GazeXR: A General Eye-Tracking System Enabling Invariable Gaze Data in Virtual Environment

Chris Lenart, Yuxin Yang, Zhiqing Gu, Cheng-Chang Lu, Karl Kosko, Richard Ferdig, and Qiang Guan(✉)

Kent State University, Kent, OH 44240, USA
{clenart4,yyang45,zgu1,cclu,kkosko,rferdig,qguan}@kent.edu

Abstract. Controlling and standardizing experiments is imperative for quantitative research methods. With the increase in the availability and quantity of low-cost eye-tracking devices, gaze data are considered as an important user input for quantitative analysis in many social science research areas, especially incorporating with virtual reality (VR) and augmented reality (AR) technologies. This poses new challenges in providing a default interface for gaze data in a common method. This paper propose GazeXR, which focuses on designing a general eye-tracking system interfacing two eye-tracking devices and creating a hardware independent virtual environment. We apply GazeXR to the in-class teaching experience analysis use case using external eye-tracking hardware to collect the gaze data for the gaze track analysis.

Keywords: Virtual reality · Eye-tracking · Human-Computer Interaction · Education technology

1 Introduction

Eye-tracking methodologies have existed since the late 1970s [20,29], significant progress, was established with the addition of the eye-mind hypothesis (EMH). The EMH attempts to establish a correlation with fixation and cognition [12]. While the premise of the EMH theory has not implicitly been proven, eye-tracking technology is based on its premise. One particular issue is covert attention, as attention was shown be independent of where a user is looking [18]. However, it has been shown that the movement of the attention will substantially move the eyes [4,9]. While eye-tracking data may not be bijective, it has been shown to give valuable information for domain-specific applications such as geometry [24].

Being a domain-specific application, virtual reality (VR) stands to benefit from eye-tracking. The first to utilize this technology was aircraft training [6]. Unlike typical, non-free movement, eye-tracking devices; virtual reality provides a 3D space for gazing. This virtual environment is consistent and a well-controlled state, unlike the world. Controlled virtual environments allows for

J. Y. C. Chen and G. Fragomeni (Eds.): HCII 2021, LNCS 12770, pp. 47–58, 2021.
https://doi.org/10.1007/978-3-030-77599-5_4

researchers to know what exists in at a given time. Knowing this information, a well crafted training experience can be utilized and annotated. While not unique to Virtual Reality, eye-tracking can show expertise through visual understanding. Like chess [21], this is particularly apparent for environments that are information-rich and dynamic [11]. With the fine grain control of environments, virtual reality and eye-tracking can provide keen insights for discerning expertise.

While eye-tacking has given virtual reality an additional layer of information, through the 3rd dimension of space, it does not come without its issues:

- **Head movement**. Traditional methods of eye-tracking, such as Pupil Center Corneal Reflection (PCCR), are error prone when head movements are introduced [31]. Fortunately, eye-tracking solutions such as Pupil Lab utilize a model-based solution that allows for free head movements [15]. This problem, in-directly, has lead to different adaptions in the gaze estimation process.
- **Lack of open standards**. Eye-tracking suffers from the lack of open standards in interfacing with these devices. With the increase of availability and quantity of eye-tracking devices [7], this issue will continue to grow. From calibrating the device, structuring of gaze data, and interfacing methodologies, eye-tracking devices differ from manufacturer to manufacturer.

The nature of our study is interdisciplinary, being that it aims to improve existing computer science methodologies; while yielding beneficiary to educational professional development.

- **Computer Scientist's Perspective**. This paper aims to improve the methodology of interfacing with multiple eye-tracking devices to unify the collection of gaze data. Creating an unique solution and method for standardization of gaze estimation pipeline.
- **Educator's Perspective**. This papers provides educators with a consistent solution for gaze datum interpretation. Which allows educators to correctly analyze and critique professional skills through behavioral and observable methods.

The paper is organized as follows. The introduction presents a logical stepping-stone for eye-tracking technology into virtual reality and establishes two clear problems in the field. The background provides the reader with research in the area and discusses key terminology that will be used throughout the paper. The system overview describes the architecture of the project and steps taken to solve any challenge. The case study section provides readers with results of our work and the performance overview. Finally, the conclusion section provides a detailed summary of solutions solved, with future works providing the next steps in the project.

