GE-Simulator: An Open-Source Tool for Simulating Real-Time Errors for HMD-based Eye Trackers

Ludwig Sidenmark
l.sidenmark@lancaster.ac.uk
Lancaster University
Lancaster, United Kingdom

Mathias N. Lystbæk
mathiasl@cs.au.dk
Aarhus University
Aarhus, Denmark

Hans Gellersen
h.gellersen@lancaster.ac.uk
Lancaster University
Lancaster, United Kingdom
Aarhus University
Aarhus, Denmark

ABSTRACT

As eye tracking in augmented and virtual reality (AR/VR) becomes established, it will be used by broader demographics, increasing the likelihood of tracking errors. Therefore, it is important when designing eye tracking applications or interaction techniques to test them at different signal quality levels to ensure they function for as many people as possible. We present GE-Simulator, a novel open-source Unity toolkit that allows the simulation of accuracy, precision, and data loss errors during real-time usage by adding gaze vector errors into the gaze vector from the head-mounted AR/VR eye tracker. The tool is customisable without having to change the source code and changes in eye tracking errors during and in-between usage. Our toolkit allows designers to prototype new applications at different levels of eye tracking in the early phases of design and can be used to evaluate techniques with users at varying signal quality levels.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; *Mixed / augmented reality*; *Virtual reality*; User interface toolkits.

KEYWORDS

Eye Tracking, Data Quality, Toolkit, Accuracy, Precision, Data Loss

ACM Reference Format:
Ludwig Sidenmark, Mathias N. Lystbæk, and Hans Gellersen. 2023. GE-Simulator: An Open-Source Tool for Simulating Real-Time Errors for HMD-based Eye Trackers. In *2023 Symposium on Eye Tracking Research and Applications (ETRA '23), May 30-June 2, 2023, Tubingen, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3588015.3588417

1 INTRODUCTION

Eye tracking is becoming an increasingly prevalent input modality for use within augmented and virtual reality (AR/VR) applications. Due to its speed and because users naturally look at objects they are interested in, gaze has been proposed as an attractive modality for interaction and inference [David-John et al. 2021; Kytö et al. 2018; Sidenmark and Gellersen 2019]. However, tracking gaze is not precise or easy, especially in the diverse settings exposed by mobile head-mounted displays (HMDs). Eye trackers commonly require a calibration process to map captured eye images to gaze directions. Poor or missing calibrations can cause *accuracy error* (that is, the difference between actual and recorded gaze directions) that can significantly change the gaze direction. Furthermore, our constantly moving eyes and their natural jitter introduces a natural level of noise within gaze signals that can increase due to tracking difficulties [Holmqvist et al. 2011]. Such *precision error* affect the stable inference of the gaze position and also algorithms with dispersion or velocity thresholds to detect gaze movements [Salvucci and Goldberg 2000] or interaction based on relative eye movements [Sidenmark et al. 2020a]. Finally, eye tracking devices require a clear view of the eyes, factors such as lighting and glasses can cause significant precision and accuracy issues, and even *data loss* [Holmqvist et al. 2011], thus making online usage of gaze data difficult [Holmqvist et al. 2011, 2012; Norouzi et al. 2019].

Many researchers report the accuracy and precision determined by the manufacturer [Akkil et al. 2014; Holmqvist et al. 2012], but these metrics are typically calculated under ideal conditions, which are not representative of in-the-wild usage. Furthermore, in user studies with online eye tracking data, participants with poor data quality are commonly discarded [Ahn et al. 2020; Feit et al. 2017; Norouzi et al. 2019]. These decisions may lead to a limited understanding of gaze-based interaction techniques and applications since scenarios with ideal eye tracking are assumed. Furthermore, if eye tracking error was not considered during the early development of applications, results may be worse than expected during evaluation as a wider demographic is employed.

Previous research has shown that eye tracking errors on the gaze vector can have a significantly negative impact on the usability of such applications [Feit et al. 2017; Norouzi et al. 2019; Sidenmark et al. 2022]. Furthermore, if eye tracking error is not considered during the development phase, there is a risk of accidentally making gaze-based applications inaccessible for users with poor eye tracking. HMD-based eye trackers are becoming more pervasive and will be used in various contexts, increasing the chance of tracking in non-ideal conditions. During the design phase of gaze-based interaction techniques, eye tracking errors should be an early consideration and tested on various quality levels to predict users' performance in the wild. Furthermore, different works on AR/VR-based eye tracking have simulated vector-based eye tracking errors with different methods, making comparison of studies

ETRA '23, May 30-June 2, 2023, Tubingen, Germany

ACM ISBN 979-8-4007-0150-4/23/05…$15.00
https://doi.org/10.1145/3588015.3588417

difficult [Graupner et al. 2008; Mughrabi et al. 2022; Norouzi et al. 2019; Sidenmark and Gellersen 2019]. Together, these factors highlight the necessity of a tool that allows easy use of eye tracking errors during the development of gaze applications. Standardisation of eye tracking errors also ensures that the method of error simulation is clear and reproducible.

