DEVELOPMENT OF MATLAB TOOLBOX FOR EYE TRACKING SYSTEMS

HAMILTON RIVERA*, ALEXANDRE L. C. BISSOLI*, CHRISTIANE GOULART**, ELIETE CALDEIRA*, TEODIANO F. BASTOS-FILHO* , **

* Department of Electrical Engineering
**Postgraduate Program in Biotechnology
Federal University of Espirito Santo (UFES), Vitoria, Brazil
E-mails: hamriver@gmail.com, alexandre-bissoli@hotmail.com, christiane_goulart@yahoo.com.br, eliete.caldeira@ufes.br, teodiano.bastos@ufes.br

Abstract— Eye tracking is a technology that consists of calculating the eye gaze point of a user as he/she looks around. A device equipped with an eye tracker (ET) enables users to use their eye gaze as an input modality that can be combined with other input devices like mouse, keyboard, touch and gesture, referred as active applications in games, operative system navigation, e-books, market research studies, and usability testing. These eye tracker applications can be used for new assistive technologies. This work presents the development of a toolbox for Matlab with four modules which allow the use of eye-tracking systems for therapy (physical, psychological, medical), control of intelligent environment and studies about visual social attention. The results show that the interface developed allows the volunteers connecting from Matlab applications to the eye tracker device, control the sampling time, setting-up and configuring the system, managing the eye tracker data, generating analysis and graphic reports, and controlling the graphic interface.

Keywords— Eye tracking, eye gaze point detection, visual social attention, assistive technology.

1 Introduction

The most publicized strategy of applying eye tracking to existing interfaces is to use the eye to perform pointing and selection tasks. Glenstrup and Engell-Nielsen (1995) have argued on numerous evidence that there is relationship between the interests of individuals and what they are looking. Another source of evidence is the model built by Card Moran and Newell (1980), which provides the time T spent for the execution of pointing tasks with the use of traditional devices such as the mouse. The equation (1) shows a simplified version of the model:

$$T = TM + TV + TP + TR \quad (1)$$

In it, the motor subtask (i.e., effectively moving the device into the correct position), takes time TP, preceded by a cognitive task that takes TM time, and a visual task (i.e., visual target search in question) that takes time TV. TR is the system response time. Today, despite evidence that this model is not accurate (Hornof and Kieras, 1999), it still provides a reasonable estimate and justify the efforts to apply tracing to look at the execution of pointing tasks.

However, the direct mapping look (more specifically of fixations) to a system selection command creates the problem identified by Jacob (1990), called "Midas Touch", in which a selection can be activated at any position of the observed screen by the user, whether he intended to do it or not. This after filtering the eye tracker data, a challenge in designing interaction technique is avoid the Midas Touch problem, implementing mechanisms for the user indicating when one wants to perform a selection command. The first approach to solve this problem has been the implementation of a lag time, in which - the selection of an object is performed only after a time interval.

Several applications using eye tracking can be found in literature. Kocejko, Bujnowski, and Wtorek (2008) presented an Eye Mouse, which is a system to people with motor disabilities where the mouse position is controlled by eye gaze. Lupu et al. (2011) presented a system called Asistsys, which is based on eye tracking, making it possible to the people with motor disability to express their wishes and needs only by visualizing options on a monitor.

Studies about the impact of a system based on eye tracking in the quality of life of people with amyotrophic lateral sclerosis, a neurodegenerative disease, are presented in (Calvo et al., 2008). In fact, the eye tracking technique has quite potential of application in interfaces to people with this disease, because they maintain their cognitive ability and, in the most cases, the ability to control the eye gaze, According to Calvo (2008), people who took part in its studies and tested the system noted an improvement in the quality of life, because they were able to communicate independently, and the communication was easier, briefer and less painful.

Figure 1 shows a block diagram of the eye tracking system considered in this work.

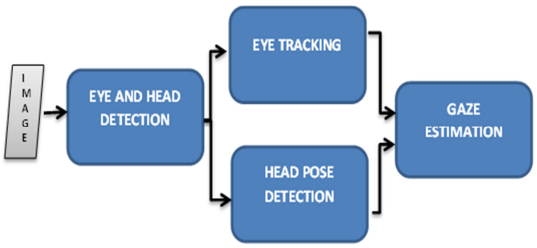

Figura 1. Diagram block of eye tracking system.