2 Background

2.1 Gaze Mapping

Eye-tracking methodologies have existed since the late 1970s [20,29]. The movement of the eye is often broken down into two categories: saccadic and visual

fixations [19]. Raw eye movement is tracked through IR cameras, segmenting the pupil and other eye features. Once process of gaze estimation finishes, it returns a gaze-vector and a confidence based on the given eye image. Traditional methods of mapping gaze become more complex in virtual reality. As eye-tracking adds an additional dimensional to gaze data, known as depth. Using ray casts from the user's head to the gazed object, depth information can be computed [2]. However, with flat videos such as that used in a skybox, the depth is constant. As the depth value is computed when the two dimensional video is mapped into 3D space.

2.2 Head and Eye-Tracking

Ordinary methods of eye-tracking, like Pupil Center Corneal Reflection (PCCR), could result in errors with angled movement of one degree [31]. Some solutions, like that of Pupil Lab's Core uses a model-based approach [15], rendering an eye in 3D space fitting the pupil to an eclipse [28], to allow for free movement. This new free movement, allows for an additional layer of head tracking to be added. One method of measuring head movement is through the use of an inertial measurement unit (IMU). This method is what allows observer head control for virtual reality headsets.

2.3 Eye-Tracking Issues

Modern eye-tracking headsets can utilize different calibration methods for gaze-vector prediction. This can be problematic, when attempting to interface with multiple eye-trackering devices. While appearance-based deep convolutional neural networks (CNNs) can solve auto-calibrate [14,27,30]. Leading to removal of the overhead of platform specific calibration methods, it would require consumer products to have support from manufactures.

2.4 Visualizing Gaze Data

Gaze data provides spatial-temporal attention details for a given subjects. This means that data can be represented in terms of both space and/or time. There exist two scopes of data:

- **Local Data**. Restricted to a range, such a time restriction
- **Global Data**. Representative of the whole, such as, a collection of sessions or a whole video session.

Each sequence of data can utilize a probability density function (PDF) to understand Areas-of-Interests (AOI). Statistical likelihood can be visualized in terms of a heatmap; this can directly overlay video frames to determine the objects that peaked the user's attention and gaze [17]. Displaying an unwrapped video,

can provide context such as field of view (FOV) [16]. Another aspect of gaze data is fixation, which typically within the threshold of 100–200 ms [23], usually represented as a path.

3 System Overview

3.1 GazeXR Architecture

For the Pupil Lab's Core device, our approach utilizes the hmd-eyes Unity plugin[1] to communicate with the Pupil Capture service through the ZMQ protocol over the network [13]. This can be useful if the HMD device is not directly connected to the eye tacking device, resulting in an additional communication layer. For our setup, we used an Oculus Rift S[2], tethered to a host machine, that ran the Pupil Lab's Capture service. Utilizing the host machine as middleware, the host can ingress gaze data and time synchronize the request with that of the headset. The middleware server serves as the hub for content and storing data. It's broken into two part, web-server and data management. The web-server provides a route to list the existing videos and is the broker for video streaming. While the data management part accepts all event logs and session gaze data ad-hoc or post-session. This process allows for the host machine to take a detached approach in collecting gaze data. Implicitly, the data will be piped from the eye-tracker, to the host machine.

Handling this data, there are two coexistent approaches. The first being the direct storage of raw pupil source frame, gray-scale to keep a minimal data storage. Saving raw source data would allow for the user to process the eye later in a gaze estimation pipeline, resulting in better data as models improve. But this method would require more space to store the image and time writing to disk over the network. Since the gaze estimation methods of some eye-trackers are private, the output of gaze vector may be the only output. In this case, the only approach is to let gaze estimation pipeline handle the process and receive its values. Resulting in the pupil data being estimated into gaze data, make storage easy and removing the need for post-posting.

The same approach works for standalone devices, like the Pico Neo 2 Eye, which was able to handle both the eye-tracking and application running from within the device. Once a gaze event happens, the hook of the custom "Gaze Manager plugin" (see Fig. 1) would fire, resulting in a request to get the head-tracking position from the headsets IMU. Based on Pupil Lab's white paper, "the average gaze estimation, from tracking pipeline to network output, was 0.124 s" [13]. Being that these values don't need to be displayed in real-time, these values can be queued for later processing. Previous head positional values must be kept in memory, temporarily, to be correctly paired with proper gaze

[1] https://github.com/pupil-labs/hmd-eyes.
[2] https://www.oculus.com/rift-s/.

data. Being that the speeds of 348 ± 92 degrees per second are peak for healthy individuals [22], this is suitable base-line for sampling. Meaning that for a latency less than one second, no more than 440 head position values will need to be kept in memory. Once the head position values are found in memory, they can be grouped together with the queued gaze data. Returning both the gaze and head position together at a specific time of gaze. This provides a fair annotation data point for a given gaze event.