To address these issues, we contribute a novel open-source Unity tool, Gaze Error Simulator (*GE-Simulator*)[1], that allows easy simulation of eye tracking errors commonly found in gaze vectors for HMD-based eye tracking. GE-Simulator can be used in both the design and evaluation of techniques to give a more comprehensive understanding of gaze-based techniques and applications at various levels of data quality. GE-Simulator is designed to be universal by supporting the most commonly found HMD-based eye trackers used for research. The methods of simulation are based on previous works that simulate eye tracking errors of gaze vectors and enable replicable simulation for comparison between different research projects. The goal of GE-Simulator is to enable developers to consider gaze errors early in the design phase of gaze-based applications to make eye tracking more accessible.

2 RELATED WORK

2.1 Types of Eye Tracking Error and Impact

Measuring and handling gaze signal error is an important aspect of eye tracking and gaze-based interaction. In this context, *accuracy* refers to the average difference between the true and measured gaze directions and provides a measure of the quality of the calibration and gaze-mapping procedure [Holmqvist et al. 2012]. Accuracy is calculated as the angular difference (θ) between the gaze ray reported and an imaginary gaze ray projected from its origin onto the target stimulus (Equation 1). Precision quantifies the ability of the eye tracker to reliably reproduce a given result, regardless of the intended gaze location, and represents an aggregate of system-inherent, oculomotor, and environmental noise [Holmqvist et al. 2011]. Precision is commonly calculated using the root mean square (RMS) of the intersample angular (α) differences (Equation 2). For HMD-based eye trackers, intersample angles can be calculated by measuring the angle between successive gaze ray samples reported by the eye tracking SDK.

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}\theta_i \tag{1}$$

$$Precision_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\alpha_i^2\right)} \tag{2}$$

Finally, *data loss*, is defined as when a tracker is unable to output a gaze direction, which can be caused by a poorly aligned tracker or by user factors such as eye shape and glasses [Holmqvist et al. 2011]. Data loss is commonly calculated as the percentage of lost data points over a defined time window. Accuracy, precision and data loss can have a significant negative impact on gaze-based interaction and may vary between tracking areas [Feit et al. 2017] or over time due to changing lighting conditions, or slippage of a head-mounted display in AR and VR settings [Drewes et al. 2012; Holmqvist et al. 2011; Niehorster et al. 2020].

Eye tracking errors and their impact on the user experience within the domain of AR and VR have recently received significant attention. Previous work has shown that precision error can have a significant impact on gaze-based target selection [Graupner et al. 2008; Mughrabi et al. 2022]. Similarly for accuracy errors, significant errors would cause the gaze position to be outside the target area [Erickson et al. 2020; Graupner et al. 2008; Norouzi et al. 2019]. Accuracy error has received significant attention within the HCI field, where researchers are using a second, more accurate modality to refine gaze positions [Kytö et al. 2018; Sidenmark et al. 2020b; Zhai et al. 1999]. These results highlight the importance of being mindful of eye tracking errors early in the design process of novel gaze applications, which we enable through our toolkit.

2.2 Error-aware Systems

The impact of eye tracking errors has led to the development of error-aware gaze-based systems that store current eye tracking calibration data that contain measurements of all gaze signal errors. These recordings are then leveraged to adjust the interaction by correcting the gaze direction [Barz et al. 2016, 2018; Fares et al. 2013], expanding the user interface widgets that are placed in areas where the eye tracking signal is poor [Feit et al. 2017], or using fallback modalities [Sidenmark et al. 2022]. Alternatively, incorrect interactions are detected via brain-computer interfaces and corrected [Kalaganis et al. 2018]. Our toolkit allows users to prototype error-aware systems with artificial data without the burden of collecting real eye tracking data at every step of the process.

2.3 Open-Source Tools

Several open-source tools have been developed for remote eye trackers to make error measurements accessible and reproducible. These tools are used for validation of data quality [Akkil et al. 2014], assessment of data quality under non-ideal conditions [Clemotte et al. 2014] or with more difficult populations [Dalrymple et al. 2018]. Most recently proposed was *GazeMetrics* by B. Adhanom et al. [2020], which enables a rigorous and standardised method to measure eye tracking errors for HMD-based eye trackers. These toolkits emphasise the need to standardise eye tracking error measurements. While these tools focus on measuring errors, we focus on the simulation of errors to enable easy prototyping and development with errors in mind to make eye tracking more accessible.

