GazeMetrics: An Open-Source Tool for Measuring the Data Quality of HMD-based Eye Trackers

Isayas B. Adhanom
Dept. of Computer Science
University of Nevada
iadhanom@nevada.unr.edu

Samantha C. Lee
Integrative Neuroscience
University of Nevada
samanthalee@nevada.unr.edu

Eelke Folmer
Dept. of Computer Science
University of Nevada
efolmer@unr.edu

Paul MacNeilage
Dept. of Psychology
University of Nevada
pmacneilage@unr.edu

ABSTRACT

As virtual reality (VR) garners more attention for eye tracking research, knowledge of accuracy and precision of head-mounted display (HMD) based eye trackers becomes increasingly necessary. It is tempting to rely on manufacturer-provided information about the accuracy and precision of an eye tracker. However, unless data is collected under ideal conditions, these values seldom align with on-site metrics. Therefore, best practices dictate that accuracy and precision should be measured and reported for each study. To address this issue, we provide a novel open-source suite for rigorously measuring accuracy and precision for use with a variety of HMD-based eye trackers. This tool is customizable without having to alter the source code, but changes to the code allow for further alteration. The outputs are available in real time and easy to interpret, making eye tracking with VR more approachable for all users.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in HCI**; *Virtual reality.*

KEYWORDS

Eye tracking, eye tracker data quality, eye movements, accuracy, precision, virtual reality

ACM Reference Format:
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 *Symposium on Eye Tracking Research and Applications (ETRA '20 Short Papers), June 2–5, 2020, Stuttgart, Germany.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379156.3391374

1 INTRODUCTION

Eye tracking has been used as a research tool for decades; however, reliable and easy-to-use HMD-based eye trackers have only recently become widely available. While these newer devices often come with software designed to make eye tracking simple, the settings for collecting, filtering, and analyzing data are not standardized across devices [Feit et al. 2017; Holmqvist et al. 2011]. This makes it difficult to meaningfully compare data collected on different devices, under different circumstances, and by users with differing levels of experience [Dalrymple et al. 2018; Ehinger et al. 2019; Feit et al. 2017]. The most important measures of data quality that facilitate comparison across studies are spatial accuracy and precision.

Accuracy quantifies the average offset between the actual fixation location and the location of the intended target in units of visual angle (Eq. 1). It provides a measure of the quality of the calibration and gaze-mapping procedure. Precision quantifies the ability of the eye tracker to reliably reproduce a given result, regardless of intended gaze location; that is, it measures the end-to-end noise in the system (Eq. 2, 3). Precision therefore captures the aggregate of system-inherent, oculomotor, and environmental noise [Holmqvist et al. 2011]. Controlling for as many factors as possible allows for better calculation of precision [Clemotte et al. 2014; Tobii Technology 2011].

Many researchers report the manufacturer-determined accuracy and precision when publishing eye tracking data [Akkil et al. 2014; Blignaut and Beelders 2012; Dalrymple et al. 2018; Holmqvist et al. 2012], but these metrics are typically calculated under ideal conditions [Tobii Technology 2011]. If manufacturer metrics are used instead of actual measures, the data will be impaired and this can invalidate experimental results and conclusions [Dalrymple et al. 2018; Holmqvist et al. 2011; Nyström et al. 2013].

To address these issues, we have developed a novel open-source tool, GazeMetrics, that allows for the easy extraction of data samples and calculation of accuracy and precision. GazeMetrics is designed to be comprehensive and universal, while enabling objective, replicable measurements of accuracy and precision [Tobii Technology 2011]. The goal of the project is to enhance user experience, ease of measurement, and current software functionality.

2 RELATED WORK

Several open-source tools have been developed for remote eye trackers to make accuracy and precision measurements more reliable. 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], and extend the usability of eye tracking to other software, such as MATLAB [Gibaldi et al. 2017]. Tools have also been developed for wearable eye trackers to measure accuracy and precision [MacInnes 2018; Pfeiffer and Latoschik 2008] and to assess the effects of non-ideal stimulus presentation on these metrics [Kowalik 2011]. These studies emphasize the need to standardize accuracy and precision measurements.