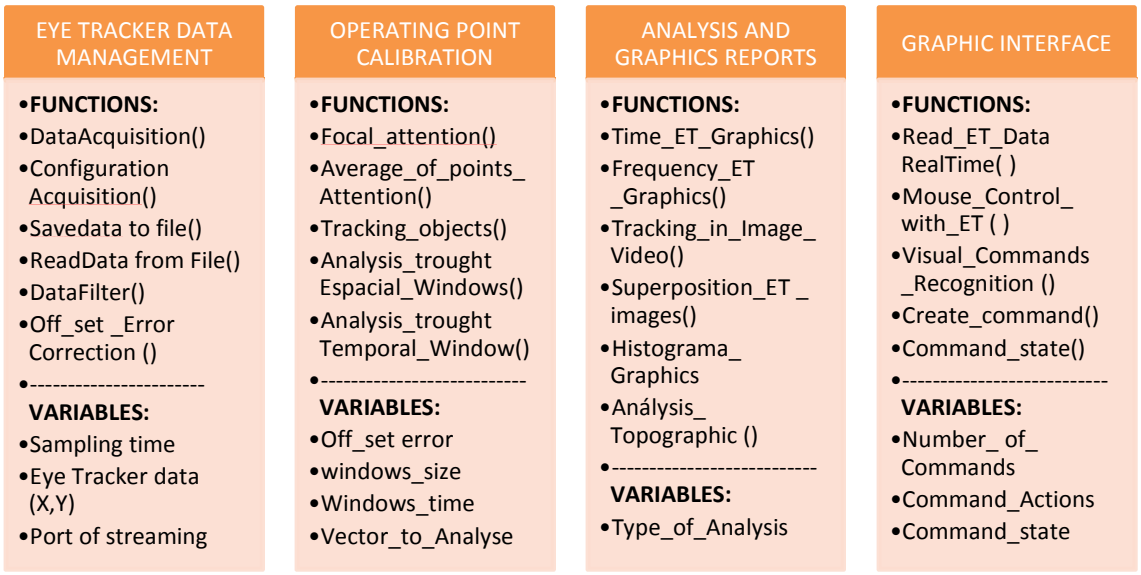

Figure 2: Class diagram of the developed eye tracking interface.

The goal of this work is to develop a general Matlab toolbox interface to facilitate the use of different eye tracker systems in assistive technologies. In this interface four modules were implemented which allow: acquiring and managing data from the eye tracker (ET), calibrating and preparing the system according to the user disability, analyze and generate graphical reports, and facilitate the implementation of graphical interfaces controlled by eye gaze using the eye tracker.

2 Materials and Methods

This research approaches an experimental procedure for development of eye tracking interface (Matlab Toolbox for eye tracking) for assistive applications in order to assess the eye tracker device possibilities how a tool for people with physical disabilities. We propose four modules for processing eye tracker data. Figure 2 shows the class diagram of the system with the functions and variables for the following four modules:

- Acquisition and managing of the ET data.
- Operating point calibration of the ET device.
- Analysis and graphic of the ET data.
- Using of graphic interface with the ET.

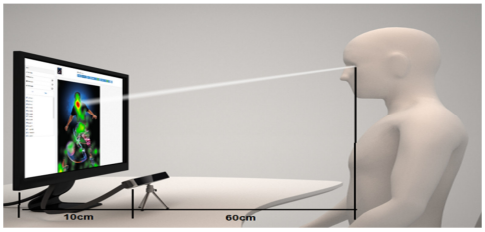

Figure 3: Set-up for the experimental tests.

This procedure had the participation of sixteen healthy adult volunteers (twelve men and four women), with mean age of 28 years old (±5.32). Each volunteer was invited to sit comfortably in a chair positioned in front of the screen of a computer (19 inches) and an eye-tracking device The Eye Tribe (2014), with eyes at 70 cm from screen and at 60 cm from eye-tracker. Figure 3 shows the setup used for the experimental tests. A self- calibration of the eye tracker device was necessary to gather a good data acquisition, which consisted of tracking visually mobile points on the screen and, subsequently, fixed points of known coordinates.

The participant viewed a set of fixed and mobile points and windows in a computer screen of 19 inch and a resolution screen of 1024 x 768 pixels. This work had the approval of the Ethics Committee of UFES, number 1.121.638.

2.1 Data Acquisition and Management Module

This module allows connecting the eye tracker server and getting data, measured and pre-processed eliminating noise and reducing erroneous data from the device, in ender to facilitate data access (write and read eye tracking data from a text file).

The communication process of the module for data acquisition is done by the functions DataAcquisition() and Configuration Acquisition(), via a sub-server in Python, which receives input from Matlab and then sends commands to the EyeTribe server, because Matlab does not have a good multithreading functionality. Figure 4 shows the server configuration for the eye tracker data acquisition.