3 SYSTEM DESCRIPTION

GE-Simulator is a standalone package that allows online simulation of eye tracking errors. The main benefit of GE-Simulator is that users can easily simulate eye tracking of different quality during every phase of design, development, and evaluation of gaze-based applications. The gaze errors are fully customisable without modification to the source code. All changes can be saved as default settings that can easily be swapped between each other.

3.1 Technical Specifications

GE-Simulator is a software package built in Unity. The toolkit provides built-in support for the Unity software development kits

[1]https://github.com/ludwan/GE-Simulator

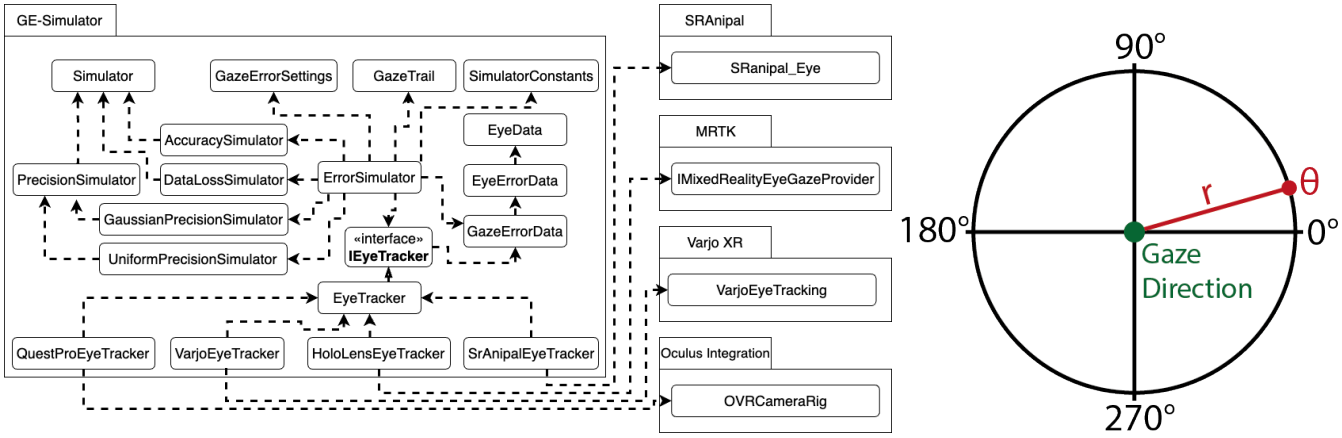

Figure 1: Left: GE-Simulator high-level class diagram. Right: The user defines an offset direction (θ) and amplitude (r) to add to the original gazer vector (green) to simulate accuracy and precision errors.

(SDKs) of multiple AR and VR head-mounted eye trackers. These SDKs, in turn, support multiple eye tracking platforms. The toolkit was built using the provider model design patterns, an extensible software design pattern that allows users to add their own support of eye tracking SDKs to the source code to support more eye trackers and applications. Figure 1 shows a high-level overview of GE-Simulator. Each main component is discussed in detail in the following subsections.

The toolkit supports eye tracking from multiple manufacturers. The toolkit includes built-in support for four eye tracking SDKs, including the native VIVE SRAnipal SDK for VIVE Pro Eye, the Varjo XR SDK, which can be used with the Varjo XR-3, Varjo VR-3, and Varjo Aero, the Oculus Integration SDK, which enables Meta Quest Pro to be used, including eye tracking, and the MRTK SDK, which enables support of Microsoft HoloLens 2. The toolkit was tested for functionality on Unity version 2020.3.42f using the eye tracking SDKs: Vive SRAnipal SDK v1.3.3.0, Varjo XR SDK v3.3.0, Oculus Integration SDK v47.0[2] and MRTK v2.8.2. The development and testing of the system were carried out on Windows 10.

3.2 Error Simulation

GE-Simulator simulates eye tracking errors based on the gaze data provided by the selected eye tracker's Unity SDK. HMD-based eye trackers commonly report gaze data in the form of a ray or origin and direction 3D vectors from which a ray can be created. The ray then originates from the head or eye position and is directed in the user's gaze direction [Duchowski et al. 2002]. Calculating an accurate ray can be a challenging problem, widely covered in the literature [Pfeiffer et al. 2008]. These issues showcase the importance of our toolkit, which allows testing eye tracking-based applications at various levels of eye tracking data quality.

GE-Simulator simulates eye tracking errors by "adding" errors to the ray provided by the eye tracking SDK. The toolkit supports three types of eye tracking error – data loss, accuracy error, and precision error. The methods of simulating errors are based on previous work that has investigated the effect of online gaze errors on user experience and performance. For each type of error, we define our simulation method and how users can adjust the error with the toolkit.

