UnitEye: Introducing a User-Friendly Plugin to Democratize Eye Tracking Technology in Unity Environments

Tobias Wagner*
tobias.wagner@uni-ulm.de
Institute of Media Informatics, Ulm University
Ulm, Germany

Mark Colley*
mark.colley@uni-ulm.de
Institute of Media Informatics, Ulm University
Ulm, Germany

Daniel Breckel
daniel.breckel@uni-ulm.de
Institute of Media Informatics, Ulm University
Ulm, Germany

Michael Kösel
michael.koesel@uni-ulm.de
Institute of Media Informatics, Ulm University
Ulm, Germany

Enrico Rukzio
enrico.rukzio@uni-ulm.de
Institute of Media Informatics, Ulm University
Ulm, Germany

ABSTRACT

Eye tracking is a powerful tool for analyzing visual attention, as an input technique, or for diagnosing disorders. However, eye tracking hardware is expensive and not accessible to everyone, thus, considerably limiting real-world usage or at-home evaluations. Although webcam-based eye tracking is feasible due to advances in computer vision, its open-source implementation as an easy-to-use tool is lacking. We implemented UnitEye, a Unity plugin enabling eye tracking on desktop and laptop computers. In a technical evaluation (N=12), we tested the precision and accuracy of our system compared to a state-of-the-art eye tracker. We also evaluated the usability of UnitEye with N=5 developers. The results confirm that our system provides reliable eye tracking performance for a webcam-based system and well-integrated features contributing to ease of use.

CCS CONCEPTS

• **Computing methodologies** → Machine learning; • **Human-centered computing** → *Accessibility*; *Empirical studies in HCI*; *Interaction techniques.*

KEYWORDS

eye tracking; open-source; technical evaluation

ACM Reference Format:
Tobias Wagner, Mark Colley, Daniel Breckel, Michael Kösel, and Enrico Rukzio. 2024. *UnitEye*: Introducing a User-Friendly Plugin to Democratize Eye Tracking Technology in Unity Environments. In *Proceedings of Mensch und Computer 2024 (MuC '24), September 01–04, 2024, Karlsruhe, Germany.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3670653.3670655

*Both authors contributed equally to this research.

MuC '24, September 01–04, 2024, Karlsruhe, Germany

ACM ISBN 979-8-4007-0998-2/24/09
https://doi.org/10.1145/3670653.3670655

1 INTRODUCTION

Eye tracking is a vital instrument within Human-Computer Interaction (HCI) [34] employed as an input method [12, 14, 22, 28, 35, 43], for activity recognition [6], as a tool for usability testing [38], eye tracking is also instrumental in various research areas for the examination of visual attention [5, 13, 16, 23, 31, 42], or to predict illnesses [19].

However, the need for specialized equipment, such as traditional infrared-based eye trackers, increases the cost and complexity of including eye tracking as a ubiquitous interaction and evaluation tool. While advancements in computer vision have led to the development of appearance-based gaze estimation methods [32, 57], these innovations are still not yet ready for widespread adoption. To be useful, they require significant time and resources for integration, necessitating the expertise of developers with specialized knowledge in machine learning.

In this context, the Unity game engine, which is utilized by approximately 230,000 developers in 2022 to create and operate over 750,000 games [45], is highly important. Its popularity extends to the academic sphere, making it one of the most commonly used game engines for research purposes. To date, the commercial platform of Tobii, Pupil Labs or, the integration of eye tracking in Meta's Quest, Pico, and Varjo products are the de facto only Unity-compatible eye tracking solutions. However, these solutions primarily address eye tracking within head-mounted displays for augmented and virtual reality, neglecting the aspect of eye tracking for desktop, mobile phone, or tablet applications. While open-source variants exist for websites [37], and smartphones [4, 28], a solution for Unity is missing. The introduction of a freely available "plug-and-play" Unity plugin for webcam-based eye tracking, therefore, signifies an important step forward.

Therefore, we introduce *UnitEye*, a tool for a widely accessible engine that has the potential to impact beyond expert developers, embodying the democratization of eye-based interactions. Such democratization can broaden the application of eye tracking, harnessing the creativity and efforts of many to introduce eye-based interactions into new contexts and applications. *UnitEye* offers an open-source solution for Unity that eliminates the need for

additional hardware. It represents an inclusive tool based on affordable and readily available hardware, especially designed to be user-friendly. By providing an easy-to-implement, low-cost eye tracking option, *UnitEye* lowers the entry barrier, facilitating the incorporation of eye tracking technology into new systems.

In a technical evaluation with N=12 participants and 528 data points, we compared the performance of *UnitEye* to a state-of-the-art eye tracker using video-based pupil and corneal reflection. While *UnitEye* understandably performs worse, we show its current capabilities and performance. Further, we report a survey with N=5 developers, which employed *UnitEye* in five case studies.
Contribution Statement: (1) With *UnitEye*, we provide an open-source Unity plugin for camera-based eye tracking along essential tools for eye tracking methodology. (2) We confirm *UnitEye*'s performance via a technical evaluation (N=12) and its ease of use via qualitative feedback from N=5 developers. (3) We present five case studies where *UnitEye* were used to enable gaze-based interaction and evaluation.

2 RELATED WORK — GAZE ESTIMATION

The study of gaze estimation has been extensive, crossing various fields such as computer vision [1], graphics [54], and human-computer interaction [4, 28]. There are two primary types of gaze-tracking devices: wearable eye trackers [2] and external cameras [51]. With the progression of deep learning and the accessibility of high-quality cameras, methods using image-based gaze estimation are gaining traction.

Various camera technologies are employed for gaze tracking, including infrared (IR) eye-trackers (e.g., Tobii), depth cameras [4, 18], and standard RGB cameras [28, 48, 51, 56]. Heuristic [17] and machine learning [28] methods that use these cameras generally fall into either model-based [44, 55] or appearance-based [32, 57] categories. Model-based methods construct geometric models of the eye's structure, like tracking the iris's shape, whereas appearance-based techniques use supervised learning with raw eye images. Appearance-based methods have seen wider success, partly thanks to large datasets like the GazeCapture dataset [29] and the MPIIGaze [58] together with the progress of CNN-based approaches [3] that enable for example iTracker [29].

Continuous improvements were made to enhance the accuracy of these systems, with recent advancements achieving high precision, especially with user-specific calibration [57]. Valliappan et al. [48] present a significant achievement in smartphone eye tracking, achieving state-of-the-art results (uncalibrated accuracy of 1.92 cm; 0.5 cm with a per-user calibration). Kong et al. [28] built on this by integrating user head orientation data, aiming to develop a model that is both generalizable and does not require user-specific calibration. A comparison of webcam and remote and integrated eye trackers [52] showed higher measurement errors for webcam-based eye tracking. Still, the authors conclude that webcams are viable for eye tracking [52].

While there have been significant advancements across various areas in eye tracking, the goal of Krafka et al. [29] "Eye Tracking for Everyone" has not been reached. Current approaches such as EyeMU [28] target mobile platforms but hinder direct integration into the OS or a game engine. Furthermore, previous open-source implementations of appearance-based gaze estimation, such as the OpenGaze toolkit [57][1] or Gaze-Unity, a project available on GitHub [2], are outdated, unmaintained for years, and therefore difficult to install for users with low programming skills. As a remedy, we present *UnitEye*, which builds on EyeMU [28], to bring eye tracking to Unity and to provide an open-source project for further integration of state-of-the-art appearance-based gaze estimation models.

