Eye-To-Eye: Towards Visualizing Eye Gaze Data

Youssef Othman
Media Engineering and Technology
Computer Science
German University in Cairo
Cairo, Egypt
youssef.gamaledin@student.guc.edu.eg

Mahmoud Khalaf
Media Engineering and Technology
Computer Science
German University in Cairo
Cairo, Egypt
mahmoud.abdelbadea@student.guc.edu.eg

Ahmed Ragab
Media Engineering and Technology
Computer Science
German University in Cairo
Cairo, Egypt
ahmed.ragabmahmoud@student.guc.edu.eg

Ahmed Salaheldin
Media Engineering and Technology
Computer Science
German University in Cairo
Cairo, Egypt
ahmad.salah-eldin@student.guc.edu.eg

Reham Ayman
Media Engineering and Technology
Computer Science
German University in Cairo
Cairo, Egypt
reham.ayman@guc.edu.eg

Nada Sharaf
Media Engineering and Technology
Computer Science
German University in Cairo
Cairo, Egypt
nada.hamed@guc.edu.eg

***Abstract*—The work in this paper introduces a platform that is able to visualize eye gaze data. The platform introduces two different directions or algorithms. The first is for videos with heat-map to convert the heat map data into raw numbers. The algorithm takes the video containing the heat map and cuts it into frames. In each frame, it locates the position of the heat-map by searching for the specific colour values and then displaying the position of heat-map in each frame. The algorithm also calculates the fixation points and how long each fixation was for the entirety of the video. The second algorithm is to do the opposite of the first and that is by converting the raw data of eye-movement to a heat map. We are also providing visualization platform for eye-tracking data.**

I. Introduction

Data visualization is defined as the communication of information using graphical representations like charts and maps, it is used as an alternative to textual or verbal communication. It actually dates back to the early days of computer graphics in the 1950s/. This is when the first graphs and figures were generated by computers [1], [2]. In this paper, we will be looking specifically at heatmap visualization that has been used commonly used for eye-tracking purposes as it gives a detailed overview of the user behavior. A heatmap interprets large amount of data easily to people who are not familiar with data analysis. 1 shows one frame of a video recorded using Eye-tracking software. Heatmap can be presented in the blue and green coloured circles where the user looked for only a very short duration. The yellow and red are where the user spends most of time looking. When a person looks at the whole video of the heatmap, they can estimate what was the main focus of the user. However, this falls short if the video was a long one since the person will not accurately remember the information from the video. This is where converting the heatmap frames into numerical data will be useful to accurately analyze results without having to estimate actual numbers by eyes. Therefore, the goal of the work is to be able to extract the data from videos containing heatmap of eye-

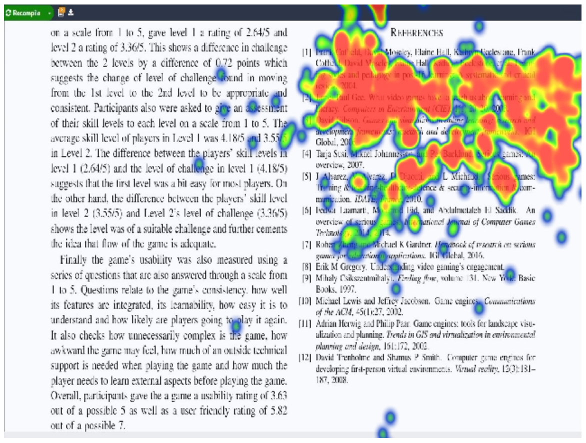


Fig. 1. Example of how a heat-map frame looks like

tracking data to help analyze and maximize their usefulness. In addition, the tools also converts raw data from eye movements to a heatmap. The best of our knowledge, no previous system was able to convert heat map (videos) to numerical data.