3.2.1 Data Loss. Similarly to previous work, we define data loss as the probability of lost gaze data due to the eye tracker not tracking the eye and thus not outputting a gaze ray [Norouzi et al. 2019]. To implement this error, for every data point the eye tracker outputs, the toolkit measures the chance that the data point is overwritten as "lost" based on a value (d) set by the designer. For example, if d is set to 0.5, there is a 50% chance that the data will be lost. If an output is defined as lost, we do not apply the following accuracy and precision errors.

3.2.2 Accuracy Error. Accuracy error represents a persistent angular offset between the true gaze direction and the measured gaze direction. To implement this error, we rotate every measured gaze ray with a constant angular amplitude in a specified rotation (Figure 1). The user defines the rotation in polar coordinates relative to the measured gaze ray. As such, the user specifies a rotation (θ) and an amplitude (r). Rotation (θ) is expressed in degrees ($0 - 359°$) where $0°$ represents the rightwards head direction, rotating anticlockwise (Figure 1). The amplitude (r) is expressed in visual degrees and represents the amplitude of the rotation in the direction θ. In Unity, the toolkit translates r and θ into a quaternion applied to the measured gaze ray. This allows users to easily define the direction and amplitude of the accuracy errors.

3.2.3 Precision Error. Precision error represents a dynamic angular difference between the true gaze direction and the measured gaze direction. Commonly this dynamic error is calculated for every gaze point, by sampling direction and amplitude values from predefined distributions. The direction (θ) is sampled from a uniform distribution (in our case, a value between $0 - 359°$) [Graupner et al. 2008]. However, the amplitude of the precision error has previously been sampled from both uniform [Norouzi et al. 2019], and Gaussian [Graupner et al. 2008; Mughrabi et al. 2022; Sidenmark et al. 2022] distributions. Therefore, GE-Simulator includes both *"Uniform"* and *"Gaussian"* modes for simulating the precision error.

[2]With the Oculus XR Plugin v2.2.3-preview.2.

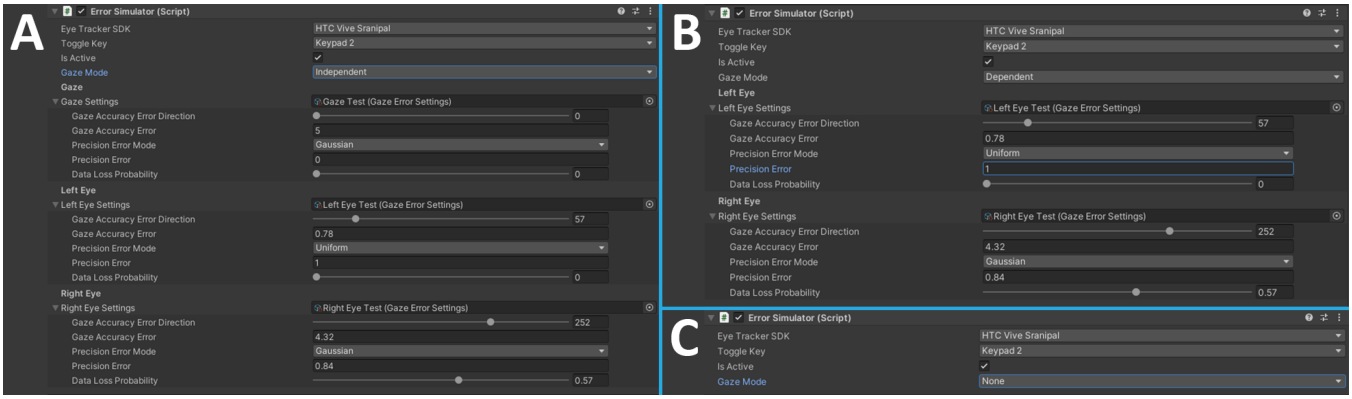

Figure 2: Easy-to-use settings allow the designer to customize various aspects of GE-Simulator. This figure shows the settings windows for each error mode (A: Independent Mode, B: Dependent Mode, C: No Error Mode).

In Uniform mode, the user defines an upper threshold value (α) for the amplitude of the precision error. r is then sampled from a range between 0 and α using the Unity function RANDOM.RANGE, which is based on the Xorshift algorithm [Marsaglia 2003]. In Gaussian mode, the user defines the standard deviation (σ) of the Gaussian distribution. We then sample x, and y from the Gaussian distribution (mean = 0, standard deviation = σ), which are translated into polar coordinates (θ, r). As in previous work [Mughrabi et al. 2022], the Gaussian distribution is approximated with the Marsaglia-polar method [Marsaglia and Bray 1964].