3 UNITEYE IMPLEMENTATION AND FEATURES

We created *UnitEye*, a Unity package that facilitates eye tracking using a single webcam, extending the EyeMU framework introduced by Kong et al. [28]. Utilizing Google's MediaPipe via the MediaPipeUnityPlugin for head and eye detection, *UnitEye* leverages the normalized eye corner coordinates and head orientation to determine head movements including yaw, pitch, and roll [28]. Thus, the

[1]https://git.hcics.simtech.uni-stuttgart.de/public-projects/opengaze, accessed 03.06.2024
[2]https://github.com/souravrs999/Gaze-Unity, accessed 03.06.2024

Figure 1: *UnitEye* UI. This window allows relocation by dragging the top bar and includes a Webcam controls section for selecting from available webcams in Unity. There's a feature for calibrating the distance to the camera, which is a one-time setup that saves the settings in "PlayerPrefs", ensuring persistence across sessions. Additionally, the UI offers calibration for the Blinking and Drowsiness detection system, and sections for adjusting Calibration and Filtering types, and initiating Calibration and Evaluation sequences that can be exited anytime with a right-click.

high-level pipeline processes RGB images by first cropping the eyes and concurrently estimating the head pose. Then, the gaze point is inferred. EyeMU's foundation is a CNN trained on the GazeCapture dataset, as noted by Krafka et al. [29], which allows for the prediction of gaze positions on a screen (for details about the model see [28]). These predictions are then aligned with corresponding game objects. We have converted the EyeMU model into the `.onnx` format (which is, unfortunately, hard, and does not support deep learning (yet)) and supplemented it with various calibration methods: raw output (`None`), Ridge Regression-based refinement [24], and a machine learning approach (`ML Calibration`) employing a multilayer perceptron. Additionally, we have implemented filtering techniques (`Kalman` filter as per Welch et al. [50], `Easing`, and 1€ filter as described by Casiez et al. [11]), along with an "area of interest" feature that identifies gaze targets either as game objects with a "Gazeable" attribute or screen regions (see Figure 1). Our API is designed to mirror the Tobii API, enabling developers to switch from Tobii to UnitEye with minimal code changes. A class diagram is available in the accompanying repository under `Documentation/Class Diagrams/`. *UnitEye* features are:

- **Webcam Integration**: *UnitEye* offers seamless integration with webcams, enabling eye tracking within Unity projects without the need for any specialized hardware—just a standard webcam.
- **Gaze Filtering**: Incorporates advanced algorithms to filter gaze data for smoother and more stable tracking results.
- **Custom Calibration**: Provides a calibration tool tailored to individual setups, enhancing accuracy and personalizing the user experience.
- **Accuracy Evaluation**: Includes a comprehensive evaluation module to test and verify the precision of eye tracking within the user's specific environment.
- **Area of Interest (AOI) Tracking**: Features an AOI system that allows users to define specific areas or objects on the screen for focused tracking. We currently offer 7 different shapes: `AOIBox`, `AOICircle`, `AOICapsule`, `AOICapsuleBox`, `AOIPolygon`, `AOICombined`, and `AOITagList`. The `AOITagList` is a shape that allows you to interact with GameObjects in Unity. It throws a RayCast into the scene at the gaze location and return hit objects that match predefined tags from a defined list.
- **Data Logging**: Equipped with CSV logging functionality, making it effortless to record and analyze eye tracking data for research and development purposes.
- **Attention Metrics**: Detects user's distance from the camera, blinks, and signs of drowsiness. Blink detection has been utilized for various purposes, including measuring drowsiness to enhance driver safety [27], as a control mechanism in assistive technology [30], and to assess engagement [41] and activity levels [26].
- **Runtime Configuration GUI**: Comes with a built-in graphical UI for convenient runtime configuration, allowing for on-the-fly adjustments without coding.
- **Developer-Friendly API**: Offers an easy-to-use API, enabling developers to implement and customize eye tracking features within their Unity projects quickly.

4 TECHNICAL EVALUATION

To test the precision and accuracy of *UnitEye* together with smoothing filters compared to a state-of-the-art eye tracker using video-based pupil and corneal reflection, we conducted a technical evaluation. In the following, we report the evaluation procedure, the data processing and analysis of our eye tracking study.

4.1 Apparatus

We designed a standalone application using Unity 2022.3.7. Regarding hardware, we used an MSI GP66 Leopard 11UG laptop with a FullHD 15.6-inch monitor (141 ppi). For *UnitEye*, we employed the built-in camera with 720p/30 Hz. As a baseline eye-tracker, we used the state-of-the-art commercial Tobii Pro Spark eye tracker mounted below the monitor with the Tobii Pro SDK Unity API (v1.11). The Tobii Pro Spark is screen-based and captures gaze and pupil data with a precision of 0.26° RMS (using built-in filtering) and an accuracy of 0.45° in optimal conditions. The head pose was unconstrained.

4.2 Participants

12 participants (Mean age = 27.40, SD = 2.30, range: [25, 32]; Gender: 25.0% women, 75.0% men, 0.00% non-binary; Education: University degree, 100.00%) took part in the technical evaluation. Seven are employees, and five are university students. Regarding physical properties of the eye region, 10 wore no makeup, while two wore little makeup (scale: no, little, yes). Five had brown, four black, and three blond hair. After Mackey et al. [33], four participants had dark brown, four light brown, two green with a brown peripupilary ring, and two light blue eyes. Figure 3 shows outlines of participants' left eye shapes. Regarding ethnicity, which also affects tracking performance [8], 10 participants were White, one Asian, and one chose "multiple". No participants wore glasses, one wore contact lenses. While we did not control for these variables, we report collected participant characteristics, as these may influence both face recognition [46] and eye tracking performance [10].

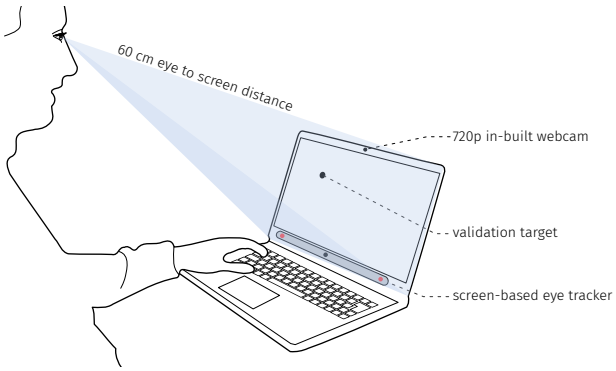


Figure 2: Exemplary participant sitting 60 cm in front of the laptop.

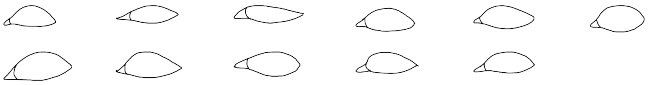


Figure 3: Outline sketches of participants' visible left eye shape. One participant did not agree to share their eye shape.

4.3 Procedure

Each participant experienced **four** conditions with each 11 validation points (see Figure 4). Each session started with a brief introduction, signing of the consent form, and a demographic questionnaire. Participants were seated in front of the laptop at an eye-screen distance of 60 cm (see Figure 2). First, they had to calibrate the eye tracker of the current condition and then fixate 11 validation points in sequence. Each validation point was shown for 4 seconds in which the point got smaller. We conducted the calibration with the following validation twice per eye tracker to have more robust data and to alter the sequence in which validation points were shown. After the validation started, the experimenter measured the brightness in the room in lux (M=493.21, SD=34.52). Both the conditions and the order of the validation points were presented in counterbalanced order.