3.2 Interfacing Multiple Devices

When handling multiple Virtual Reality headsets, it is important to keep an agnostics approach to handle multi-platform support. This approach allows for a singular monolithic codebase to handle the functionality, allowing for consistent experience, despite the platform. A multiple component system can be utilized to break up the platform's code where functionality differs. This solution attempts to generalize both Virtual Reality functionalities and Eye-tracking solutions, in order to, interface and record the appropriate data. There exists a plethora of SDK solutions to manage interfacing with virtual reality devices such as OpenVR and Unity XR. These solutions provide a way to communicate with multiple native interfaces, for a single subsystem to handle. Allowing for Inputs devices such as controllers, IMU sensors, and displays to be controlled by a software interface.

Unlike interfacing with a virtual reality device, no general interface exists to handle eye-tracking by default. Eye-tracking devices can be broken down into two categories: built-in eye-tracking and external eye-tracking devices. Built-in eye-tracking can be implemented through features within a manufacturer's native SDK, such as that in the Pico Neo 2 Eye[3]. External eye-tracking hardware from the HMD device, like a Pupil Core[4] devices requires an extra communication level of abstraction to handle gaze data. To generically handle multiple devices, an additional level of abstraction is required to handle gaze data. Being limited to two eye-tracking devices, one built-in eye-tracking device and one external eye-tracking device. Our approach was generalized to attempt to handle the addition of future devices. Given the constraints, our solution provides a custom Unity plugin to create a gaze event hook. This method will take the multiple sources of gaze data and generalize it so that it becomes generic and consistent implementation. Allowing for universal functions to be created despite the data source.

[3] https://www.pico-interactive.com/us/neo2.html.
[4] https://pupil-labs.com/products/vr-ar/.

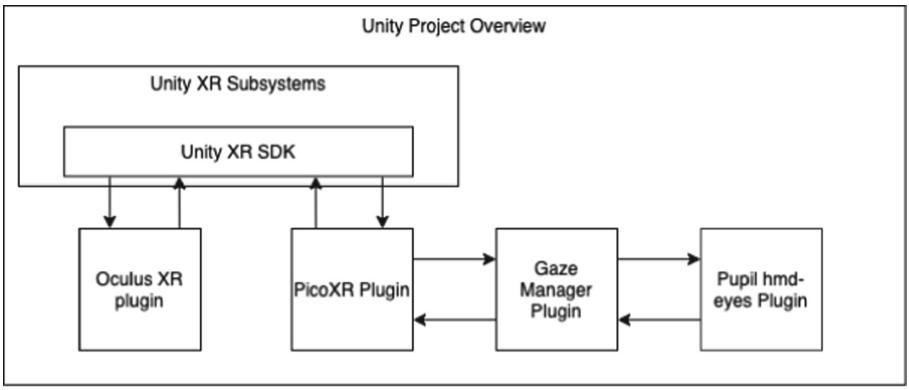

Fig. 1. The Gaze Manager Plugin is a default interfaces with PicoXR and Pupil Lab's hdm-eye plugin. Allowing for a direct communications for managing gaze datum.

3.3 Video's Role in Gaze Data

Being that our research targets gaze events that exist within a video, a more few parameters must be considered to be able to properly recreate the experience. One important property is the video frame number, the still image number at the time of gaze. Much like obtaining head position, the video frame must take into account latency from the gaze event. Unlike the head position, this value can be computed since it is time dependent. This is done by taking the current frame, $f_{current}$, and subtracting the frames from since that time. The amount of frames since latency is the product of the video's frames per second, fps, and the latency time, $Time_{current}$. Where f_x is targeted frame in the video.

$$F_x = F_{current} - (fps * Time_{current}) \tag{1}$$

Being that f_x is reliant on several variable, the equation is built to be devices agnostic. As video formats can change the frame-rate based on the device, this information can be pulled from the video's meta-data.