While these tools focus on comprehensive measures of accuracy and precision, they use eye trackers that are not integrated with VR. VR presents its own challenges with calculating correct metrics

ETRA '20 Short Papers, June 2–5, 2020, Stuttgart, Germany
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7134-6/20/06…$15.00
https://doi.org/10.1145/3379156.3391374

across depth. Pfeiffer and Latoschik [Pfeiffer and Latoschik 2008] focus on applications of accuracy and precision in VR, such as depth fixation detection, but mention that their experimental set up is limited by the projection technology and the monitor-based display.

3 MOTIVATION

Data quality is vital to the comparability and standardization of experimental results when using eye tracking [Akkil et al. 2014; Blignaut and Beelders 2012; Ehinger et al. 2019; Holmqvist et al. 2012; Nyström et al. 2013]. Accuracy and precision measured on-site are almost always worse than what is expected based on manufacturer specifications [Akkil et al. 2014; Clemotte et al. 2014; Feit et al. 2017]. Furthermore, accuracy and precision will affect how data is collected and how well the results turn out.

During collection, these metrics contribute to factors such as data loss and detection of fixations, saccades, and other eye movement events [Holmqvist et al. 2011]. While applying filters to the data can help enhance precision, poor accuracy is more difficult to fix [Clemotte et al. 2014; Feit et al. 2017]. After data have been collected and filtered, any inaccuracy and imprecision will become apparent when assessing the results. While some higher-order measures are robust to inaccuracy and imprecision, most are adversely affected.

Additionally, accuracy and precision can be difficult to assess [Akkil et al. 2014; Blignaut and Beelders 2012; Niehorster et al. 2018]. Preprogrammed calibration procedures often do not show the outcome of the calibration to the user. For longer sessions, measurements begin to drift over time, causing a decay of data quality, sometimes up to 30% over the span of 4 minutes [Ehinger et al. 2019], which may not be noticeable until after the data collection.

Together, these factors highlight the necessity for a tool that is easy to implement and whose results are clear and reproducible. Standardization across devices allows researchers to compare results meaningfully and assess the actual effectiveness of their own data.

4 SYSTEM DESCRIPTION

4.1 General Overview

GazeMetrics is a stand-alone package that allows for online quantification of data quality. The most powerful feature is that nearly every functionality can be customized without modification to the source code, making it easy to use for users who are not necessarily experts in coding. All changes can be saved as default settings and multiple settings can be saved for implementation with various eye tracking studies.

4.2 Technical Specifications

GazeMetrics is a software package built in Unity. It provides built-in support for three popular eye trackers' Unity software development kits (SDKs), which in turn support multiple HMD-based eye tracking platforms. Moreover, to allow users to add their own implementations to the source code to support other devices or applications, the system was built using an extensible software design pattern, the provider model design pattern [Howard 2006]. Figure 1 shows a high-level class diagram of GazeMetrics. The components of the system will be discussed in detail in this section.

GazeMetrics comes with built-in support for the Tobii XR SDK from Tobii Pro, which can be used in conjunction with the HTC VIVE Pro Eye integrated eye tracker, the Pico Neo 2 Eye integrated eye tracker, and the Tobii Pro VR Integration eye tracker. In addition, the tool supports the Pupil Labs SDK, which works with the Pupil Labs add-on eye trackers that can be installed into the HTC VIVE, VIVE Pro, and VIVE Cosmos headsets. Finally, GazeMetrics supports the VIVE SRAnipal SDK, the native SDK for VIVE Pro Eye. Combined, they comprise most VR HMD-based eye trackers available on the market today.

GazeMetrics was tested for functionality on Unity version 2019.1.6, using the eye tracking SDKs: Tobii XR SDK v1.7.0.160, Pupil labs HMD eyes SDK v1.1 and Vive SRanipal SDK v1.1.0.1. Development and testing of the system was done in Windows 10.