The introduction for the *UnitEye* smooth pursuit calibration was: *First, you will see yourself on the screen. Check that your head is centered. You will then be guided through a calibration process. Follow the black, moving circle with your eyes. This first zigzags from bottom to top, from left to right. In the next step, it zigzags from top to bottom, from left to right.*

The introduction to Tobii's 5-point calibration was: *You will be guided through a 5-point calibration process. Please fixate on the red dots as soon as they appear. Fixate each point for the entire duration.*

After the calibration, the introduction to the validation in all conditions was: *You will then be asked to fix eleven points in sequence. These initially appear in full size and then become smaller. Fixate these points for the entire duration.*

The study took 15 min. Participants were remunerated with 3€.

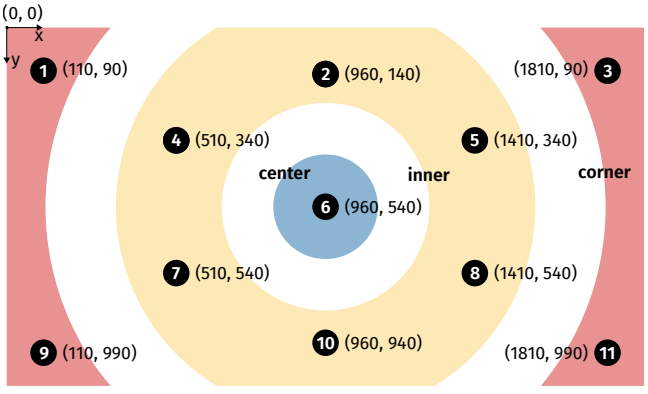


Figure 4: The 11-point validation pattern. Only the black dots were visible during the validation process.

4.4 Privacy and Ethics Statement

This study employs eye tracking technology to evaluate *UnitEye* technically. Participant consent was obtained through signing, ensuring awareness of data use. Data security is upheld with adherence to our university's guidelines. To mitigate risks, no raw camera data was logged. Local regulations did not require a formal ethics review.

4.5 Objective Measurements

We collected the x and y coordinates of gaze points on the display with timestamps to later calculate the precision and accuracy. During each validation phase, gaze data were logged by either *UnitEye* or the Tobii system. *UnitEye* logged the gaze data at 30 Hz and the Tobii system logged at 60 Hz. While *UnitEye* supports higher framerates, it was limited through the in-built camera framerate of 30 fps. For our experiments and use cases, 30 Hz proved to be reliable and adequate for fixation-based eye tracking. *UnitEye* estimates the location of the gaze point on the display with normalized x and y coordinates ranging from 0.0 to 1.0. Additionally, *UnitEye* provides smoothed data. For this evaluation, we selected the 1€ filter ($\beta = 0.001$, $f_{cmin} = 0.001$, $f_{cutoff} = 1$). The raw and smoothed data is written in a comma-separated file together with an associated timestamp. Similarly, Tobii provides normalized x and y coordinates of the gaze location on the display, but individually for the left and right eye with a timestamp and a field that marks the data entry as valid or invalid. The data for Tobii is written in an XML file. In addition to the gaze data, we collected the brightness in the room measured in lux for each validation phase using the light meter Uni-T Mini Light Meter UT383 (accuracy 0-999 Lux ±4%+8) [47].

4.6 Data Analysis

The gaze data collected through Tobii were preprocessed by converting it into a comma-separated file and averaging left and right eye coordinates to obtain a single gaze point. Then, the collected and normalized gaze points through *UnitEye* and Tobii were multiplied by the display size (i.e., FullHD). To ensure that we only use data to measure accuracy and precision where we are confident that participants are already fixating on the validation point, we averaged data in the time segment between 1.5 seconds and 3.5 seconds of the validation point being visible. No invalid data points were found in the data collected by Tobii or *UnitEye* for this time segment. This resulted in overall 264 data points per *eye tracker* which were used for the performance evaluation. Performance was measured by precision and accuracy. The precision was calculated using the Root Mean Square sample-to-sample deviation (RMS-S2S) and the standard deviation (SD) in x and y direction [36]. The accuracy was measured by calculating the x and y distance and Euclidean distance between the gaze and the validation point.

$$RMS - S2S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \quad (1)$$

Before every statistical test, we checked the required assumptions (e.g., normality distribution). All measures were analyzed in pixels and converted for the plots to cm and degrees of visual angle (°) for comparability with prior work. We used the ARTool package by Wobbrock et al. [53] for non-parametric data, as the typical

ANOVA is inappropriate with non-normally distributed data. The procedure is abbreviated, as in the original publication, with ART. R in version 4.4.0 and RStudio in version 2024.04.1 were employed. All packages were up-to-date in June 2024. Effect sizes were calculated using Rosenthal's formula [40].

4.7 Results

4.7.1 Precision. Precision was calculated using RMS-S2S and SD in x and y direction. Regarding RMS-S2S, the ART found a significant main effect of *eye tracker* ($F(2, 22) = 213.48$, p<0.001, r=-0.279, Z=-7.86) and of *validation area* ($F(2, 22) = 74.17$, p<0.001, r=-0.227, Z=-6.39). In addition, the ART found a significant interaction effect of *eye tracker* × *validation area* ($F(4, 44) = 83.25$, p<0.001; see Figure 5b). With the 1€ filter enabled, *UnitEye* performs better than Tobii for the RMS-S2S value. The highest improvement is visible in the center area.

On SD in x direction, ART found a significant main effect of *eye tracker* ($F(2, 22) = 96.90$, p<0.001, r=-0.241, Z=-6.78) and of *validation area* ($F(2, 22) = 44.07$, p<0.001, r=-0.199, Z=-5.61). Additionally, Art found a significant interaction effect of *eye tracker* × *validation area* ($F(4, 44) = 54.52$, p<0.001; see Figure 6a). The raw *UnitEye* values perform worst. However, the corner values for Tobii perform the worst, while this is the best-performing area for *UnitEye* (raw and smoothed).

On SD in y direction, the ART found a significant main effect of *eye tracker* ($F(2, 22) = 126.29$, p<0.001, r=-0.254, Z=-7.15) and of *validation area* ($F(2, 22) = 11.47$, p<0.001, r=-0.126, Z=-3.55). Further, ART found a significant interaction effect of *eye tracker* × *validation area* on SD in y direction ($F(4, 44) = 15.74$, p<0.001; see Figure 6b). The raw *UnitEye* values perform worst. However, the corner values for Tobii perform the worst, while this is the best-performing area for *UnitEye* (raw and smoothed). Generally, the values for SD in the x and y directions show the same characteristics.

4.7.2 Accuracy. Accuracy was calculated using the Euclidean distance between target and gaze position. Here, the conducted ART found a significant main effect of *eye tracker* ($F(2, 22) = 200.93$, p<0.001, r=-0.276, Z=-7.78) and of *eye tracker* × *validation area* ($F(4, 44) = 6.94$, p<0.001; see Figure 5a). While Tobii performs best for all validation areas, the difference becomes smaller when the 1€ filter is added. However, for the corners, the filter does not improve accuracy.

4.7.3 AOI Size. Based on the precision given as the SD in x and y direction, and the accuracy given as the distance to the target, we calculated the minimum AOI size on a 34.5 cm wide and 19.5 cm high 15'6 inch display. To account for deviations in both directions, the minimum area size is equal to $2x(accuracy + precision)$. As shown in Table 1, the minimum AOI size is the smallest for the Tobii system (1.3 x 1.5 cm), followed by the *UnitEye* system with smoothing (6.2 x 5.8 cm), and the *UnitEye* system without smoothing (7.7 x 7.4 cm).

Eye Tracking System	Accuracy (cm)		Precision (cm)		Area Size (cm)		Number of Areas	
	x	y	x	y	x	y	x	y
Tobii	0.41	0.51	0.24	0.23	1.3	1.5	27	13
UnitEye	2.66	2.55	1.18	1.17	7.7	7.4	4	3
UnitEye Smoothed	2.55	2.40	0.53	0.50	6.2	5.8	6	3

Table 1: Number and size of smallest possible areas of interest (AOIs) on a 15'6 inch display with the dimensions of (w) 34.5 x (h) 19.5 cm.