3.4 Data Management

Once, this frame has been calculated, the all the required properties are computed for the gaze event. Meaning that the gaze event data can be saved as a JSON object (see the Fig. 2 for format) and queued to be added to the database. Interfacing with a NoSQL database means no structures, that the JSON structure can directly be pushed as it comes in. Collections can be made for each of the 3 structures: videos, videoSessions, and gazeEvents.

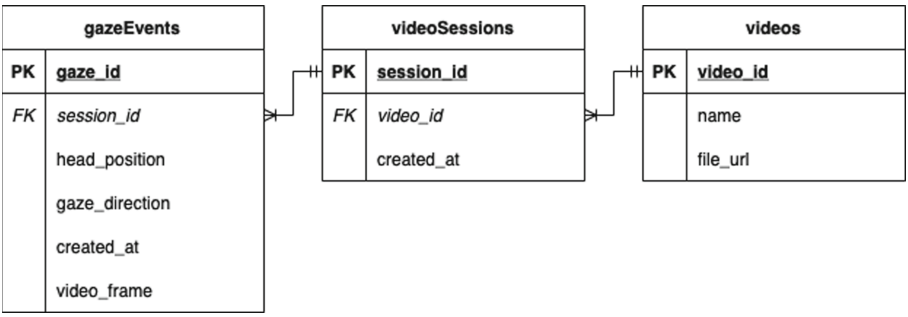

Fig. 2. The data relationship model for gaze events

3.5 Creating a Virtual Environment

Given a 360° equirectangular video, there are two classical approaches for video mapping: equirectangular projection (ERP) [25] and cubic mapping (CMP) [8]. Projecting is the process of taking a 2D video and projecting it onto a geometric object. Where an observer, the VR users, stands in the middle of said object. Giving the video the appearance of depth, creating a virtual environment. A panoramic video can be converted from its longitude and latitude layout and mapped to a UV texture. Then this texture that can be used as a skybox, resulting in a similar appearance of a CMP (see Fig. 3). This means that the projection gains all the benefits that come with CMP such as visual quality improves due to texel density. Since the resulting video is projected, the same calculations and shader functions can be used to transform the 2D gaze data into the needed its spot in 3D space.

Due to the large size of panoramic videos and ambisonic audio, it can be difficult for a standalone virtual reality headset to save to disk. To combat this issue, a hosting external machine, running a web-server and a database can aid. As seen in Fig. 2, a collection of videos and there name can be pulled from the web-server by a HTTP request. Then when a video is requested to be played on a headset, the video is sent by MPEG-DASH (DASH) to be streamed. Using DASH allows for content to be streamed over the network in segments [26]. Since the Pico Neo 2 is running a version of Android, and the support for DASH isn't full-supported yet [5], the web-server will serve the content as a progressive stream as a fallback. If paired with a tile based method and a hexaface sphere-mapping, it can save up to 72% bandwidth [10].

3.6 User Event Handling

Many things can go wrong physically when it comes to a VR headset. The cables could tangle, the controller's batteries could die, the headset could slip-off the users, or a software related issue could exist. Slipping, can cause an issue with the data collected during an experiment. It's even recommended to readjust the headset and re-calibrate the eye-trackers every five to ten minute [1]. It is

Fig. 3. This frame is the front facing cube texture, or positive X, of the skybox video projection. The video is streamed to the application through DASH.

more important to be able to stop recording data when things go wrong. For this reason, a event-driven architecture (EDA) was chosen, as it is important to send events asynchronous and tie them to state. Two current states exist for this architecture: recording and video player state. Given the possible failure, it is critical to pause before data become unusable. On the other spectrum, being that users are in a virtual space, being able to pause and to take a further look is important too. This allows for a more data to be collected on an area-of-interest (AOI). Including the additional data produced by the event logs, an additional layer of information such as time of pause and gaze points during pause. Since the frame of video is recorded on a gaze event, pausing will still record gaze data for the proper video context. Along with this, data that exists over-multiple frames; while the time of gaze from the system's clock increases, can be used to sequentially build paused video segments of a session.

3.7 Results

With gaze visualization methods, gaze can be projected into virtual classrooms. Using gaze data, fixations can be determined and plotted to a heatmap which displays clear AOIs in the classroom (as seen in Fig. 4A). Through the visualization of gaze vectors and binding perspective to be the top view, or positive Z-axis, rotational data be show where users stopped turned. In the case of Fig. 4D, the user spent most of the time in the front and didn't even turn around.