4.3 Accuracy and Precision Calculation

GazeMetrics calculates accuracy and precision based on the eye gaze data provided by the selected eye tracker's SDK in Unity. Most HMD-based eye trackers report eye gaze data in the form of a gaze ray or origin and direction vectors from which a gaze ray can be constructed. The gaze ray is anchored at the position of the eye or the head and is directed in the eye gaze direction of the participant. This approach of representing eye gaze as a ray vector is widely used in the literature [Barabas et al. 2004; Duchowski et al. 2002]. Calculating an accurate 3D coordinate of a gaze point in the virtual environment (VE), however, is a challenging problem and is also extensively covered in previous literature [Pfeiffer and Latoschik 2008].

GazeMetrics uses the gaze ray reported by the eye tracking SDK to calculate inaccuracy, or offset, of each sample. This is calculated as the angular difference between the reported gaze ray and an imaginary gaze ray projected from its origin onto the target stimulus (θ_i). The average of all offset angles is taken to calculate the overall accuracy of the eye tracker:

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

$$\theta_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\theta_i^2\right)} = \sqrt{\frac{\theta_1^2 + \theta_2^2 + \ldots + \theta_n^2}{n}} \tag{2}$$

$$SD_{precision} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3}$$

Spatial precision metrics quantify variability in gaze measurements. There are two popular ways to measure precision: one involves calculating the root mean square (RMS) of the inter-sample angular distances (Eq. 2) and the other involves measuring the standard deviation of the samples (Eq. 3) [Holmqvist et al. 2012]. For HMD-based eye trackers, inter-sample angles can be calculated by measuring the angle between successive gaze ray samples reported by the eye tracking SDK. To calculate precision using standard deviation, data is needed about the x, y, and z coordinates of the gaze point in the VE. However, calculating the gaze point in the VE is a challenging problem, so not all eye tracking SDKs report gaze point coordinates in the VE. Therefore, the tool is designed to report only the RMS precision when gaze-point data is not available.

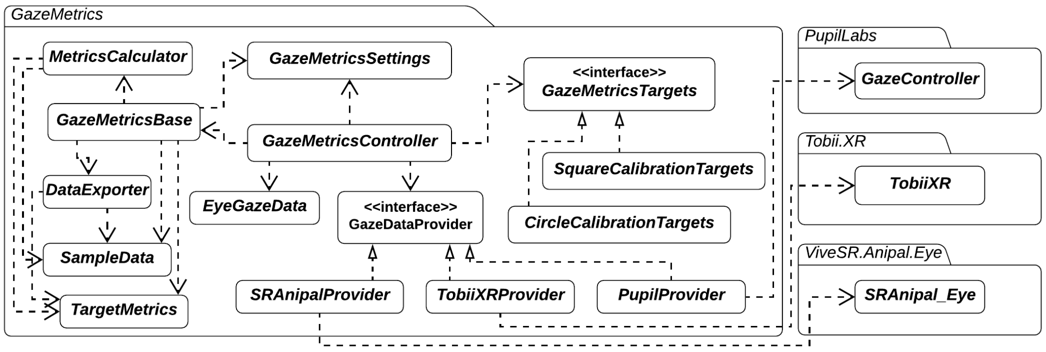


Figure 1: High-level UML class diagram of GazeMetrics.

When gaze-point data is available, precision via standard deviation is also calculated and reported separately in each direction. The methods that calculate accuracy and precision are defined in the `MetricsCalculator` class.

4.4 Stimulus Target Presentation

After selecting the SDK, the user has the option to alter the calibration targets from the user interface and to change several of the elements, including the presentation geometry (Fig. 2); target size, number, and color; and target display length, depth (distance from eyes), and eccentricity. There is also the option to change the background color that the targets will be presented on.

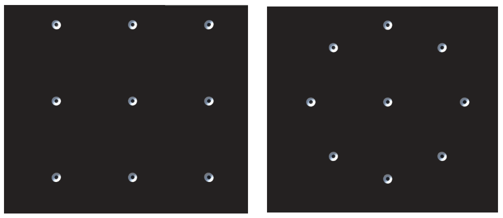


Figure 2: Calibration targets can be arranged in a rectangular format (a) or in a circular format (b).