5 CASE STUDIES

In line with usability testing and in the spirit of "eating one's own dogfood" [20], *UnitEye* was integrated in various Unity projects by students and members of our group to use eye tracking for explicit input or as a research method for eye movement analysis. This resulted in **five** case studies that contributed to further development. The case studies included a gaze-controlled game ("Shed Some Fear") [15] (see subsection 5.1), two lab studies (see subsection 5.2 and subsection 5.3), a simulation with a remote local installation (see subsection 5.4), and a web-based simulation (see subsection 5.5). We interviewed the developers of these projects to evaluate the usability of *UnitEye*. As no eye tracking was used in this developer survey, we refrain from an additional privacy and ethics statement. We first introduce the use cases and then report the findings from the interviews.

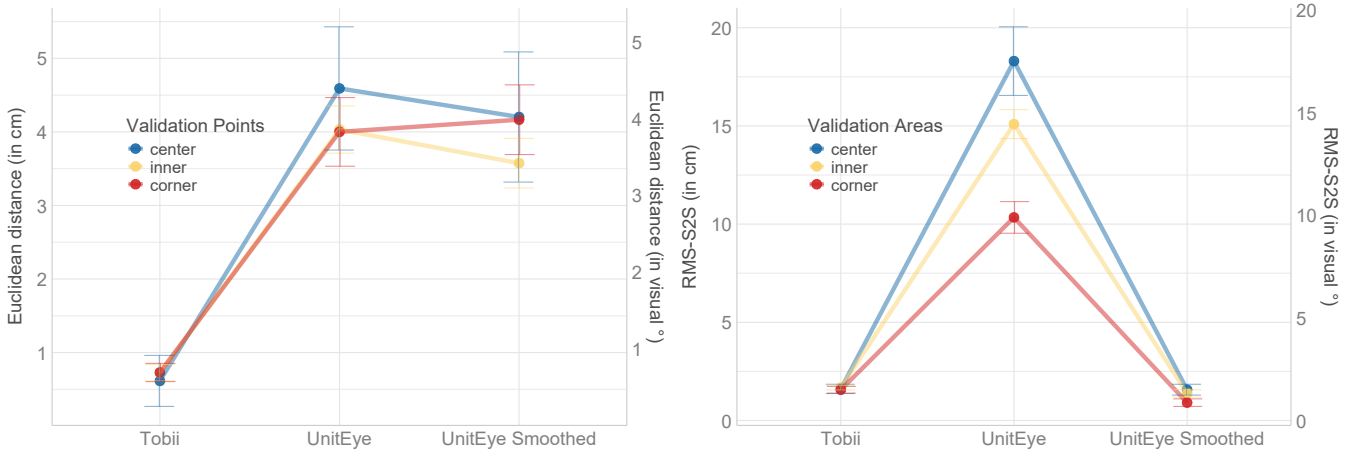
5.1 Use Case 1: Eye Gaze-based Game

The study [15] introduces "Shed Some Fear", a 2D platform game that incorporates gaze-based interactions (via *UnitEye*) and traditional keyboard input (see Figure 7.1). The game explores the trade-offs between enjoyment and digital eye strain through a five-day longitudinal study involving 17 participants. The findings indicate that while gaze-based interaction led to higher perceived competence, it also resulted in increased internal eye strain.

Usage of UnitEye. UnitEye is used as an input for "Shed Some Fear" to enable gaze-based mechanisms as a core interaction method. The players' gaze position is shown as a bat, acting as a companion. It enables six types of gaze-based interactions: "Move/Hold on Gaze" for block manipulation, "Reveal on Gaze" for lighting areas, "Repel on Gaze" to ward off enemies, "Damage on Gaze" for harming opponents, "Indirect Damage/Heal on Gaze" for affecting game objects, and two "Follow with Gaze" variants requiring rapid gaze changes or smooth tracking. Players who didn't own an eye tracker, used *UnitEye* as a cost-effective eye tracking method to be able to play the game at home.

5.2 Use Case 2: Remote Operation

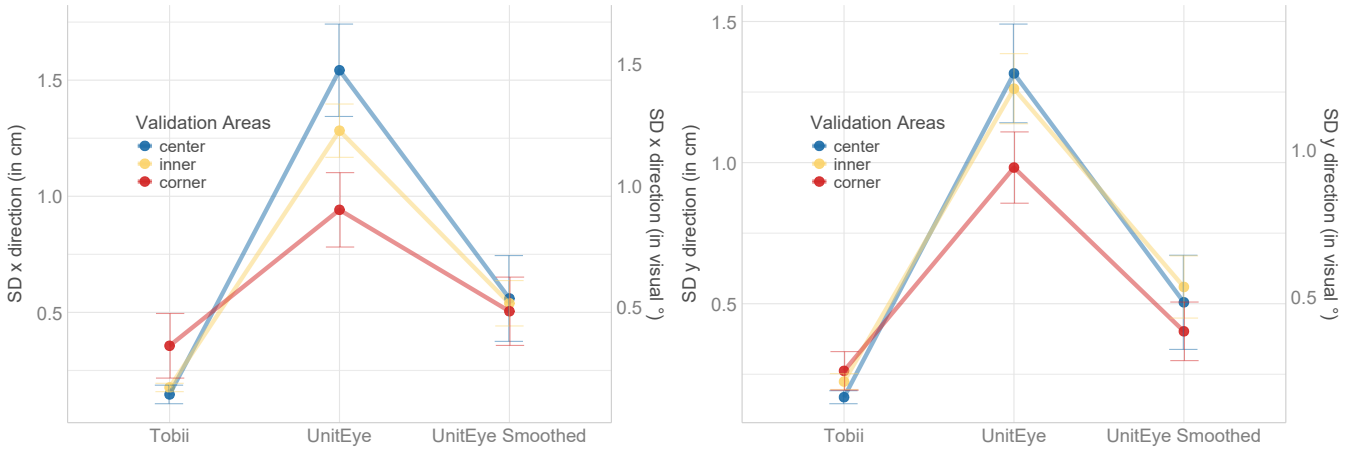
The study designs, implements, and evaluates three interaction interfaces (*path planning*, *trajectory guidance*, and *waypoint guidance*) for human operators intervening in automated vehicles (AVs), finding that *path planning* was the most preferred and usable, while *trajectory guidance* performed the poorest in resolving requests, thus contributing to the development of human-machine interfaces for remote assistance in AVs.



(a) Euclidean distance per validation area.

(b) RMS-S2S value per validation area.

Figure 5: Euclidean distance and RMS-S2S value in cm (left y-axis) and degrees of visual angle (right y-axis).



(a) SD in the x direction per validation area.

(b) SD in the y direction per validation area.

Figure 6: SD in cm (left y-axis) and degrees of visual angle (right y-axis) for x and y direction.

Usage of UnitEye. In this lab-based study, *UnitEye* was employed to evaluate the interface by recording participants' gaze behavior while they interacted with interaction concepts for remote operation (the task of two parallel requests can be seen in Figure 7.2).

5.3 Use Case 3: Longitudinal Effects of Automotive User Interfaces

In this study, the effect of different information reduction strategies for UIs in vehicles (see Figure 7.3) was evaluated. This work is still ongoing.

Usage of UnitEye. UnitEye is used to evaluate the gaze behavior of participants when assessing the UIs.

5.4 Use Case 4: Bayesian Optimization Study

The paper introduces OptiCarVis, a human-in-the-loop system that uses multi-objective Bayesian optimization to fine-tune AV feedback visualizations (see Figure 7.4), demonstrating through an online study that this approach significantly improves key factors like trust and perceived safety while reducing cognitive load.