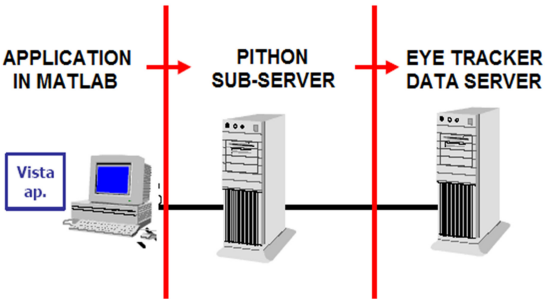

Figure 4: Communication system for eye-tracker data acquisition.

In this work also propose a system with a filter low pass DataFilter(), to filter the signal acquired from the eye tracker and avoiding the difficulty of some users to stare at a fixed point. This filter can be adjusted to the user´s eye speed and eye tracker own latency.

In ender to access and manage the eye tracker information, Save_data to file() and Read_Data_from File() to save and read data from a text file, were developed.

2.2 Operating Set-up Point Calibration

The procedure for eye tracking set up, before using eye tracking to map gaze onto a screen, is calibrated to each user. Sometimes, off-set error, delay in the velocity of tracking, trouble in selecting the right size of the window and doubt about the appropriate speed of command activation. Was developed experimental tests to measure and evaluate those four problems.

In this module was developed several functions to perform these tests: Focal_attention() to calculate the focal point; Average_of_points_ Attention() to calculate the focal point for a data vector; Tracking_objects() to follow trajectories in the screen; Analysis_trought Espacial_Windows() and Analysis_trought Temporal_Window() for analysis in especific sector of the screen or time.

Test 1: Calibration (off-set correction)

Figure 5 shows the experimental test to quantify the off-set error. In this figure five points corresponding to the center, right, left and bottom side and top of the screen (red polygons) are marked. Then the volunteer is asked to look for 5 seconds for each brand in an opposite sequence to the clockwise. The data is saved and the error is calculated using the equation (2) for off-set error estimate of off-set where Ppos is the mark position, ETdata is the estimated position measure with the eye tracker device and max is the maximum data number.

$$Error = \frac{\sum_{n=1}^{\max} Ppos - ETdata}{\max} \qquad (2)$$

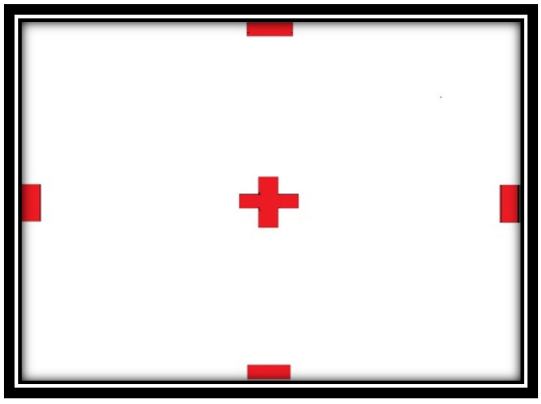

Figure 5: Test to estimate the off-set.

Test 2: Velocity of Tracking

Figure 6 shows the experimental test to estimate latency of the system. In this experiment the user must follow a point that changes the position in the screen every 5 seconds. The aim is to measure how long the delay time from the point is displayed. The user is able to focus the point according to the equation (3) In this equation, Tlatency is the delay time, which is calculated by subtracting the time the user takes to focus Tfoco since the screen appeared; num is the number of points to be evaluate.

$$Tlatency = \frac{\sum_{n=1}^{num} Tfoco - Tscreen}{num} \qquad (3)$$

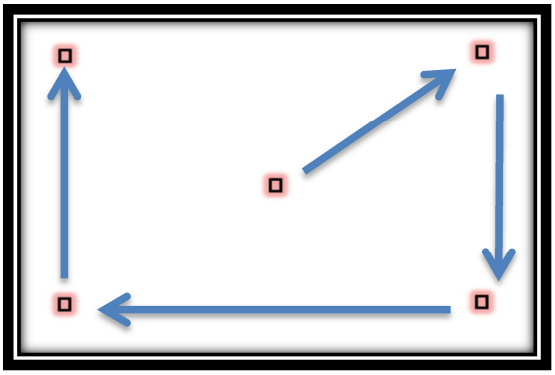

Figure 6: Test to estimate the velocity of tracking.

Test 3: Concentric Windows

Figure 7 shows the experimental test to evaluate different window sizes, which is more appropriated for the function we want to do. In this test, a window with three concentric polygons (150, 100 and 50 pixels size) is displayed for 5 seconds, then the window moves on the screen in 5 different positions. The measured data are evaluated to know what percentage is within each window and know which size would be more appropriate for the user.