3.3 Adding the Eye Tracking Error

After generating the values for data loss, accuracy error, and precision error, we add them to the gaze ray. Accuracy and precision errors are only added if the data is not deemed lost. The accuracy error is first added, and the precision error is added to the gaze ray with the added accuracy error.

Previous work that has simulated eye tracking errors generally only does so on the combined gaze ray [Mughrabi et al. 2022; Sidenmark et al. 2022]. However, it may be of interest to add errors to only one eye to simulate users whose eye tracking only works on one eye or for applications which use the individual gaze signal from each eye. To facilitate customisation, GE-Simulator provides different *"error modes"* (Figure 2) that define the relationship between the errors of each eye and the combined gaze ray:

Independent Mode: In Independent mode, the user defines the probability of data loss, accuracy error, and precision errors separately for each eye. There is no relationship between the different gaze signals.

Dependent Mode: In this mode, the user defines all errors for the left and right eyes, and the respective error is added to each eye. The gaze signal is then calculated as the mean of the eye signals. If the signal from only one eye is available, the gaze ray will be equal to the valid eye ray. If the data from both eyes are deemed lost, so is the combined gaze data. This mode is not functional for eye trackers, which only output the combined gaze ray (i.e., HoloLens 2).

Table 1: Averages and standard deviations of data quality metric values across all participants for each condition.

Metric	Normal	AccSim	PrecSim	LossSim
Accuracy (°)	1.49 ± 0.82	6.00 ± 1.14	3.61 ± 0.25	1.49 ± 0.84
Precision - RMS (°)	0.29 ± 0.17	0.15 ± 0.07	4.92 ± 0.05	0.13 ± 0.10
Data Loss (%)	8.52 ± 4.10	5.20 ± 2.92	1.91 ± 1.22	51.84 ± 2.63

No Error Mode: No errors are added to the data. This allows users to integrate GE-Simulator to their gaze data pipeline with the option of not introducing additional errors without having to manually adjust multiple gaze error parameters.

3.4 Evaluation

To evaluate and confirm the functionality of GE-Simulator, we performed a small study where we collected sample data from four users (2 female, 2 male, 24.58±4.25 years age) with varying levels of eye tracking error. Participants all had normal vision without correction and had previous occasional experience with eye tracking and VR. For this purpose, we used the HTC Vive Pro Eye headset and its SRAnipal SDK. We integrated GE-Simulator with the *GazeMetrics* toolkit, which records data quality for HMD-based eye trackers through a typical calibration procedure [B. Adhanom et al. 2020]. We used the preset 9 targets placed in a circular arrangement with a radius of 0.3 metres at a 1-metre distance. We displayed each target for 2 seconds and excluded the first 500 ms from the calculation. The participants performed this procedure four times with varying errors introduced:

Normal: No added eye tracking error.
AccSim: 5° added accuracy error in the 0° direction.
PrecSim: Added Gaussian precision error with $\sigma = 5°$
LossSim: Added 50% probability of data loss.

All errors were applied to the gaze ray in independent mode. Error order was counterbalanced with a balanced Latin square. Results can be found in Table 1. The results show that our toolkit successfully adds each type of error to the gaze signal. Note that due to the way accuracy error is calculated (Equation 1), accuracy error

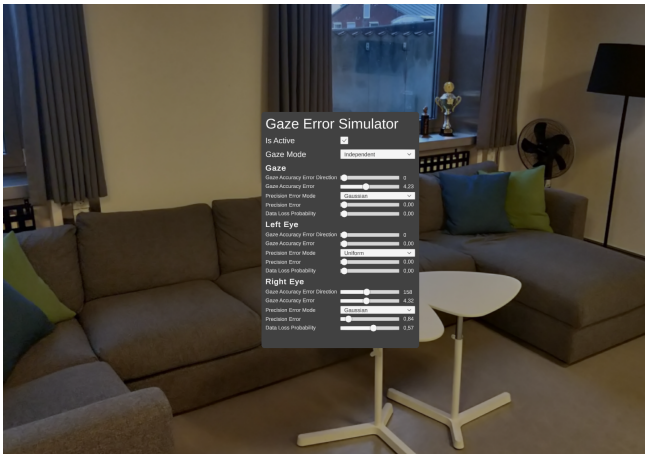

Figure 3: In-game menu of GE-Simulator where error parameters can be adjusted during runtime.

increase with an increase in added precision error. Furthermore, note that this study is aimed at merely confirming the functionality of our toolkit and should not be treated as an empirical study of the eye tracker.

4 SYSTEM USAGE

GE-Simulator is designed as an easy-to-use Unity package that can be added to any Unity project. The toolkit includes events that can be subscribed for easy integration into the eye tracking pipeline, whether for simple prototyping of new interaction techniques or applications or as part of formal user studies.