Usage of UnitEye. UnitEye was leveraged to assess participants' eye movements while they were inspecting visualizations, which were optimized using the multi-objective Bayesian optimization. They had to install the simulation on their personal computers.

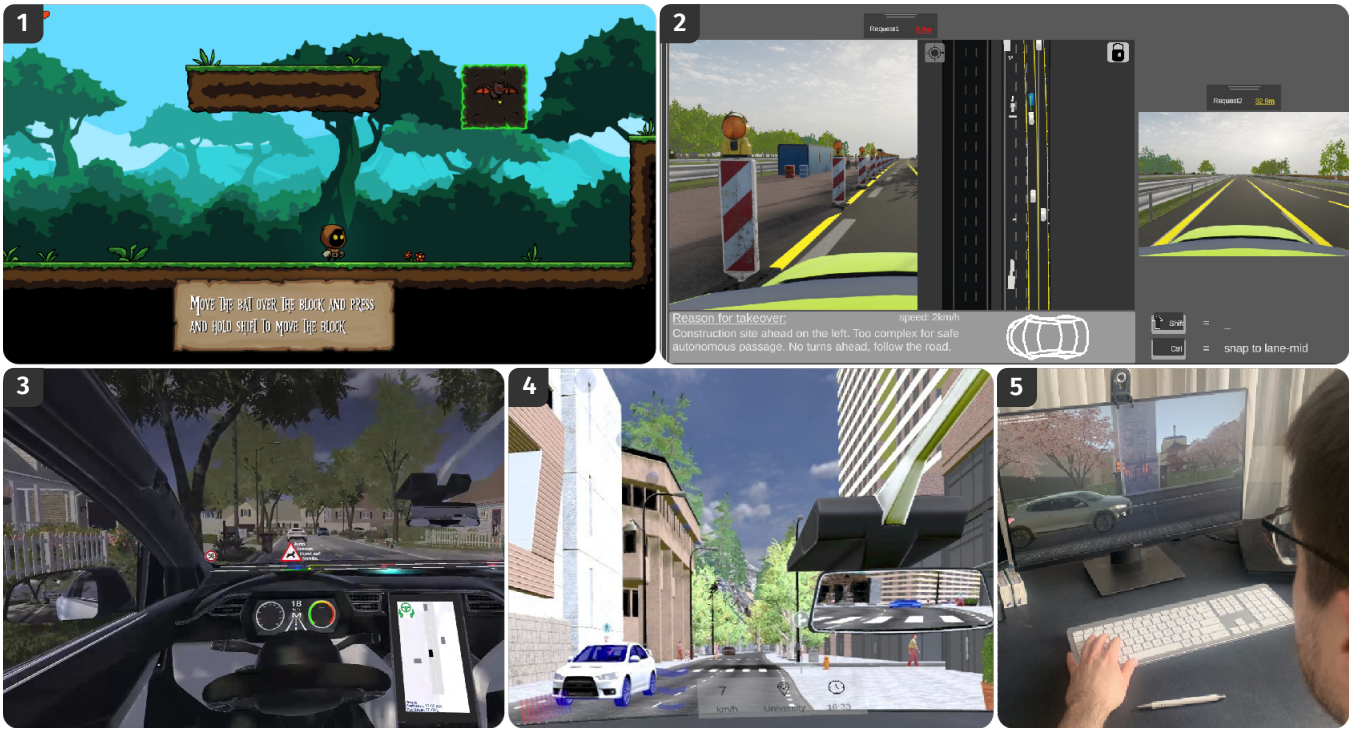


Figure 7: This figure shows an overview of the five case studies. In (1), a level of "Shed Some Fear" is shown where *UnitEye* was used to control a companion bat. (2) shows the UI for the remote operation of an automated vehicle where *UnitEye* was used for eye tracking analysis, which was also done for the longitudinal study in (3), the human-in-the-loop system OptiCarVis (4), and for the cultural comparison of external communication (5). Figures are reprinted with permission.

5.5 Use Case 5: Cultural Effects of External Communication of Automated Vehicles

The study investigates the impact of external Human-Machine Interfaces (eHMI) on pedestrian safety and behavior around AVs, utilizing LED strips for communication. Two experimental designs were used: a Virtual Reality study in Germany and an online study involving North American and European subjects. The findings suggest that frequent interaction with eHMIs improves pedestrian trust, understanding, and perceived safety while reducing mental workload, with North Americans taking higher crossing risks and experiencing lower workloads.

Usage of UnitEye. A web-based simulation was built, including *UnitEye*, to log the eye gaze during the crossing scenarios (see Figure 7.5). These measurements were then sent to the server.

5.6 Participants

The five developers (Mean age = 25.80, SD = 1.90, range: [23, 28]; Gender: 40.0% women, 60.0% men, 0.00% non-binary; Education: high school, 20.00%; university degree, 80.00%) of the case studies took part in the survey as participants. They had medium experience in software projects (M=5.60 years, SD=1.52) and little experience with eye tracking (M=1.40 projects finished, SD=1.14). At the time of the questionnaire, four participants were students and one was employed.

5.7 Measurements and Procedure

We asked participants to rate the Task Load Index via the NASA-TLX [21] on a 20-point scale. We omitted the subscale physical demand due to its inapplicability in this context. Participants also rated the usability via the system usability scale (SUS) [9]. Finally, we evaluated acceptance using the van der Laan acceptance scale [49], which measures acceptance on the two subscales Usefulness and Satisfying. Afterward, participants were able to give open qualitative feedback:

- What did you find particularly negative about using UnitEye? Please be detailed.
- What did you notice most positive about UnitEye? Please be detailed.
- What other functionality should UnitEye offer?
- What is your opinion about the performance of UnitEye?
- If you have experience with other eye tracking software for development (i.e., SDKs) - how do you rate UnitEye? Please mention concrete manufacturers. If you have no other experience, please write NA.
- You can write more about UnitEye here.

5.8 Results

5.8.1 Questionnaires. Table 2 shows the descriptive results regarding task load, usability, usefulness, and satisfaction of *UnitEye*. For the NASA-TLX scores, which can range from 1–20, we find that all

values were medium. Usability was satisfactory. The acceptance scales Usefulness and Satisfying, which range from -3 to +3 were on the positive side.

5.8.2 Negative Aspects About UnitEye. Key challenges are identified across different themes: UI, Calibration and Accuracy, Documentation, Debugging, Error Handling, and System Integration.
UI: Users mentioned challenges in accessing functions through the provided UI, suggesting that an easier-to-reach interface or additional documentation might be beneficial.
Calibration and Accuracy: Calibration was found to be cumbersome, time-consuming, and tiring for the eyes. Post-calibration, the system's accuracy was still questioned, particularly in the Y-direction, despite seemingly good evaluation metrics. Additionally, there were inconsistencies in drowsiness detection based on eye-opening degrees compared to calibration settings.
Debugging and Error Handling: Users noted inconsistencies in displaying areas of interest (AOIs) for debugging and mentioned difficulties with error handling, particularly in webcam detection and activity status, especially with multiple webcams.
System Integration: There were issues with object tracking, notably of smaller objects and untagged objects being erroneously tracked. Moreover, the system was reported to be sensitive to the user's proximity to the camera.
Miscellaneous: The settings were described as quite confusing for those unfamiliar, and the CSV log was cited as unclear when alterations were needed. There were also specific problems mentioned with multiple Unity layers overlapping, and the system's performance was impaired when a second webcam (virtual or not) was connected.