II. Background

A. Eye-Tracking

Eye-tracking is the process of recording eye movements and events using an eye tracking device. Collected data could then be analyzed using visualization and analytic techniques to test different hypotheses. There are many eye-tracking devices with different capabilities which can record where a user is looking at in a computer screen, projector screen, and mobiles, ... etc. It is a powerful approach to analyze the behaviour of the user [3]. Since the process of looking at something is automatic, making use of the eye-tracking tools would generate both qualitative and quantitative informative data. Such data could be used by researchers in a wide range of research including as Education, psychological studies and web design. Eye-tracking is a relatively simple measure. However, the tricky part is knowing what type of data it generates and how to use this data. There are numerous ways for approaching the eye-tracking technology, however the most two commonly used ways are either using eye-tracking tools such as Tobii[1] or installing eye-tracking software. Eye-Tracking tools provide a vast range of precise and accurate data metrics such as the gaze points and fixation time. This shows the user where their eye gaze landed on the screen along with the areas of interest on the test subject. The second most commonly used approach is by installing an eye-tracking software that simulates how eye-tracking tools work using a webcam which is more economic and user friendly. However, such software tools are limited. They generate a few data metrics that could mostly used for visualizing the eye-tracking experience like the eye gaze heat map. Nevertheless, the data turns out to be not very useful when it comes to analyzing the data for future use [4].

B. *Gaze Recorder*

Eye tracking tools are popular for generating plenty of useful and accurate data metrics. However, they have expensive operational costs. Thus, the aim of the work is to focus on the more affordable solution which is the eye-tracking software tools [5]. There are different available tools that could be used for eye tracking with small differences between them in the data metrics they generate and the ease of their use. The work in the paper focuses on GazeRecorder [2]. GazeRecorder records a video of the eye-gaze of the user through a webcam. The eye-gaze of the user is shown as a heatmap data in the video frames which visually indicates the behavior of the user's eye while doing a specific activity on their laptop or PC.

C. *HeatMaps*

Heatmaps represent a visualization methodology where values are shown as colors. It is a popular two-dimensional graphical representation [6]. Currently, heatmaps could be generated while using conventional webcams and eye-tracking software tools.

Fixation represents the times where the user is fixing their eyes on a specific object. In other words, the eyeball is almost static [7]. Heatmaps shows the viewing behavior of users. However, without analysis techniques, heatmaps could be ambiguous [6].

According to [8], the visualization and analysis of eye movement data is an emerging research field of research. Visually displayed data helps to find patterns, identify issues, and represent a huge amount of data.

III. Proposed Data Extraction and Visualization Systems

A. *HeatMap Data Extraction*

The first tsk was to acquire heat map data using an affordable approach. Acquiring the data from the eye-tracking tool proved to be challenging at first. Since there were not any programs that provided numerical data that can be later used or analysed. Most programs only display live data during the experiment. However, once it was over, the program does not provide the data. A few programs, including GazeRecorder, allows users to save the data in the form of a video where the eye coordinates can be seen in the form of a heatmap.

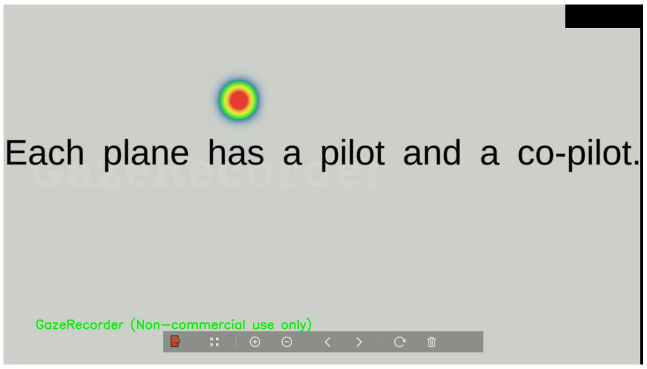

Each plane has a pilot and a co-pilot.
GazeRecorder
GazeRecorder (Non-commercial use only)

Fig. 2. Example of a frame to be operated on

The recorded videos of the eye-tracking data were abayzed using OpenCV, an Open Source Computer Vision and Machine Learning software Library [3].