GE-Simulator includes all the source code and assets necessary for usage. The user simply has to add the toolkit, which can be found on its GitHub page[3] and the SDK of their eye tracker. Once the package has been added, the developer has to add the "GE-Controller" or "GE-Menu" prefab to the project scene by dragging and dropping it in the hierarchy window. The GE-Controller prefab enables changes to the error simulation via the editor window (Figure 2). Users can change the eye tracker, how to activate the error simulator (via a Unity Action), error mode, and, depending on the error mode, the error parameters of the gaze and individual eyes. Error parameters are implemented as scriptable objects, which means that error parameters can be saved as presets that can be dragged and dropped into GE-Controller.

The GE-Menu prefab also includes a menu added to the virtual environment (Figure 3). This allows developers to adjust parameters without removing the HMD to change settings in the Unity editor. The in-game menu is implemented via the standard Unity UI package, and the toolkit uses the same provider pattern as eye trackers so that users can define how to interact with the menu.

Finally, the toolkit includes a gaze trail script which visualizes the erroneous or original gaze trails during use (Figure 4). Users simply define which eye (gaze, right, left) and signal to trace (original, erroneous). Multiple traces can be added without issue.

[3]https://github.com/ludwan/GE-Simulator

Figure 4: Gaze trail of GE-Simulator to visualize gaze.

5 LIMITATIONS AND FUTURE WORK

GE-Simulator only approximates gaze errors, and using eye tracking in the wild may lead to different error behaviours and thus results for any designed application or interaction technique. Furthermore, our error simulation could be expanded. The errors included in the toolkit focus on gaze vector errors. The toolkit could be expanded to include errors on the pupil size, and latency of the eye tracker. Furthermore, the included errors could also be expanded. For example, our data loss calculation does not support the simulation of long periods of data loss caused by misalignment of the eye tracker or blinks. In addition, data loss is often accompanied by artefacts causing significant shifts in the data [Holmqvist et al. 2011]. These aspects could be supported in future versions of the toolkit. In addition, previous research has shown that eye trackers tend to be more accurate when users are looking straight ahead compared to glancing [Feit et al. 2017]. Being able to define areas of the tracking area with different levels of eye tracking error could be a beneficial addition. Furthermore, our precision simulation assumes symmetrical distributions. However, previous work has shown that noise does not have to be symmetrical [Feit et al. 2017]. Being able to define asymmetric precision simulation would make the error simulation more akin to eye tracking in the wild. Finally, a significant limitation of our work is that we only tested a part of the toolkit's functionality on users and had a relatively small sample size. While we are confident in its functionality, the effect of the toolkit on user studies has not been formally evaluated.

6 CONCLUSION

Data quality is a vital component for the accessible and frictionless use of gaze. Therefore it is important to consider data quality and how to handle poor data early in the design phase when developing gaze-based applications and interaction techniques. GE-Simulator is open-source software designed to enable developers to easily incorporate eye tracking errors as early as possible during development. GE-Simulator's fully customisable user interface makes the toolkit accessible for users with various coding experiences, and the in-game menu allows rapid prototyping of different eye tracking error levels. As eye tracking for HMD becomes more prevalent and used in various contexts, considering poor data quality will become increasingly important. We aim to continue the support of GE-Simulator to extend functionality and include more and future HMD eye trackers to help designers make eye tracking more accessible for users.

ACKNOWLEDGMENTS

This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant No. 101021229, GEMINI: Gaze and Eye Movement in Interaction) and by the Innovation Fund Denmark, as part of the Manufacturing Academy of Denmark (MADE) FAST project.

REFERENCES

Sunggeun Ahn, Jeongmin Son, Sangyoon Lee, and Geehyuk Lee. 2020. Verge-It: Gaze Interaction for a Binocular Head-Worn Display Using Modulated Disparity Vergence Eye Movement. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. ACM, New York, NY, USA, 1–7. https://doi.org/10.1145/3334480.3382908

Deepak Akkil, Poika Isokoski, Jari Kangas, Jussi Rantala, and Roope Raisamo. 2014. TraQuMe: A Tool for Measuring the Gaze Tracking Quality. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (Safety Harbor, Florida) *(ETRA '14)*. Association for Computing Machinery, New York, NY, USA, 327–330. https://doi.org/10.1145/2578153.2578192

Isayas B. Adhanom, Samantha C. Lee, Eelke Folmer, and Paul MacNeilage. 2020. GazeMetrics: An Open-Source Tool for Measuring the Data Quality of HMD-Based Eye Trackers. In *ACM Symposium on Eye Tracking Research and Applications* (Stuttgart, Germany) *(ETRA '20 Short Papers)*. Association for Computing Machinery, New York, NY, USA, Article 19, 5 pages. https://doi.org/10.1145/3379156.3391374