5.8.3 Positive Aspects About UnitEye. The following aspects were highlighted as positive in the text fields.
Ease of Deployment and Integration: The system is acknowledged for its simplicity in deployment and integration into existing or finished projects. Users found it straightforward to install and utilize once familiarized with the setup process. The quick integration and a good example provided were appreciated.
Tracking Performance: Horizontal tracking was found to perform surprisingly well, considering technical limitations associated with webcam-based systems. The system's performance in pose, hand, face recognition, and joint accuracy, even during head rotation, was commended, alongside effective blinking detection.
UI and Documentation: The runtime UI was noted as very helpful for initial familiarization with the system. The ReadMe file is appreciated for its clarity and thoroughness, contributing to an easier integration into a project. The clarity in field names displayed in the inspector was also mentioned positively.
Cost-effective Alternative: The system is considered a good, budget-friendly alternative to more expensive eye tracking solutions. The feature allowing the display of the user's eyes was appreciated, enhancing user interaction.
Miscellaneous: Users value the clear and non-confusing naming conventions in the inspector, which, together with effective recognition functionalities in the `Visualization.cs` script, contribute to a positive user experience.

5.8.4 Functionality Requests and Performance. Users desire improvements in the system's interface and accuracy, particularly around camera selection and eye position tracking. They wish for a simplified interface to switch cameras during runtime, addressing issues faced with virtual cameras. They also seek more accurate eye position tracking, visible feedback when no CSV-log file is created, a better indication of which webcam is in use, and a feature for layer prioritization (as available in Tobii) to enhance precision.

The performance feedback centers around calibration success, system speed, and tracking accuracy. Users found horizontal tracking satisfactory but faced challenges with vertical tracking and, on older or possibly less powerful hardware, experiencing slow responses or non-functionality. The calibration process was often viewed as lengthy and exhausting, more so on slower machines. They observed the system to be error-prone, especially with slight user movements, affecting tracking accuracy. Nonetheless, some found the program to run smoothly on any laptop without persistent low frame rates, indicating a level of flexibility in system requirements. Despite these positives, the point of inaccuracy post-calibration, particularly regarding the crosshair's position, was a common concern, hinting at a desire for more effective and perhaps varied calibration options to enhance tracking precision. Based on this feedback, we added Pursuit calibration as described by Pfeuffer et al. [39].

Variable	Min	q_{1}	$\tilde{x}$	$\bar{x}$	q_{3}	Max	s	IQR
Mental Demand [21]	6.00	7.00	13.00	10.60	13.00	14.00	3.78	6.00
Temporal Demand [21]	2.00	4.00	9.00	7.60	10.00	13.00	4.51	6.00
Performance [21]	4.00	5.00	12.00	9.60	13.00	14.00	4.72	8.00
Effort [21]	6.00	8.00	12.00	10.60	13.00	14.00	3.44	5.00
Frustration [21]	8.00	10.00	12.00	13.00	17.00	18.00	4.36	7.00
Usability [9]	17.50	50.00	57.50	54.00	67.50	77.50	22.89	17.50
Usefulness [49]	0.00	1.00	1.00	1.08	1.60	1.80	0.70	0.60
Satisfying [49]	-1.25	-0.50	0.00	-0.05	0.50	1.00	0.87	1.00

Table 2: Descriptive statistics for the numeric dependent variables. The number in brackets represents the used questionnaire (see Section 5.7).

6 DISCUSSION, LIMITATIONS, AND FUTURE WORK

This work presented *UnitEye*, a user-friendly plugin to democratize eye tracking in Unity environments. We presented *UnitEye*'s features and implementation, and we report the technical evaluation in comparison to a current state-of-the-art eye tracker with a video-based pupil and corneal reflection eye tracking technique. Future work could enhance our work into a benchmark study with an evaluation using the Bland-Altman analysis [7]. Finally, we report five case studies that have used *UnitEye* successfully. The case studies showed that *UnitEye* is easily integrable, useful, and accurate enough when using smoothing algorithms. This notion is also supported by our technical evaluation (see Figure 5 and Figure 6) and the determined minimum AOI size (see Table 1). Despite its capabilities, there are still limitations and improvement possibilities. Currently, the use of *UnitEye* is limited to scenarios where a lower

degree of accuracy is sufficient, such as passive eye monitoring with large AOIs, or attentive user interfaces [34, 57].

Enhancing *UnitEye* includes several advancements. Improvements in vision algorithms can increase performance, which currently does not match the sub-centimeter accuracy of state-of-the-art eye trackers. This is supported by *UnitEye*'s modular structure, which allows for further integration of additional estimation models, calibration methods, and filters. Regarding hardware, webcams with a higher frame rate and resolution can further increase estimation performance [25, p. 66]. Moreover, although *UnitEye* operates using only a single webcam, it requires broader testing across various devices. Future iterations might also explore sensor fusion, integrating RGB and depth camera data for enhanced gaze tracking (e.g., see [4]). We currently avoid the necessity for depth cameras to allow a more widespread adoption. As Unity allows porting to Android and iOS, future work should also evaluate the performance of *UnitEye* on smartphones and tablets.

7 CONCLUSION

In this work, *UnitEye*, a Unity plugin designed to make eye tracking technology more accessible, has been introduced. Our framework employs cutting-edge methods to match the performance found in previous works when applying smoothing. Our technical evaluation indicates an average accuracy of *UnitEye*, measured in Euclidean distance, of about 3.8 cm (3.67°) and a comparable precision, measured in RMS-S2S, to a state-of-the-art eye tracker, which is accurate enough for users to engage with large interface elements on laptops and monitors, as shown in five case studies. We anticipate that with further refinements, our approach could operate inconspicuously in the background, paving the way for innovative gaze-based opportunities where dedicated eye trackers are missing.

OPEN SCIENCE

UnitEye and all evaluation scripts and data are openly available under https://github.com/wgnrto/uniteye. This also includes a preliminary Bland-Altman analysis.

ACKNOWLEDGMENTS

We thank all study participants. This project was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation, 425867974, to Anke Huckauf and Enrico Rukzio) and is part of Priority Program SPP2199 Scalable Interaction Paradigms for Pervasive Computing Environments.

REFERENCES

[1] Karan Ahuja, Ruchika Banerjee, Seema Nagar, Kuntal Dey, and Ferdous Barbhuiya. 2016. Eye center localization and detection using radial mapping. In *2016 IEEE International Conference on image processing (ICIP)*. IEEE, New York, NY, USA, 3121–3125.

[2] Karan Ahuja, Rahul Islam, Varun Parashar, Kuntal Dey, Chris Harrison, and Mayank Goel. 2018. Eyespyvr: Interactive eye sensing using off-the-shelf, smartphone-based vr headsets. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 2 (2018), 1–10.

[3] Andronicus A. Akinyelu and Pieter Blignaut. 2020. Convolutional Neural Network-Based Methods for Eye Gaze Estimation: A Survey. *IEEE access : practical innovations, open solutions* 8 (2020), 142581–142605. https://doi.org/10.1109/ACCESS.2020.3013540

[4] Riku Arakawa, Mayank Goel, Chris Harrison, and Karan Ahuja. 2022. RGBDGaze: Gaze Tracking on Smartphones with RGB and Depth Data. In *Proceedings of the 2022 International Conference on Multimodal Interaction* (Bengaluru, India) *(ICMI '22)*. Association for Computing Machinery, New York, NY, USA, 329–336. https://doi.org/10.1145/3536221.3556568

[5] Aakash Bansal, Bonita Sharif, and Collin McMillan. 2023. Towards Modeling Human Attention from Eye Movements for Neural Source Code Summarization. *Proc. ACM Hum.-Comput. Interact.* 7, ETRA, Article 167 (may 2023), 19 pages. https://doi.org/10.1145/3591136

[6] Kenan Bektaş, Jannis Strecker, Simon Mayer, Dr. Kimberly Garcia, Jonas Hermann, Kay Erik Jenß, Yasmine Sheila Antille, and Marc Solèr. 2023. GEAR: Gaze-Enabled Augmented Reality For Human Activity Recognition. In *Proceedings of the 2023 Symposium on Eye Tracking Research and Applications* (Tubingen, Germany) *(ETRA '23)*. Association for Computing Machinery, New York, NY, USA, Article 9, 9 pages. https://doi.org/10.1145/3588015.3588402

[7] J Martin Bland and DouglasG Altman. 1986. Statistical methods for assessing agreement between two methods of clinical measurement. *The lancet* 327, 8476 (1986), 307–310.