4 Discussion

With the high resolution video being sent through the network, it can lead to performance bottleneck for networks that exceed its bandwidth. We profile the network traffic shown in Fig. 5, while downloading the 360 video into VR headset.

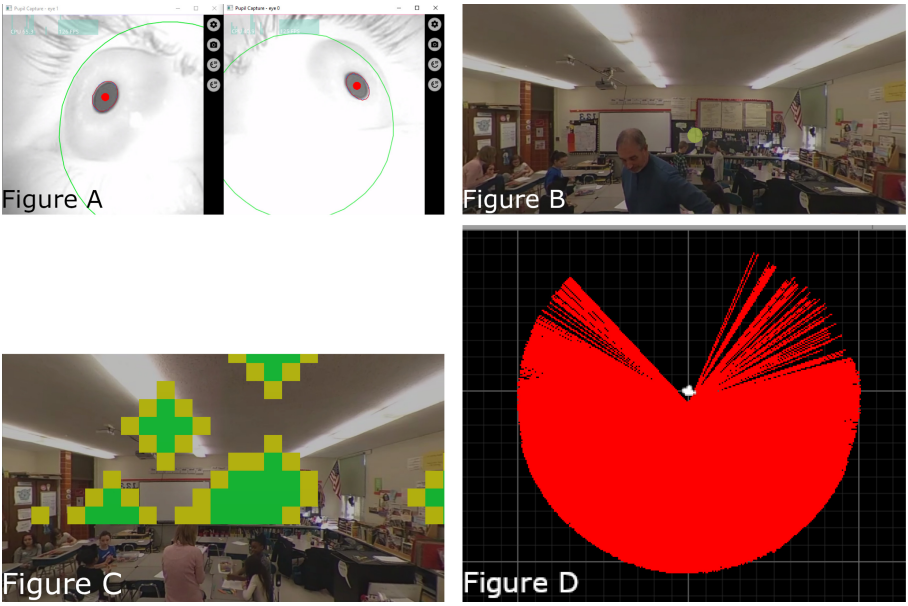

Fig. 4. The figure A provides is Pupil Lab's Capture program, which is tracking the pupil and passing it through the gaze estimation. The remaining figures [B-D], shows gaze in a virtual environment: through fixation markers, heatmaps, or gaze vectors.

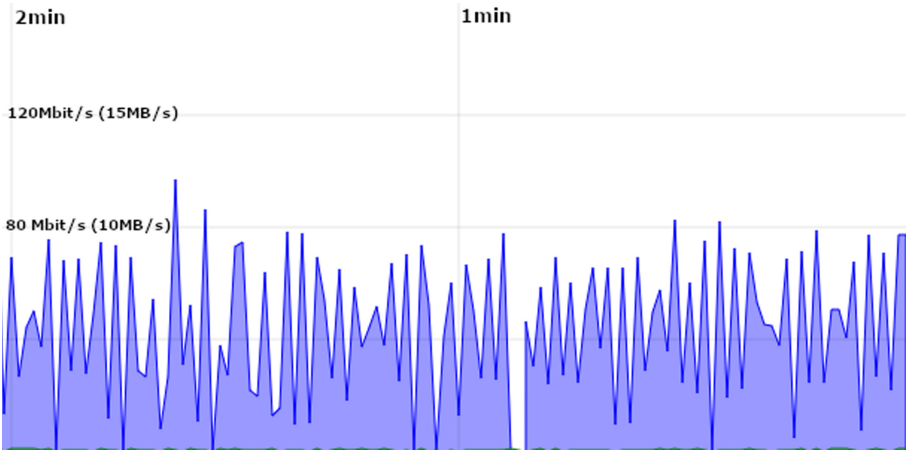

Fig. 5. The bandwidth of a 2.94 GB equirectangular video being streamed over Pico Neo 2 for 3 min, through http progressive streaming. The blue represents the inbound traffic, peaking at 97.39 Mbit/s or 12.17 MB/s. (Color figure online)

With standalone headsets, like Pico Neo 2, sending data through the WiFi is the only networking method. Networking standards like WiFi 4, or IEEE 802.11n-2009, might struggled if interfaced with multiple devices due to 600Mbit/s theoretical limit. However, more durable solutions like WiFi 5, IEEE 802.11ac-2013, or WiFi 6, IEEE 802.11ax, can handle 866 Mbit/s and 1201 Mbit/s respectively [3]. Local-storage can solve this solution, but isn't applicable on all standalone devices. As these videos can take up significant file-storage space. With moderation, sending videos over the network can be successful.