Accuracy tends to be better if the conditions used in the initial calibration procedure are similar to the conditions used in the data quality test [Tobii Technology 2011]. However, different eye trackers use different calibration settings; for instance, Tobii eye trackers use a square calibration point arrangement (Fig. 2a) while Pupil Labs eye trackers use a circular arrangement (Fig. 2b). GazeMetrics allows the user to choose their desired arrangement by selecting one of the preset settings. Experienced users can also develop their own arrangement by implementing their own classes based on the `GazeMetricsTargets` abstract class.

Desired eccentricity of targets can also be manipulated for different experiments, such as testing foveal versus peripheral acuity. Users can set a different eccentricity, depth, and position of the central target for each arrangement of targets during a single run using settings on the user interface.

The settings can be previewed on the user's screen with a single button press to ensure desired placement before the participant interacts with the experiment. The `GazeMetricsSettings` class is used to create `ScriptableObjects`, which are used to create a data container that is used to store data that persists between sessions. Therefore GazeMetrics settings can be saved for use with other sessions and reported in the methods to allow other users to replicate the procedure.

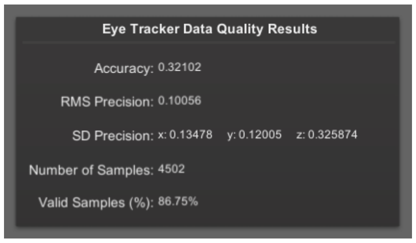
4.5 Data Collection and Reporting

GazeMetrics communicates with the selected eye trackers' Unity SDK to collect real-time data from the eye tracker. The sampling rate at which data is collected can be set by the experimenter using the settings window. It is recommended that the sampling rate be equal to the eye tracker's sampling frequency. GazeMetrics collects binocular data from the selected SDK, which is sufficient for most interactive and analytical uses. If users prefer to use monocular data for the precision and accuracy calculations, they can modify the `GazeDataProvider` implementation of their selected SDK, or write their own implementation of the `GazeDataProvider` interface.

In addition, there is an option to change how the data is collected and thus, how the metrics are calculated. For example, it is possible to include or exclude samples collected within certain time frames during target presentation, such as excluding the first 800ms and last 200ms to ensure fixation without including overshoots or glissades [Holmqvist et al. 2012; Nyström and Holmqvist 2010]. The targets can be displayed for a selected amount of time, so the amount of data collected from each target can be modified to suit the user's needs.

The tool reports results in two ways: 1) by displaying live results through a panel on the user's view of the VE, and 2) by storing raw data and experiment results on external files. The live data quality results panel (Fig. 3) allows the experimenter to check the data quality of the eye tracker under the current experimental conditions without interrupting the experiment.

For more detailed post-hoc analysis, GazeMetrics stores all collected and processed data in the form of CSV files in a storage location that the user chooses on the settings panel. While the data displayed on the results panel contain only aggregate data, calculated using the formulae in section 4.3, the data exported to the



Eye Tracker Data Quality Results

Accuracy: 0.32102

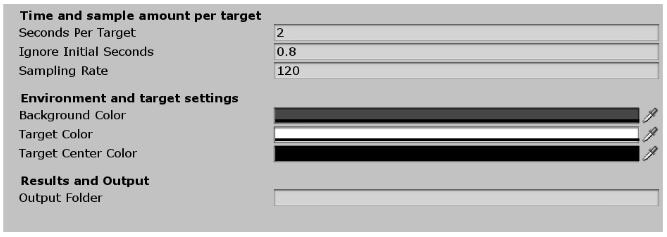
RMS Precision: 0.10056

SD Precision: x: 0.13478 y: 0.12005 z: 0.325874

Number of Samples: 4502

Valid Samples (%): 86.75%

Figure 3: Data quality measurement results panel used to display live results.