[8] Pieter Blignaut and Daniël Wium. 2014. Eye-Tracking Data Quality as Affected by Ethnicity and Experimental Design. *Behavior Research Methods* 46, 1 (March 2014), 67–80. https://doi.org/10.3758/s13428-013-0343-0

[9] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[10] Benjamin T. Carter and Steven G. Luke. 2020. Best Practices in Eye Tracking Research. *International Journal of Psychophysiology* 155 (Sept. 2020), 49–62. https://doi.org/10.1016/j.ijpsycho.2020.05.010

[11] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1 € Filter: A Simple Speed-Based Low-Pass Filter for Noisy Input in Interactive Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2527–2530. https://doi.org/10.1145/2207676.2208639

[12] Mark Colley, Pascal Jansen, Enrico Rukzio, and Jan Gugenheimer. 2022. SwiVR-Car-Seat: Exploring Vehicle Motion Effects on Interaction Quality in Virtual Reality Automated Driving Using a Motorized Swivel Seat. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 4, Article 150 (dec 2022), 26 pages. https://doi.org/10.1145/3494968

[13] Mark Colley, Luca-Maxim Meinhardt, Alexander Fassbender, Michael Rietzler, and Enrico Rukzio. 2023. Come Fly With Me: Investigating the Effects of Path Visualizations in Automated Urban Air Mobility. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 2, Article 52 (jun 2023), 23 pages. https://doi.org/10.1145/3596249

[14] Mark Colley, Bastian Wankmüller, and Enrico Rukzio. 2022. A Systematic Evaluation of Solutions for the Final 100m Challenge of Highly Automated Vehicles. *Proc. ACM Hum.-Comput. Interact.* 6, MHCI, Article 178 (sep 2022), 19 pages. https://doi.org/10.1145/3546713

[15] Mark Colley, Beate Wanner, Max Rädler, Marcel Rötzer, Julian Frommel, Teresa Hirzle, Pascal Jansen, and Enrico Rukzio. 2024. Effects of a Gaze-Based 2D Platform Game on User Enjoyment, Perceived Competence, and Digital Eye Strain. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)* (Honolulu, HI, USA, May 11–16). ACM, New York, NY, USA, 14. https://doi.org/10.1145/3613904.3641909

[16] Shuwen Deng, David R. Reich, Paul Prasse, Patrick Haller, Tobias Scheffer, and Lena A. Jäger. 2023. Eyettention: An Attention-Based Dual-Sequence Model for Predicting Human Scanpaths during Reading. *Proc. ACM Hum.-Comput. Interact.* 7, ETRA, Article 162 (may 2023), 24 pages. https://doi.org/10.1145/3591131

[17] Yoshinobu Ebisawa. 1998. Improved video-based eye-gaze detection method. *IEEE Transactions on instrumentation and measurement* 47, 4 (1998), 948–955.

[18] Kenneth Alberto Funes Mora, Florent Monay, and Jean-Marc Odobez. 2014. EYEDIAP: A Database for the Development and Evaluation of Gaze Estimation Algorithms from RGB and RGB-D Cameras. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (Safety Harbor, Florida) *(ETRA '14)*. Association for Computing Machinery, New York, NY, USA, 255–258. https://doi.org/10.1145/2578153.2578190

[19] Anuj Harisinghani, Harshinee Sriram, Cristina Conati, Giuseppe Carenini, Thalia Field, Hyeju Jang, and Gabriel Murray. 2023. Classification of Alzheimer's Using Deep-learning Methods on Webcam-based Gaze Data. *Proceedings of the ACM on Human-Computer Interaction* 7, ETRA (May 2023), 157:1–157:17. https://doi.org/10.1145/3591126

[20] Warren Harrison. 2006. Eating your own dog food. *IEEE Software* 23, 3 (2006), 5–7.

[21] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, Amsterdam, The Netherlands, 139–183.

[22] Teresa Hirzle, Jan Gugenheimer, Florian Geiselhart, Andreas Bulling, and Enrico Rukzio. 2019. A Design Space for Gaze Interaction on Head-Mounted Displays. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300855

[23] Teresa Hirzle, Marian Sauter, Tobias Wagner, Susanne Hummel, Enrico Rukzio, and Anke Huckauf. 2022. Attention of Many Observers Visualized by Eye Movements. In *2022 Symposium on Eye Tracking Research and Applications* (Seattle,

WA, USA) *(ETRA '22)*. Association for Computing Machinery, New York, NY, USA, Article 65, 7 pages. https://doi.org/10.1145/3517031.3529235

[24] Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: applications to nonorthogonal problems. *Technometrics* 12, 1 (1970), 69–82.

[25] Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost van de Weijer. 2011. *Eye Tracking: A Comprehensive Guide to Methods and Measures*. OUP Oxford.

[26] Shoya Ishimaru, Kai Kunze, Koichi Kise, Jens Weppner, Andreas Dengel, Paul Lukowicz, and Andreas Bulling. 2014. In the Blink of an Eye: Combining Head Motion and Eye Blink Frequency for Activity Recognition with Google Glass. In *Proceedings of the 5th Augmented Human International Conference* (Kobe, Japan) *(AH '14)*. Association for Computing Machinery, New York, NY, USA, Article 15, 4 pages. https://doi.org/10.1145/2582051.2582066

[27] Takehiro Ito, Shinji Mita, Kazuhiro Kozuka, Tomoaki Nakano, and Shin Yamamoto. 2002. Driver blink measurement by the motion picture processing and its application to drowsiness detection. In *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*. IEEE, New York, NY, USA, 168–173.

[28] Andy Kong, Karan Ahuja, Mayank Goel, and Chris Harrison. 2021. EyeMU Interactions: Gaze + IMU Gestures on Mobile Devices. In *Proceedings of the 2021 International Conference on Multimodal Interaction* (Montréal, QC, Canada) *(ICMI '21)*. Association for Computing Machinery, New York, NY, USA, 577–585. https://doi.org/10.1145/3462244.3479938

[29] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba. 2016. Eye Tracking for Everyone. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, New York, NY, USA, 2176–2184.

[30] Aleksandra Królak and Paweł Strumiłło. 2012. Eye-blink detection system for human–computer interaction. *Universal Access in the Information Society* 11 (2012), 409–419.

[31] Gustav Kuhn, Benjamin W Tatler, John M Findlay, and Geoff G Cole. 2008. Misdirection in magic: Implications for the relationship between eye gaze and attention. *Visual Cognition* 16, 2-3 (2008), 391–405.

[32] Chih-Chuan Lai, Yu-Ting Chen, Kuan-Wen Chen, Shen-Chi Chen, Sheng-Wen Shih, and Yi-Ping Hung. 2014. Appearance-based gaze tracking with free head movement. In *2014 22nd International Conference on Pattern Recognition*. IEEE, New York, NY, USA, 1869–1873.

[33] David A Mackey, Colleen H Wilkinson, Lisa S Kearns, and Alex W Hewitt. 2011. Classification of iris colour: review and refinement of a classification schema. *Clinical & experimental ophthalmology* 39, 5 (2011), 462–471.