Michael Barz, Florian Daiber, and Andreas Bulling. 2016. Prediction of Gaze Estimation Error for Error-Aware Gaze-Based Interfaces. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications* (Charleston, South Carolina) *(ETRA '16)*. ACM, New York, NY, USA, 275–278. https://doi.org/10.1145/2857491.2857493

Michael Barz, Florian Daiber, Daniel Sonntag, and Andreas Bulling. 2018. Error-Aware Gaze-Based Interfaces for Robust Mobile Gaze Interaction. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications* (Warsaw, Poland) *(ETRA '18)*. ACM, New York, NY, USA, Article 24, 10 pages. https://doi.org/10.1145/3204493.3204536

A. Clemotte, M. Velasco, D. Torricelli, R. Raya, and R. Ceres. 2014. Accuracy and Precision of the Tobii X2-30 Eye-tracking under Non Ideal Conditions. In *Proceedings of the 2nd International Congress on Neurotechnology, Electronics and Informatics - NEUROTECHNIX,*. INSTICC, SciTePress, Setúbal, Portugal, 111–116. https://doi.org/10.5220/0005094201110116

Kirsten A. Dalrymple, Marie D. Manner, Katherine A. Harmelink, Elayne P. Teska, and Jed T. Elison. 2018. An Examination of Recording Accuracy and Precision From Eye Tracking Data From Toddlerhood to Adulthood. *Frontiers in Psychology* 9 (2018), 12 pages. https://doi.org/10.3389/fpsyg.2018.00803

Brendan David-John, Candace Peacock, Ting Zhang, T. Scott Murdison, Hrvoje Benko, and Tanya R. Jonker. 2021. Towards Gaze-Based Prediction of the Intent to Interact in Virtual Reality. In *ACM Symposium on Eye Tracking Research and Applications* (Virtual Event, Germany) *(ETRA '21 Short Papers)*. ACM, New York, NY, USA, Article 2, 7 pages. https://doi.org/10.1145/3448018.3458008

Jan Drewes, Guillaume S. Masson, and Anna Montagnini. 2012. Shifts in Reported Gaze Position Due to Changes in Pupil Size: Ground Truth and Compensation. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (Santa Barbara, California) *(ETRA '12)*. ACM, New York, NY, USA, 209–212. https://doi.org/10.1145/2168556.2168596

Andrew Duchowski, Eric Medlin, Nathan Cournia, Hunter Murphy, Anand Gramopadhye, Santosh Nair, Jeenal Vorah, and Brian Melloy. 2002. 3-D eye movement analysis. *Behavior Research Methods, Instruments, & Computers* 34, 4 (01 Nov 2002), 573–591. https://doi.org/10.3758/BF03195486

Austin Erickson, Nahal Norouzi, Kangsoo Kim, Joseph J. LaViola, Gerd Bruder, and Gregory F. Welch. 2020. Effects of Depth Information on Visual Target Identification Task Performance in Shared Gaze Environments. *IEEE Transactions on Visualization and Computer Graphics* 26, 5 (2020), 1934–1944. https://doi.org/10.1109/TVCG.2020.2973054

Ribel Fares, Shaomin Fang, and Oleg Komogortsev. 2013. Can We Beat the Mouse with MAGIC?. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. ACM, New York, NY, USA, 1387–1390. https://doi.org/10.1145/2470654.2466183

Anna Maria Feit, Shane Williams, Arturo Toledo, Ann Paradiso, Harish Kulkarni, Shaun Kane, and Meredith Ringel Morris. 2017. Toward Everyday Gaze Input: Accuracy and Precision of Eye Tracking and Implications for Design. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 1118–1130. https://doi.org/10.1145/3025453.3025599

Sven-Thomas Graupner, Michael Heubner, Sebastian Pannasch, and Boris M. Velichkovsky. 2008. Evaluating Requirements for Gaze-Based Interaction in a See-through Head Mounted Display. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications* (Savannah, Georgia) *(ETRA '08)*. Association for Computing Machinery, New York, NY, USA, 91–94. https://doi.org/10.1145/1344471.1344495

Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Jarodzka Halszka, and Joost van de Weijer. 2011. *Eye Tracking : A Comprehensive Guide to Methods and Measures*. Oxford University Press, Oxford, United Kingdom. 560 pages.