The eye gaze location and fixation duration were the most important measurements to extract. To acquire the eye-gaze locations, the x and y coordinates of the eye gaze that are found in every frame that is represented by the heatmap gaze are needed. An example of a frame is shown in Figure 2.

As seen in Figure 2, the gaze is represented by a blue outline and three colours inside: green, yellow and red in order of intensity. The center of this circle represents the coordinates that are needed. To obtain the coordinates, the pixels of the frame were manipulated according to colour. First, a colour mask was used to change every pixel's colour to either black or white depending on the whether the pixel was within the range of colours that are set in the mask. The range of colours were set from [0, 50, 20] to [15, 255, 255] HSV. The [0-15] hue range represents the red colour as well as a slight hint of orange. By having these specifics ranges, gazes of all intensities can be identified. The result of this mask is shown in Figure 3 where all pixels having the shade of red found the range become white pixels and the rest turn to black.

The next step was to operate on the white pixels and find their centroids. However, it is important first to eliminate any white areas that do not represent the eye gaze like the one found in the bottom left of Figure 3. To do this, a simple area range was used as a checker. The range was from 1000 to 6000. The second step is locating the centre itself. This was done using contours and moments. OpenCV can identify the existence of any contour in an image using a contour finder function. This function is used on the white spots in the frame. Afterwards, the contourArea function was used to

[1]https://www.tobii.com/
[2]https://gazerecorder.com/
[3]https://opencv.org/

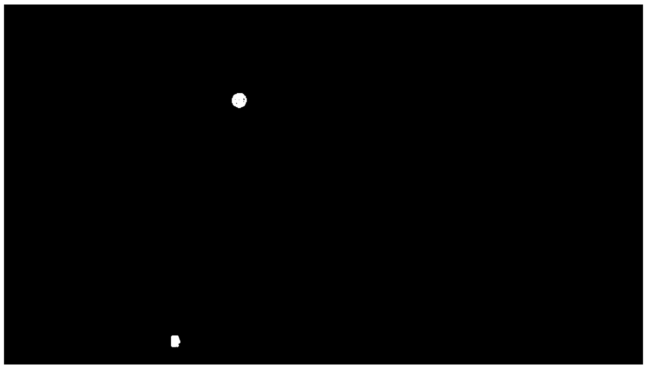


Fig. 3. Example of a frame after applying the masking operation

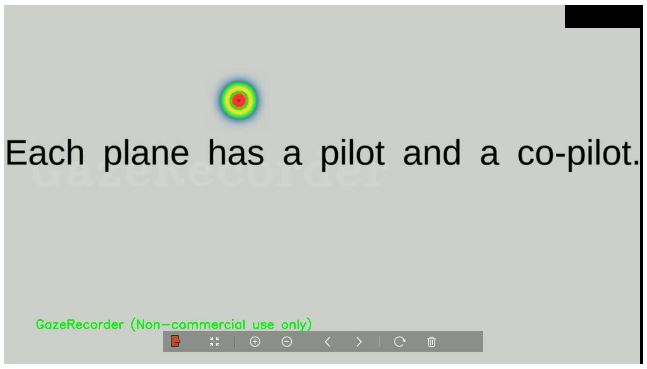

Each plane has a pilot and a co-pilot.
GazeRecorder (Non–commercial use only)

Fig. 4. Example of a frame after the centers have been identified, highlighted and the image inverted back to normal

calculate the area of all shapes that are enclosed by contours (the white spots). After checking if it is within the area range, the HuMoments function was used on the contours. The HuMoments function is used to generate seven Hu invariants. The invariants are generated according to an image moment. An image moment could be calculated using the intensities of the pixels in the image. A moment could be the weighted average of such intensities [4].

Finally, after identifying and retrieving the coordinates, a very small circle (dot) is drawn on the coordinate and contours around the shape. They are drawn to indicate that the coordinates have been correctly identified if they were to be checked manually, as shown in Figure 4.