5 Conclusion and Future Work

In this paper, we present GazeXR, which is designed for projecting eye-tracking data into virtual reality headsets. Utilizing the Gaze Manager plugin, both Pupil Lab's Core Device and Pico Neo 2 Eye are able to interface with the virtual realty based in-class teaching experience analysis application on their respective platforms. GazeXR provides a customs platform for handling gaze events that can be managed and analyzed by researchers.

In future work, we plan to collect research data through systematic trials. to provide a way of computing insight for social science researchers utilizing gaze data in any virtual environment applications. GazeXR can help to compute a score of expertise in the field, through gazed objects recognizing from video frames. Using machine learning (ML), objects can be detected and classified. An expert in the field then can be introduced to input a baseline knowledge to determine important actions. To determine if a user fixates on an object, a bounding-boxes or image segmentation can be used to see if the user‘s gaze collides, which may result in a change in score. Along with this, a mobile application could be built to interface with this system.

Acknowledgement. This project is funded by National Science Foundation, Grant #1908159. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Clay, V., König, P., König, S.U.: Eye tracking in virtual reality. Journal of Eye Movement Research **12**(1), (2019)
2. Clay, V., König, P., König, S.U.: Eye tracking in virtual reality. Journal of Eye Movement Research 12(1) (Apr 2019). https://doi.org/10.16910/jemr.12.1.3, https://bop.unibe.ch/JEMR/article/view/4332-Clay-final-sub
3. Corporation, T.L.: Archer AX10 — AX1500 Wi-Fi 6 Router — TP-Lin (2021), https://www.tp-link.com/us/home-networking/wifi-router/archer-ax10/#specifications, accessed on 02.09.2021
4. Deubel, H., Schneider, W.X.: Saccade target selection and object recognition: Evidence for a common attentional mechanism. Vision Research **36**(12), 1827–1837 (Jun 1996). https://doi.org/10.1016/0042-6989(95)00294-4, https://doi.org/10.1016/0042-6989(95)00294-4

5. Developers, G.: Supported media formats | Android Developers (2021), https://developer.android.com/guide/topics/media/media-formats.html#recommendations, accessed on 02.09.2021
6. Duchowski, A., Shivashankaraiah, V., Rawls, T., Gramopadhye, A., Melloy, B., Kanki, B.: Binocular eye tracking in virtual reality for inspection training. In: ETRA (2000)
7. Ferhat, O., Vilariño, F.: Low cost eye tracking: The current panorama. Computational Intelligence and Neuroscience **2016**, 1–14 (2016). https://doi.org/10.1155/2016/8680541, https://doi.org/10.1155/2016/8680541
8. Greene, N.: Environment mapping and other applications of world projections. IEEE Computer Graphics and Applications **6**(11), 21–29 (Nov 1986). https://doi.org/10.1109/mcg.1986.276658,https://doi.org/10.1109/mcg.1986.276658
9. Hoffman, J.E., Subramaniam, B.: The role of visual attention in saccadic eye movements. Perception & Psychophysics **57**(6), 787–795 (Jan 1995). https://doi.org/10.3758/bf03206794,https://doi.org/10.3758/bf03206794
10. Hosseini, M., Swaminathan, V.: Adaptive 360 VR video streaming: Divide and conquer. In: 2016 IEEE International Symposium on Multimedia (ISM). IEEE (Dec 2016). https://doi.org/10.1109/ism.2016.0028, https://doi.org/10.1109/ism.2016.0028
11. Jarodzka, H., Holmqvist, K., Gruber, H.: Eye tracking in educational science: Theoretical frameworks and research agendas 10 (01 2017). https://doi.org/10.16910/jemr.10.1.3
12. Just, M., Carpenter, P.: A theory of reading: from eye fixations to comprehension. Psychological review **87**(4), 329–54 (1980)
13. Kassner, M., Patera, W., Bulling, A.: Pupil. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication. ACM (Sep 2014). https://doi.org/10.1145/2638728.2641695, https://doi.org/10.1145/2638728.2641695
14. Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., Torralba, A.: Eye tracking for everyone. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (Jun 2016). https://doi.org/10.1109/cvpr.2016.239, https://doi.org/10.1109/cvpr.2016.239
15. Lab, P.: Pupil Lab's Pupil Capture (2021), https://docs.pupil-labs.com/core/software/pupil-capture/#pupil-detection, accessed on 02.09.2021
16. Löwe, T., Stengel, M., Förster, E.C., Grogorick, S., Magnor, M.: Gaze visualization for immersive video. In: Burch, M., Chuang, L., Fisher, B., Schmidt, A., Weiskopf, D. (eds.) Eye Tracking and Visualization, pp. 57–71. Springer International Publishing, Cham (2017)
17. Masse, B., Lathuiliere, S., Mesejo, P., Horaud, R.: Extended gaze following: Detecting objects in videos beyond the camera field of view. In: 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019). IEEE (May 2019). https://doi.org/10.1109/fg.2019.8756555, https://doi.org/10.1109/fg.2019.8756555
18. Posner, M.I.: Orienting of attention. Quarterly Journal of Experimental Psychology **32**(1), 3–25 (1980). https://doi.org/10.1080/00335558008248231, https://doi.org/10.1080/00335558008248231
19. Purves, D., Augustine, G., Fitzpatrick, D., Katz, L., LaMantia, A., McNamara, J., Williams, S.: Neuroscience 2nd edition. sunderland (ma) sinauer associates. Types of Eye Movements and Their Functions (2001)