Time and sample amount per target

Seconds Per Target	2
Ignore Initial Seconds	0.8
Sampling Rate	120

Environment and target settings

Background Color	⬛
Target Color	⬜
Target Center Color	⬛

Results and Output

Output Folder	

Figure 4: Easy-to-use settings allow the experimenter to customize various aspects of the tool. This figure shows one of the settings windows for the tool.

files contain: raw gaze samples, the target's position on the VE, the participant's head position, the validity of the collected samples, and the time stamp at which each sample was collected. To allow users to have complete control of the data, the exported data includes all samples that were excluded from the online calculation; that is, it includes time frames that were originally excluded by user preference. Finally, the accuracy and precision for each target position are stored in a separate CSV file. The methods used to export GazeMetrics data are implemented in the `DataExporter` class.

To evaluate the functionality of GazeMetrics, we collected sample data from two users using the HTC Vive Pro Eye headset, and SRAnipal SDK. We used 9 targets with a circular target arrangement, with a radius of 0.3m. The targets were placed at a depth of 1m. We displayed each target for 2 seconds and excluded the first 500 ms from the calculation. After running the accuracy and precision tests with GazeMetrics, we obtained an average accuracy score of 1.23°, RMS precision score of 0.62°, and SD precision values of 3.95mm, 0.46mm, and 3.34mm respectively on the X, Y and Z directions. The manufacturer of the Vive Pro Eye reports spatial accuracy of $0.5° - 1.1°$ for the eye tracker. The manufacturer does not report spatial precision of the eye tracker. Our accuracy result is worse than the manufacturer reported accuracy. This underscores the importance of testing the data quality of eye trackers under the actual experiment conditions rather than relying on the manufacturer reported metrics. However, we would like to clarify that this test was performed to evaluate the functionality of the tool, and should not be taken as an empirical evaluation of the data quality of the tested eye tracker.

5 HOW TO USE THE TOOL

GazeMetrics is prepared as an easy-to-use Unity software package that can be added to a new or existing Unity software project. While it can be used as a stand-alone procedure to assess data quality, it can also be used inside an existing eye tracking experiment to verify data quality. GazeMetrics can be initiated by pressing a single key at any point during an eye tracking experiment. The tool can be run right after the initial calibration procedure or during the experiment to check if the data quality is high enough to continue.

GazeMetrics contains all the required source code and related assets. The user can download the latest version of the tool from its Github page [1]. Once the package has been imported to the assets folder, the developer needs to add the 'GazeMetricsController' prefab to the project by dragging-and-dropping it in the hierarchy window.

[1]https://github.com/isayasMatter/GazeMetrics

GazeMetrics comes with preset settings suitable for most experiments. However, the user can also change the settings by clicking on the "Gaze Metrics Settings" or the "Target settings" of the GazeMetricsController object. These settings are self-explanatory and contain pop-up tool-tips for ease of use. Users start by selecting which SDK (eye tracker) to use under the "Eye tracker type" setting and then changing elements of target presentation, data collection, and sampling rate.

After establishing the preferred settings, the procedure can be started by pressing the 'S' key anytime during the experiment. If the user wants to preview the actual positions of the target stimuli inside the VE, they can press the 'P' key to preview the target markers without initiating the procedure. Users have the ability to change the activation keys. Once the procedure is started the results panel will display live results. This panel is visible to the user, but is hidden from the participant in the VE. The tool displays useful status and log messages throughout the procedure to notify the experimenter and the participant about the status of the experimental procedure.

6 CONCLUSION

Accuracy and precision are two vital components of data quality that have implications for data collection, data analysis, and validity of results [Holmqvist et al. 2011, 2012]. As such, it is best practice to report correct metrics rather than relying on manufacturer data sheets and have these measures outlined within the methods of a study to ensure reproducible results. GazeMetrics is a free, open-source software made to address these issues. It has a user interface that is customizable without having to alter the source code, making it accessible for users with all levels of coding experience.

As eye tracking in VR develops, addressing the issue of standard accuracy and precision reporting becomes increasingly important. We plan to continue supporting GazeMetrics, and to extend the functionality of GazeMetrics to include other HMD VR devices, such as FOVE, and other platforms for presenting stimuli to enhance data quality and encompass more of the research community.