As for the duration of each gaze, the video is used by an algorithm that returns the x and y coordinate found in every frame. This data is placed in a CSV file. Another algorithm is made to identify the gaze duration frame-by-frame. By checking the frame's eye gaze coordinates with the following frame, it could be determined whether the user had been looking in the same spot (with 20 pixels difference). The frames are checked one by one until the user is no longer fixating on this spot. Afterwards, an average for all the previous coordinates in this specific gaze is calculated.

[4]https://docs.opencv.org/2.4

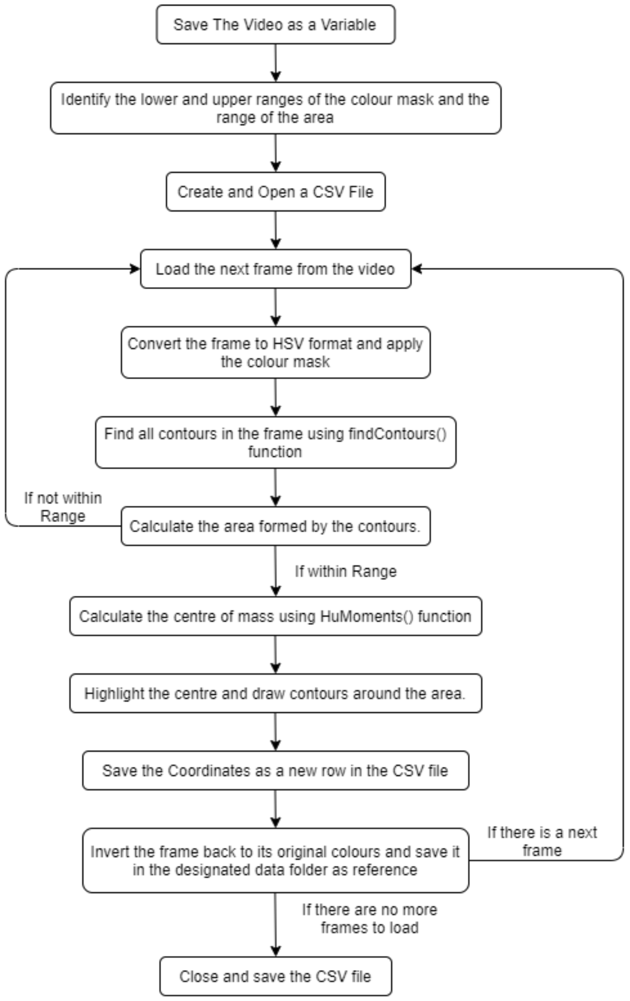

Save The Video as a Variable
Identify the lower and upper ranges of the colour mask and the range of the area
Create and Open a CSV File
Load the next frame from the video
Convert the frame to HSV format and apply the colour mask
Find all contours in the frame using findContours() function
If not within Range
Calculate the area formed by the contours.
If within Range
Calculate the centre of mass using HuMoments() function
Highlight the centre and draw contours around the area.
Save the Coordinates as a new row in the CSV file
Invert the frame back to its original colours and save it in the designated data folder as reference
If there is a next frame
If there are no more frames to load
Close and save the CSV file

Fig. 5. Flow chart diagram of the coordinates calculation algorithm

The final result is for example, "[541.25, 239.75] from frames 15 to 25 for a duration of 0.167 seconds". The seconds are calculated by using the video's FPS rate which is 60 in this case. Figures 5 and 6 show flow charts for the algorithms.

For the data to be retrieved, several tweaks and adjustments have been made. The heat gaze map's settings in GazeRecorder can be altered. The settings to be changed relate to the Time Window, how much time should pass before the visual heat map fades, Gaze plot size, how big is the heat map gaze in relation to the screen and the option to extend the Time Window during a static scene and by how much. The settings used for the experiment were 60 ms for Time Window and Gaze plot size set 4% and the option to extend the time window during a static scene set to off. The gaze plot size was set to 4% to have a clear gaze plot for every frame and a short time window of just 60ms was used to prevent having multiple contours in a single frame.