20. Rayner, K.: Eye movements in reading and information processing: 20 years of research. Psychological Bulletin **124**(3), 372–422 (1998). https://doi.org/10.1037/0033-2909.124.3.372, https://doi.org/10.1037/0033-2909.124.3.372
21. Reingold, E., Sheridan, H.: Eye movements and visual expertise in chess and medicine, vol. 528–550, pp. 528–550 (08 2011). https://doi.org/10.1093/oxfordhb/9780199539789.013.0029
22. Röijezon, U., Djupsjöbacka, M., Björklund, M., Häger-Ross, C., Grip, H., Liebermann, D.G.: Kinematics of fast cervical rotations in persons with chronic neck pain: a cross-sectional and reliability study. BMC Musculoskeletal Disorders 11(1) (Sep 2010). https://doi.org/10.1186/1471-2474-11-222, https://doi.org/10.1186/1471-2474-11-222
23. Salvucci, D.D., Goldberg, J.H.: Identifying fixations and saccades in eye-tracking protocols. In: Proceedings of the symposium on Eye tracking research & applications - ETRA '00. ACM Press (2000). https://doi.org/10.1145/355017.355028, https://doi.org/10.1145/355017.355028
24. Schindler, Maike, Lilienthal, Achim J.: Domain-specific interpretation of eye tracking data: towards a refined use of the eye-mind hypothesis for the field of geometry. Educational Studies in Mathematics **101**(1), 123–139 (2019). https://doi.org/10.1007/s10649-019-9878-z
25. Snyder, J.P.: Flattening the earth: two thousand years of map projections. University of Chicago Press (1997)
26. Sodagar, I.: The MPEG-DASH standard for multimedia streaming over the internet. IEEE Multimedia **18**(4), 62–67 (Apr 2011). https://doi.org/10.1109/mmul.2011.71, https://doi.org/10.1109/mmul.2011.71
27. Sugano, Y., Matsushita, Y., Sato, Y.: Learning-by-synthesis for appearance-based 3d gaze estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1821–1828 (2014)
28. Swirski, L., Dodgson, N.: A fully-automatic, temporal approach to single camera, glint-free 3d eye model fitting. Proc. PETMEI pp. 1–11 (2013)
29. Ten Kate, J., Frietman, E.E., Willems, W., Romeny, B.T.H., Tenkink, E.: Eye-switch controlled communication aids. In: Proceedings of the 12th International Conference on Medical & Biological Engineering. pp. 19–20 (1979)
30. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: Appearance-based gaze estimation in the wild. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (Jun 2015). https://doi.org/10.1109/cvpr.2015.7299081, https://doi.org/10.1109/cvpr.2015.7299081
31. Zhu, Z., Ji, Q.: Novel eye gaze tracking techniques under natural head movement. IEEE Transactions on biomedical engineering **54**(12), 2246–2260 (2007)