Kenneth Holmqvist, Marcus Nyström, and Fiona Mulvey. 2012. Eye Tracker Data Quality: What It is and How to Measure It. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (Santa Barbara, California) *(ETRA '12)*. ACM, New York, NY, USA, 45–52. https://doi.org/10.1145/2168556.2168563

Fotis P. Kalaganis, Elisavet Chatzilari, Spiros Nikolopoulos, Ioannis Kompatsiaris, and Nikos A. Laskaris. 2018. An error-aware gaze-based keyboard by means of a hybrid BCI system. *Scientific Reports* 8, 1 (04 Sep 2018), 13176. https://doi.org/10.1038/s41598-018-31425-2

Mikko Kytö, Barrett Ens, Thammathip Piumsomboon, Gun A. Lee, and Mark Billinghurst. 2018. Pinpointing: Precise Head- and Eye-Based Target Selection for Augmented Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/3173574.3173655

George Marsaglia. 2003. Xorshift RNGs. *Journal of Statistical Software* 8, 14 (2003), 1–6. https://doi.org/10.18637/jss.v008.i14

G. Marsaglia and T. A. Bray. 1964. A Convenient Method for Generating Normal Variables. *SIAM Rev.* 6, 3 (1964), 260–264. https://doi.org/10.1137/1006063

Moaaz Hudhud Mughrabi, Aunnoy K Mutasim, Wolfgang Stuerzlinger, and Anil Ufuk Batmaz. 2022. My Eyes Hurt: Effects of Jitter in 3D Gaze Tracking. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)* (Christchurch, New Zealand). IEEE, 310–315. https://doi.org/10.1109/VRW55335.2022.00070

Diederick C. Niehorster, Thiago Santini, Roy S. Hessels, Ignace T. C. Hooge, Enkelejda Kasneci, and Marcus Nyström. 2020. The impact of slippage on the data quality of head-worn eye trackers. *Behavior Research Methods* 52, 3 (01 Jun 2020), 1140–1160. https://doi.org/10.3758/s13428-019-01307-0

Nahal Norouzi, Austin Erickson, Kangsoo Kim, Ryan Schubert, Joseph LaViola, Gerd Bruder, and Greg Welch. 2019. Effects of Shared Gaze Parameters on Visual Target Identification Task Performance in Augmented Reality. In *Symposium on Spatial User Interaction* (New Orleans, LA, USA) *(SUI '19)*. ACM, New York, NY, USA, Article 12, 11 pages. https://doi.org/10.1145/3357251.3357587

Thies Pfeiffer, Marc E. Latoschik, and Ipke Wachsmuth. 2008. Evaluation of Binocular Eye Trackers and Algorithms for 3D Gaze Interaction in Virtual Reality Environments. *JVRB - Journal of Virtual Reality and Broadcasting* 5(2008), 16 (2008), 14 pages. https://doi.org/10.20385/1860-2037/5.2008.16

Dario D. Salvucci and Joseph H. Goldberg. 2000. Identifying Fixations and Saccades in Eye-Tracking Protocols. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications* (Palm Beach Gardens, FL, USA) *(ETRA '00)*. ACM, New York, NY, USA, 71–78. https://doi.org/10.1145/355017.355028

Ludwig Sidenmark, Christopher Clarke, Xuesong Zhang, Jenny Phu, and Hans Gellersen. 2020a. Outline Pursuits: Gaze-Assisted Selection of Occluded Objects in Virtual Reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376438

Ludwig Sidenmark and Hans Gellersen. 2019. Eye&Head: Synergetic Eye and Head Movement for Gaze Pointing and Selection. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. ACM, New York, NY, USA, 1161–1174. https://doi.org/10.1145/3332165.3347921

Ludwig Sidenmark, Diako Mardanbegi, Argenis Ramirez Gomez, Christopher Clarke, and Hans Gellersen. 2020b. BimodalGaze: Seamlessly Refined Pointing with Gaze and Filtered Gestural Head Movement. In *ACM Symposium on Eye Tracking Research and Applications* (Stuttgart, Germany) *(ETRA '20 Full Papers)*. ACM, New York, NY, USA, Article 8, 9 pages. https://doi.org/10.1145/3379155.3391312

Ludwig Sidenmark, Mark Parent, Chi-Hao Wu, Joannes Chan, Michael Glueck, Daniel Wigdor, Tovi Grossman, and Marcello Giordano. 2022. Weighted Pointer: Error-aware Gaze-based Interaction through Fallback Modalities. *IEEE Transactions on Visualization and Computer Graphics* 28, 11 (2022), 3585–3595. https://doi.org/10.1109/TVCG.2022.3203096

Shumin Zhai, Carlos Morimoto, and Steven Ihde. 1999. Manual and Gaze Input Cascaded (MAGIC) Pointing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Pittsburgh, Pennsylvania, USA) *(CHI '99)*. ACM, New York, NY, USA, 246–253. https://doi.org/10.1145/302979.303053