To test the accuracy of the coordinates calculating algorithm, an Eye Visualization Platform (discussed in Section

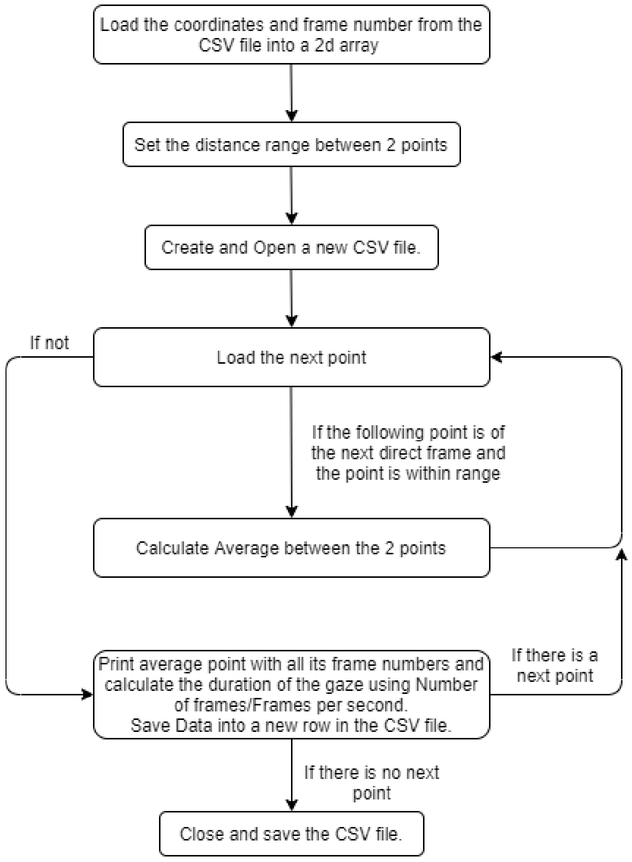

Load the coordinates and frame number from the CSV file into a 2d array
Set the distance range between 2 points
Create and Open a new CSV file.
If not
Load the next point
If the following point is of the next direct frame and the point is within range
Calculate Average between the 2 points
Print average point with all its frame numbers and calculate the duration of the gaze using Number of frames/Frames per second. Save Data into a new row in the CSV file.
If there is a next point
If there is no next point
Close and save the CSV file.