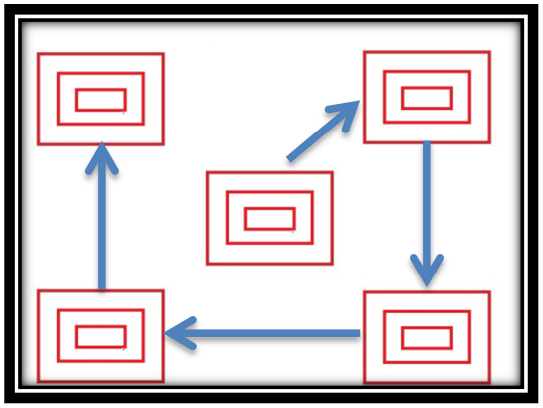

Figure 7: Concentric Windows.

Test 4: Command Rate per time

Figure 8 shows the experimental test to quantify the average command speed. The system has a graphical application with nine commands and asks each user to select the commands in order from one to nine. The number of commands per unit of time is measured the user is able to transmit and the number of errors in those commands.

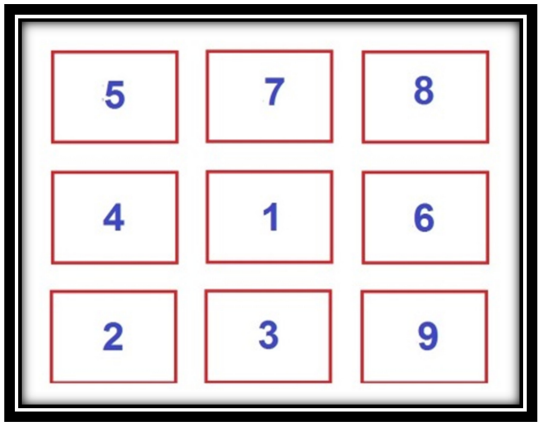

Figure 8: Command rate evaluation.

2.3 Analysis and Graphic Reports

The module for analysis and graphic reports was developed to analyze and display different types of graphics using the functions such as: Time_ET_Graphics() to plot data of eye tracking and variation in time; Frequency_ET _Graphics() to plot the frequency data of eye tracking, Tracking_in_Image_Video() to track in an image or a video; Superposition _ET _ images() to plot superposition with other graphic, image or video; and Histogram_ Graphics () and Analysis_ Topographic () to representate data in histograms or topographic. Figure 9 shows examples of this module.

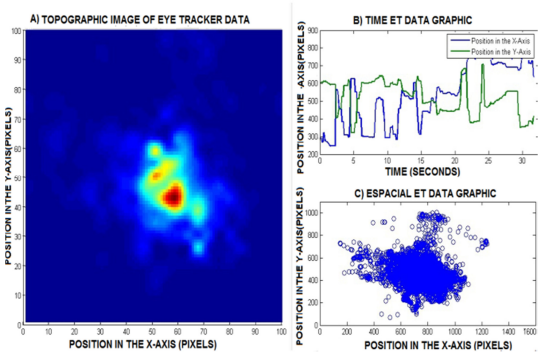

Figure 9: Module for analysis and graphic report.

2.4 Graphic User Interface (GUI)

The graphic interface is a 3 x 3 graphic matrix for a total of 9 commands, a push button to execute the application, another push button for graphic options and a title with information of eye position and activation commands. Figure 10 shows the configurable basic user interface.

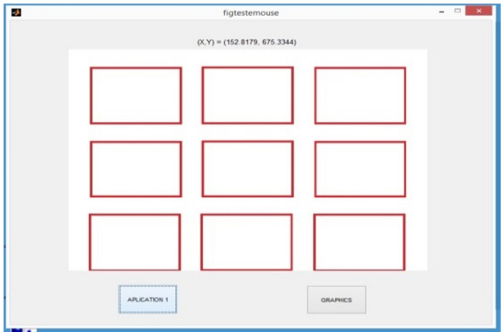

Figure 10: Graphic User Interface.

3 Results

3.1 Data Acquisition and Management Module

Taking into account the sixteen participants the low pass filter designed to eliminate noise was a Butterworth filter for a maximum velocity of 220ms has cut frequency of 5Hz. Figure 11 shows the output signal (red) for the eye tracker and the blue line with the filtered signal, with filtering reduces errors in handling mouse with the eye tracker.

