABE, Optimeyes, and WebGazer, or the saliency-based approach in GazeRefineNet have a clear advantage over point-calibration. The latter becomes more user-friendly as the number of calibration points is reduced, but at the cost of less accurate predictions. Developers are thus typically faced with a trade-off between accuracy and usability, which should be evaluated carefully with regard to the requirements of the use case.

6.3. Implications for research

Webcam eye tracking facilitates behavioral research that uses eye tracking to analyze user intentions and states outside of research labs. The global Covid-19 pandemic, which made it virtually impossible to invite human subjects to the lab, has demonstrated just how essential it is for researchers to be able to collect data without using expensive or complicated hardware.

Previous reviews have shown that a plethora of eye tracking methods have been developed in research labs, but only a small fraction has been made publicly available. It would be a tremendous gain for the community if researchers who have already implemented a system released the source code so that others can more easily build on their advances. We hope that the evidence from this literature review will motivate a more open sharing of research artifacts to accelerate the design of better eye trackers and the diffusion of eye tracking applications.

References

Allen, L., & Jensen, A. (2014). *Webcam-based gaze estimation* (tech. rep.). Stanford. https://github.com/LukeAllen/optimeyes

Cheng, Y., Wang, H., Bao, Y., & Lu, F. (2021). Appearance-based gaze estimation with deep learning: A review and benchmark. https://doi.org/10.48550/ARXIV.2104.12668

Creative Commons. (2017). About the licenses. https://creativecommons.org/licenses/?lang=en

Cui, W., Cui, J., & Zha, H. (2017). Specialized gaze estimation for children by convolutional neural network and domain adaptation. *ICIP*, 3305–3309. https://doi.org/10/hxpp

Deja, S. (2013). *The Comparison of Accuracy and Precision of Eye Tracking: GazeFlow vs. SMI RED 250* (tech. rep.). SIMPLY USER. https://gazerecorder.com/Raport/

Drewes, H., Pfeuffer, K., & Alt, F. (2019). Time- and space-efficient eye tracker calibration. *ETRA*. https://doi.org/10.1145/3314111.3319818

Duchowski, A. T. (2002). A breadth-first survey of eye-tracking applications. *Behav. Res. Methods Instrum. Comput.*, *34*(4), 455–470. https://doi.org/10.3758/bf03195475

Duchowski, A. T. (2018). Gaze-based interaction: A 30 year retrospective. *Computers & Graphics*, *73*, 59–69. https://doi.org/10/gds5mv

Ferhat, O., Llanza, A., & Vilariño, F. (2015). A feature-based gaze estimation algorithm for natural light scenarios. *IbPRIA*, 569–576. https://doi.org/10.1007/978-3-319-19390-8_64

Ferhat, O., & Vilariño, F. (2016). Low cost eye tracking: The current panorama. *Comput. Intell. Neurosci.*, *2016*. https://doi.org/10/ggfnwk

Ferhat, O., Vilariño, F., & Sanchez, F. J. (2014). A cheap

portable eye-tracker solution for common setups. *J. Eye Mov. Res.*, *7*(3). https://doi.org/10.16910/jemr.7.3.2

FSF. (2007). GNU general public license. https://doi.org/10.1007/3-540-28623-3_23

Gibaldi, A., Vanegas, M., Bex, P. J., & Maiello, G. (2017). Evaluation of the tobii eyex eye tracking controller and matlab toolkit for research. *Behavior research methods*, *49*(3), 923–946.

Hansen, D., & Ji, Q. (2010). In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Trans. Pattern Anal. Mach. Intell.*, *32*(3), 478–500. https://doi.org/10.1109/TPAMI.2009.30

Holland, C., & Komogortsev, O. (2012). Eye tracking on unmodified common tablets: Challenges and solutions. *ETRA*. https://doi.org/gfs3z4

Huang, Q., Veeraraghavan, A., & Sabharwal, A. (2017). Tabletgaze: Dataset and analysis for unconstrained appearance-based gaze estimation in mobile tablets. *Mach. Vision Appl.*, *28*(5–6), 445–461. https://doi.org/10/hxpg

Hughes, J., Rhodes, S., & Dunne, B. (2017). Eye gaze detection system for impaired user gui control. *MWSCAS*, 1348–1351. https://doi.org/10/hxt5