Fig. 6. Flow chart diagram of the gaze duration calculation algorithm

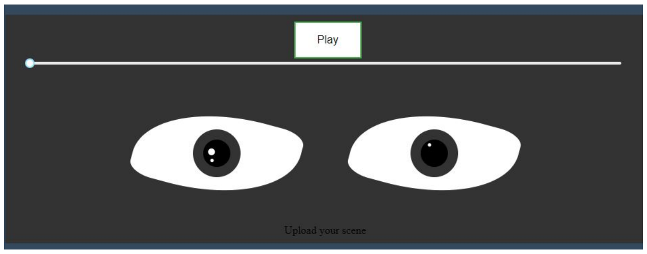


Fig. 7. Eye movements simulation

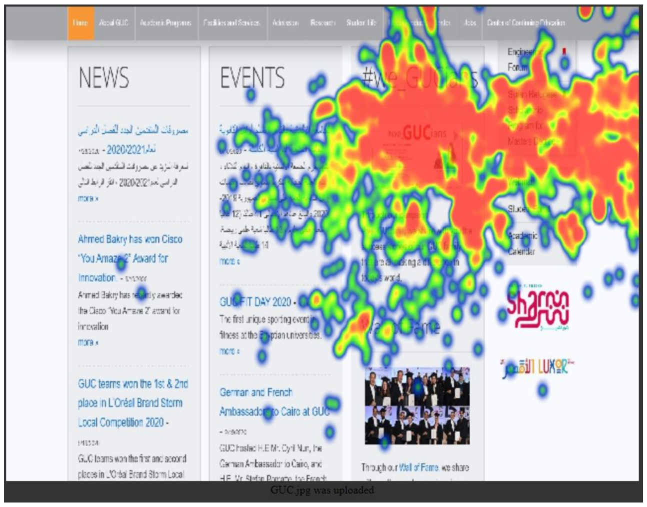


Fig. 8. Example of a heat-map

III-B) was used to produce multiple images with an eye gaze spot with random coordinates. The algorithm was ten used to collect the coordinates. They were compared them with the original ones. The distance between each point and its calculated counterpart was calculated and a mean absolute error of just 2.071 pixels was achieved. This value is relatively small as a shift of approximately 2 pixels across the average computer screen resolution will not have a significant effect on the area of the heatmap.

B. *Eye Visualization Platform*

An Eye Visualization platform was implemented for eye-tracking data analysis. This platform provides effective visualization techniques to better understand and present data in an efficient format. Dash [5] was used for implementing the visualizations.

The eye visualization platform enables users to visualize eye-tracking data through different visualization techniques. It has several components to be able to handle inputs, upload data, and visualize data.

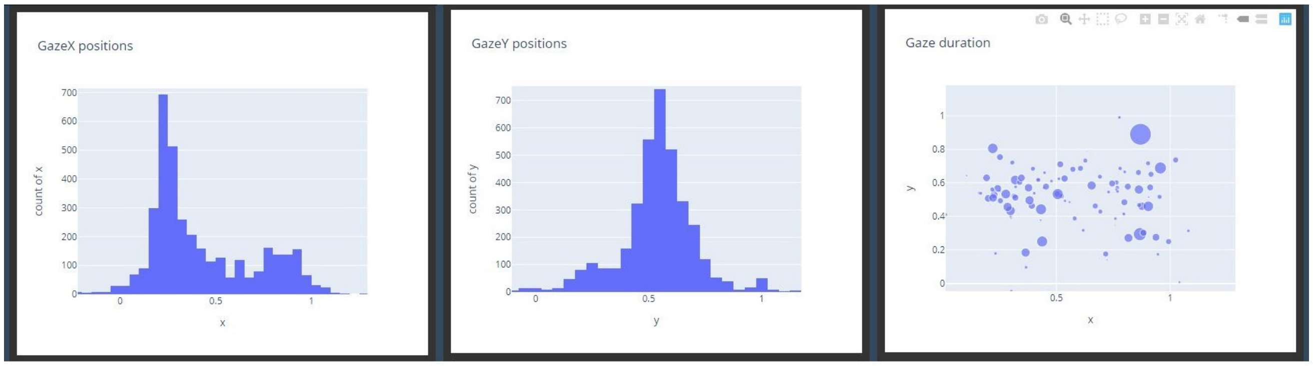
The first component is the eye movements' component. It allows the user to experience all the eye-tracking data points as a recorded video for eye movements, as shown in Figure 7. The user can choose the way of processing the data or the value that is represented by the slider as point by point or using the time to explain the position of the eye in a specific time of the experiment. For example, if the slider represents points, then the length of it will be the length of the data file and if it represents time, then the length of it will be the sum of the duration points. The users can start visualizing gaze movements by clicking play button which is converted to a pause button once visualization starts. In addition, users can also control the slider.

The other component allows the user to visualize the data using statistical diagrams such as bar charts, line charts, or scatter plot. Users can select from a dropdown list which methodology they want to use. In addition, more than one graph can be displayed at the same time for further exploration of data, as shown in Figure 9.

In addition to the previous diagram types, a heat map could be also generated. The data was clustered using k-means algorithm directly after uploading it to get the means of fixation points and then the areas of interest. An example of the heatmap technique used in this platform is shown in Figure 8.