[34] Päivi Majaranta and Andreas Bulling. 2014. Eye Tracking and Eye-Based Human–Computer Interaction. In *Advances in Physiological Computing*, Stephen H. Fairclough and Kiel Gilleade (Eds.). Springer, London, 39–65. https://doi.org/10.1007/978-1-4471-6392-3_3

[35] Omar Namnakani, Penpicha Sinrattanavong, Yasmeen Abdrabou, Andreas Bulling, Florian Alt, and Mohamed Khamis. 2023. GazeCast: Using Mobile Devices to Allow Gaze-Based Interaction on Public Displays. In *Proceedings of the 2023 Symposium on Eye Tracking Research and Applications* (Tubingen, Germany) *(ETRA '23)*. Association for Computing Machinery, New York, NY, USA, Article 92, 8 pages. https://doi.org/10.1145/3588015.3589663

[36] Diederick C. Niehorster, Raimondas Zemblys, Tanya Beelders, and Kenneth Holmqvist. 2020. Characterizing Gaze Position Signals and Synthesizing Noise during Fixations in Eye-Tracking Data. *Behavior Research Methods* 52, 6 (Dec. 2020), 2515–2534. https://doi.org/10.3758/s13428-020-01400-9

[37] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediyana Daskalova, Jeff Huang, and James Hays. 2016. WebGazer: Scalable Webcam Eye Tracking Using User Interactions. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI, 3839–3845.

[38] Yesaya Tommy Paulus and Gerard Bastiaan Remijn. 2021. Usability of various dwell times for eye-gaze-based object selection with eye tracking. *Displays* 67 (2021), 101997.

[39] Ken Pfeuffer, Melodie Vidal, Jayson Turner, Andreas Bulling, and Hans Gellersen. 2013. Pursuit Calibration: Making Gaze Calibration Less Tedious and More Flexible. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13)*. Association for Computing Machinery, New York, NY, USA, 261–270. https://doi.org/10.1145/2501988.2501998

[40] Robert Rosenthal, Harris Cooper, and L Hedges. 1994. Parametric measures of effect size. *The handbook of research synthesis* 621, 2 (1994), 231–244.

[41] Tsugunosuke Sakai, Harunya Tamaki, Yosuke Ota, Ryohei Egusa, Shigenori Imagaki, Fusako Kusunoki, Masanori Sugimoto, and Hiroshi Mizoguchi. 2017. Eda-based estimation of visual attention by observation of eye blink frequency. *International Journal on Smart Sensing and Intelligent Systems* 10, 2 (2017), 1–12.

[42] Marian Sauter, Tobias Wagner, Teresa Hirzle, Bao Xin Lin, Enrico Rukzio, and Anke Huckauf. 2023. Behind the Screens: Exploring Eye Movement Visualization to Optimize Online Teaching and Learning. In *Proceedings of Mensch Und Computer 2023* (Rapperswil, Switzerland) *(MuC '23)*. Association for Computing Machinery, New York, NY, USA, 67–80. https://doi.org/10.1145/3603555.3603560

[43] Rongkai Shi, Yushi Wei, Xueying Qin, Pan Hui, and Hai-Ning Liang. 2023. Exploring Gaze-Assisted and Hand-Based Region Selection in Augmented Reality. *Proc. ACM Hum.-Comput. Interact.* 7, ETRA, Article 160 (may 2023), 19 pages. https://doi.org/10.1145/3591129

[44] Rainer Stiefelhagen, Jie Yang, and Alex Waibel. 1997. A model-based gaze tracking system. *International Journal on Artificial Intelligence Tools* 6, 02 (1997), 193–209.

[45] Dean Takahashi. 2022. Unity report: Number of games made with Unity grew 93% in 2021. https://venturebeat.com/games/unity-report-number-of-games-made-with-unity-grew-93-in-2021/. [Online; accessed 12-DECEMBER-2023].

[46] Philipp Terhörst, Jan Niklas Kolf, Marco Huber, Florian Kirchbuchner, Naser Damer, Aythami Morales Moreno, Julian Fierrez, and Arjan Kuijper. 2022. A Comprehensive Study on Face Recognition Biases Beyond Demographics. *IEEE Transactions on Technology and Society* 3, 1 (March 2022), 16–30. https://doi.org/10.1109/TTS.2021.3111823

[47] UNI-T. 2023. UT383/UT383BT Mini Light Meters. https://meters.uni-trend.com/product/ut383-ut383bt/. [Online; accessed 12-DECEMBER-2023].

[48] Nachiappan Valliappan, Na Dai, Ethan Steinberg, Junfeng He, Kantwon Rogers, Venky Ramachandran, Pingmei Xu, Mina Shojaeizadeh, Li Guo, Kai Kohlhoff, et al. 2020. Accelerating eye movement research via accurate and affordable smartphone eye tracking. *Nature communications* 11, 1 (2020), 4553.

[49] Jinke D. Van Der Laan, Adriaan Heino, and Dick De Waard. 1997. A simple procedure for the assessment of acceptance of advanced transport telematics. *Transportation Research Part C: Emerging Technologies* 5, 1 (1997), 1–10. https://doi.org/10.1016/S0968-090X(96)00025-3

[50] Greg Welch, Gary Bishop, et al. 1995. An introduction to the Kalman filter. SIGGRAPH course.

[51] Katarzyna Wisiecka, Krzysztof Krejtz, Izabela Krejtz, Damian Sromek, Adam Cellary, Beata Lewandowska, and Andrew Duchowski. 2022. Comparison of Webcam and Remote Eye Tracking. In *2022 Symposium on Eye Tracking Research and Applications* (Seattle, WA, USA) *(ETRA '22)*. Association for Computing Machinery, New York, NY, USA, Article 32, 7 pages. https://doi.org/10.1145/3517031.3529615

[52] Katarzyna Wisiecka, Krzysztof Krejtz, Izabela Krejtz, Damian Sromek, Adam Cellary, Beata Lewandowska, and Andrew Duchowski. 2022. Comparison of Webcam and Remote Eye Tracking. In *2022 Symposium on Eye Tracking Research and Applications*. ACM, Seattle WA USA, 1–7. https://doi.org/10.1145/3517031.3529615

[53] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The Aligned Rank Transform for Nonparametric Factorial Analyses Using Only Anova Procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 143–146. https://doi.org/10.1145/1978942.1978963

[54] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. 2016. Learning an Appearance-Based Gaze Estimator from One Million Synthesised Images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications* (Charleston, South Carolina) *(ETRA '16)*. Association for Computing Machinery, New York, NY, USA, 131–138. https://doi.org/10.1145/2857491.2857492

[55] Erroll Wood and Andreas Bulling. 2014. EyeTab: Model-Based Gaze Estimation on Unmodified Tablet Computers. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '14)*. Association for Computing Machinery, New York, NY, USA, 207–210. https://doi.org/10.1145/2578153.2578185

[56] Pingmei Xu, Krista A Ehinger, Yinda Zhang, Adam Finkelstein, Sanjeev R Kulkarni, and Jianxiong Xiao. 2015. Turkergaze: Crowdsourcing saliency with webcam based eye tracking.

[57] Xucong Zhang, Yusuke Sugano, and Andreas Bulling. 2019. Evaluation of Appearance-Based Methods and Implications for Gaze-Based Applications. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300646

[58] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2019. MPIIGaze: Real-World Dataset and Deep Appearance-Based Gaze Estimation. *IEEE transactions on pattern analysis and machine intelligence* 41, 1 (Jan. 2019), 162–175. https://doi.org/10.1109/TPAMI.2017.2778103