ACKNOWLEDGMENTS

This research was supported by NIH under grant number P20 GM103650 and by NSF under grant number 1911041.

REFERENCES

Deepak Akkil, Poika Isokoski, Jari Kangas, Jussi Rantala, and Roope Raisamo. 2014. TraQuMe: A tool for measuring the gaze tracking quality. In *Eye Tracking Research and Applications Symposium (ETRA)*. https://doi.org/10.1145/2578153.2578192

James Barabas, Robert B. Goldstein, Henry Apfelbaum, Russell L. Woods, Robert G. Giorgi, and Eli Peli. 2004. Tracking the line of primary gaze in a walking simulator: Modeling and calibration. *Behavior Research Methods, Instruments, and Computers* (2004). https://doi.org/10.3758/BF03206556

Pieter Blignaut and Tanya Beelders. 2012. TrackStick: A data quality measuring tool for Tobii eye trackers. In *Eye Tracking Research and Applications Symposium (ETRA)*. https://doi.org/10.1145/2168556.2168619

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 *NEUROTECHNIX 2014 - Proceedings of the 2nd International Congress on Neurotechnology, Electronics and Informatics*. 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* (2018). https://doi.org/10.3389/fpsyg.2018.00803

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, and Computers* (2002). https://doi.org/10.3758/BF03195486

Benedikt V. Ehinger, Katharina Groß, Inga Ibs, and Peter König. 2019. A new comprehensive eye-tracking test battery concurrently evaluating the Pupil Labs glasses and the EyeLink 1000. *PeerJ* 2019, 7 (2019). https://doi.org/10.7717/peerj.7086

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 *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3025453.3025599

Agostino Gibaldi, Mauricio Vanegas, Peter J. Bex, and Guido Maiello. 2017. Evaluation of the Tobii EyeX Eye tracking controller and Matlab toolkit for research. *Behavior Research Methods* (2017). https://doi.org/10.3758/s13428-016-0762-9

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. In *Eye Tracking: A comprehensive guide to methods and measures*. OUP Oxford, Chapter 2, 9–64.

Kenneth Holmqvist, Marcus Nyström, and Fiona Mulvey. 2012. Eye tracker data quality: What it is and how to measure it. In *Eye Tracking Research and Applications Symposium (ETRA)*. https://doi.org/10.1145/2168556.2168563

Rob Howard. 2006. Provider Model Design Pattern and Specification. http://msdn.microsoft.com/en-us/library/ms972319.aspx. [Online; accessed 7-April-2020].

Michal Kowalik. 2011. Do-It-Yourself Eye Tracker: Impact of the Viewing Angle on the Eye Tracking Accuracy. *Proceedings of CESCG 2011: The 15th Central European Seminar on Computer Graphics* (2011). https://doi.org/S/N

Jeff MacInnes. 2018. Wearable Eye-tracking for Research : comparisons across devices. *bioRxiv* (2018).

Diederick C. Niehorster, Tim H.W. Cornelissen, Kenneth Holmqvist, Ignace T.C. Hooge, and Roy S. Hessels. 2018. What to expect from your remote eye-tracker when participants are unrestrained. *Behavior Research Methods* (2018). https://doi.org/10.3758/s13428-017-0863-0

Marcus Nyström, Richard Andersson, Kenneth Holmqvist, and Joost van de Weijer. 2013. The influence of calibration method and eye physiology on eyetracking data quality. *Behavior Research Methods* (2013). https://doi.org/10.3758/s13428-012-0247-4

Marcus Nyström and Kenneth Holmqvist. 2010. An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavior Research Methods* (2010). https://doi.org/10.3758/BRM.42.1.188

Thies Pfeiffer and Marc Erich Latoschik. 2008. Evaluation of Binocular Eye Trackers and Algorithms for 3D Gaze Interaction in Virtual Reality Environments. *Journal of Virtual Reality and Broadcasting* (2008).

Tobii Technology. 2011. Accuracy and precision test method for remote eye trackers. *Test* (2011).