Huynh, S., Balan, R. K., & Ko, J. (2022). Imon: Appearance-based gaze tracking system on mobile devices. *IMWUT*, *5*(4). https://doi.org/10.1145/3494999

Jigang, L., Francis, B. S. L., & Rajan, D. (2019). Free-head appearance-based eye gaze estimation on mobile devices. *ICAIIC*, 232–237. https://doi.org/10.1109/ICAIIC.2019.8669057

Karamchandani, H., Chau, T., Hobbs, D., & Mumford, L. (2015). Development of a low-cost, portable, tablet-based eye tracking system for children with impairments. *i-CREATe*. https://dl.acm.org/doi/abs/10.5555/2846712.2846718

Klein Salvalaio, B., & Ramos, G. d. O. (2019). Self-adaptive appearance-based eye-tracking with online transfer learning. *BRACIS*, 383–388. https://doi.org/10.1109/BRACIS.2019.00074

Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., & Torralba, A. (2016). Eye tracking for everyone. *CVPR*, 2176–2184. https://doi.org/10/gf8npv

Lewandowska, B. (2019). *RealEye eye-tracking system* (tech. rep.). RealEye. https://support.realeye.io/articles-mentioning-realeye/

Liberati, A., Altman, D., Tetzlaff, J., Mulrow, C., Gøtzsche, P., Ioannidis, J., Clarke, M., Devereaux, P., Kleijnen, J., & Moher, D. (2009). The prisma statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions. *J. Clin. Epidemiol.*, *62*(10). https://doi.org/cpcqdt

Open Source Initative. (n.d.). The MIT license. https://opensource.org/licenses/MIT

Papoutsaki, A., Sangkloy, P., Laskey, J., Daskalova, N., Huang, J., & Hays, J. (2016). Webgazer: Scalable webcam eye tracking using user interactions. *IJCAI*, 3839–3845. https://dl.acm.org/doi/10.5555/3061053.3061156

Park, J., Park, S., & Cha, H. (2021). Gazel: Runtime gaze tracking for smartphones. *PerCom*, 1–10. https://doi.org/10/hxn7

Park, S., Aksan, E., Zhang, X., & Hilliges, O. (2020). Towards end-to-end video-based eye-tracking. *ECCV*, 747–763. https://doi.org/10/hxpc

Park, S., De Mello, S., Molchanov, P., Iqbal, U., Hilliges, O., & Kautz, J. (2019). Few-shot adaptive gaze estimation. *ICCV*, 9367–9376. https://doi.org/10.1109/ICCV.2019.00946

Sewell, W., & Komogortsev, O. (2010). Real-time eye gaze tracking with an unmodified commodity webcam employing a neural network. *CHI EA*, 3739–3744. https://doi.org/10/cb33zn

Shehu, I., Wang, Y., Athuman, A., & Fu, X. (2021). Remote eye gaze tracking research: A comparative evaluation on past and recent progress. *Electronics*, *10*(24). https://doi.org/10/hxpb

Wang, X., Bylinskii, Z., Castelhano, M., Hillis, J., & Duchowski, A. (2021). Emics'21. *CHI EA*. https://doi.org/10.1145/3411763.3441357

Wisiecka, K., Krejtz, K., Krejtz, I., Sromek, D., Cellary, A., Lewandowska, B., & Duchowski, A. T. (in press). Comparison of webcam and remote eye tracking. *ETRA*. https://doi.org/10/hxn8

Wood, E., & Bulling, A. (2014). Eyetab: Model-based gaze estimation on unmodified tablet computers. *ETRA*, 207–210. https://doi.org/10/gfs3z2

Xu, P., Ehinger, K., Zhang, Y., Finkelstein, A., Kulkarni, S., & Xiao, J. (2015). Turkergaze: Crowdsourcing saliency with webcam based eye tracking. https://doi.org/10.48550/ARXIV.1504.06755

Zhang, X., Sugano, Y., & Bulling, A. (2019). Evaluation of appearance-based methods and implications for gaze-based applications. *CHI*, 1–13. https://doi.org/10.1145/3290605.3300646

Zhang, X., Sugano, Y., Fritz, M., & Bulling, A. (2017). It's written all over your face: Full-face appearance-based gaze estimation, 2299–2308. https://doi.org/10.1109/CVPRW.2017.284

Zieliński, P. (2009). Opengazer: open-source gaze tracker for ordinary webcams. http://www.inference.phy.cam.ac.uk/opengazer/