Heatmap shows the areas of attention. However,a focusmap allows the users to view the information available in the areas of attention as shown in Figure 10. The users can also easily

[5]https://dash.plotly.com



(a) Fixation x count (b) Fixation y count (c) Duration of point x-y

Fig. 9. Some of the statistical diagrams

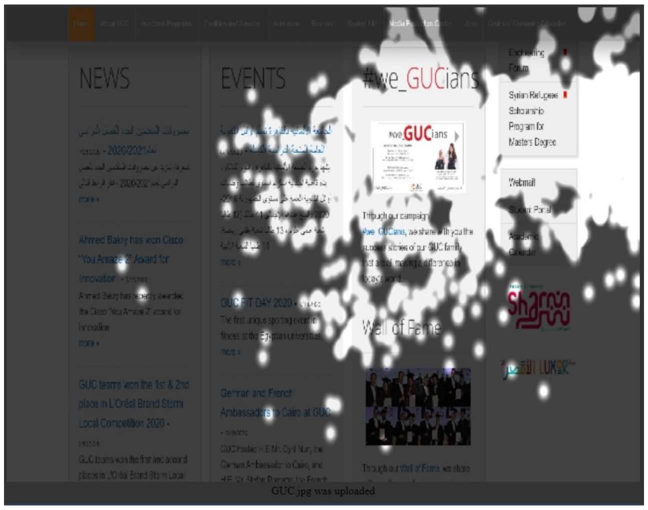


Fig. 10. Example of a focus-map

change between heatmap and focusmap through dropdown list.

IV. Conclusions and Future Work

The paper introduced different aspects of eye-gaze data visualization. First, an algorithm was introduced to extract numerical eye-gaze data from heatmap videos using Gazerecorder and OpenCV. In addition, using Dash Python framework, a visualization platform for eye-tracking data was built. It provides different techniques to enable users to easily visualize and analyze the eye-tracking data.

In the future, the visualization platform should enable the user to compare the results of different eye-tracking data sets.

References

[1] Frits Post, Gregory Nielson, and Georges-Pierre Bonneau. Data visualization: The state of the art. 01 2003.

[2] Matthew Ward, Georges Grinstein, and Daniel Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications*. A. K. Peters, Ltd., USA, 2010.

[3] Carlos H Morimoto and Marcio RM Mimica. Eye gaze tracking techniques for interactive applications. *Computer vision and image understanding*, 98(1):4–24, 2005.

[4] Daan R van Renswoude, Maartje EJ Raijmakers, Arnout Koornneef, Scott P Johnson, Sabine Hunnius, and Ingmar Visser. Gazepath: An eye-tracking analysis tool that accounts for individual differences and data quality. *Behavior research methods*, 50(2):834–852, 2018.

[5] Bryn Farnsworth. 10 most used eye tracking metrics and terms. 08 2018.

[6] Agnieszka (Aga) Bojko. Informative or misleading? heatmaps deconstructed. In Julie A. Jacko, editor, *Human-Computer Interaction. New Trends*, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[7] Antonio Diaz Tula, Andrew Kurauchi, Flávio Coutinho, and Carlos Morimoto. Heatmap explorer: An interactive gaze data visualization tool for the evaluation of computer interfaces. In *Proceedings of the 15th Brazilian Symposium on Human Factors in Computing Systems*, pages 1–9, 2016.

[8] Kuno Kurzhals, Michael Burch, Tanja Blascheck, Gennady L. Andrienko, Natalia V. Andrienko, and Daniel Weiskopf. A task-based view on the visual analysis of eye-tracking data. In Michael Burch, Lewis L. Chuang, Brian D. Fisher, Albrecht Schmidt, and Daniel Weiskopf, editors, *Eye Tracking and Visualization, ETVIS 2015, Chicago, Illinois, USA, October 25, 2015*, Mathematics and Visualization, pages 3–22. Springer, 2015.