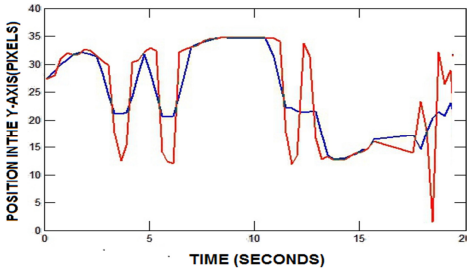

Figure 11: Original and filtered signal of the eye tracker.

3.2. Set-up for Point Calibration

Table 1 shows the errors for the off-set calibration in pixels, centimeters and degrees.

Table 1: Errors for off-set test.

Screen axis	error	Standard deviation
X(Pixels)	14.74	34
Y(Pixels)	1.94	40
X(cm)	3.4	7.8
Y(cm)	1.9	8.9
X(°)	0.3	0.6
Y(°)	0.1	0.7

For the velocity test, the latency for the device system was 40 ms and the eye response delay was 192 ms. Table 2 shows the results.

Table 2: Velocity of tracking test.

Type of delay	Delay (ms)	Standard deviation
Device Latency (ms)	40	33
Eye response delay (ms)	192	45

In Tables 1 and 2 the standard deviation is too high because according to the documentation of the eye tracker device The Eye Tribe (2014), the latency of the device is above 16ms and eye response for healthy people is above 200ms.

Table 3 shows the test with concentric windows. The result for windows of 150 pixels was of 95%, for 100 pixels, 85%, and for 50 pixels, 67%.

Table 3: concentric Windows test.

Windows Size (pixels)	Data within the window (%)	Standard deviation
Size 150	95	6.3
Size 100	85	7.2
Size 50	67	8.7

Table 4 shows the results of the test to estimate the command rate per time of the system and the number of errors for three different command time: 0.5, 1 and 2 seconds.

Table 4: Command rate test.

Command time(seconds)	Command velocity (%)	Number of errors
0.5	1.5	3
1	0.8	1
2	0.4	0

3.3 Analysis and Graphic Reports

Figures 9 and 11 are examples for the analysis and graphic reports module. All the results in this work were calculated using this module.

Other graphics allowed are the histograms. Figure 12 shows an example of validation for six images and four time windows. The histogram calculates the percentage of visual focus time for each image and each time range selected.

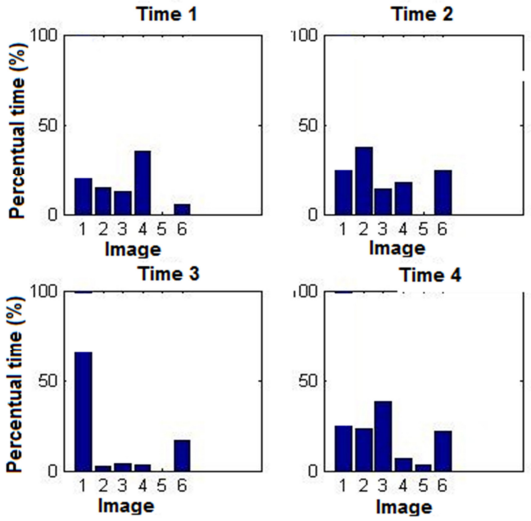

Figure 12: Histogram for different images.

Figure 13 shows an example of superposition of eye tracker data and image or video files. The system allows real time tracking or off-line superposition in image and video files.

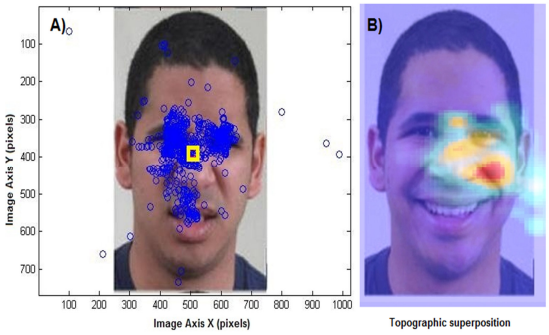

Figure 13: Example of superposition a) superposition ET data and image, b) superposition Topographic ET and image.

3.4 Graphic User Interface (GUI)

The graphic user interface was developed using the results of the different methods to ensure greater reliability and user comfort with the following characteristics: maximum configurable command is nine, but the user can use less commands, the minimal command time is 0.5 seconds, but by default 1 second is recommended for optimum work. The size of each command window is 200 x 150 pixels and the graphic screen is 1024 x 768 pixels to facilitate use in computers, laptops or tablets. The system reliability and the success rate is 90% for 1 second command rate, and 99% for 2 second command rate. Figure 14 shows an example of the graphic user interface. The application allows the control of a robot using eye tracking, in which six commands were configured to control four directions, in addition to the stop options and a menu.

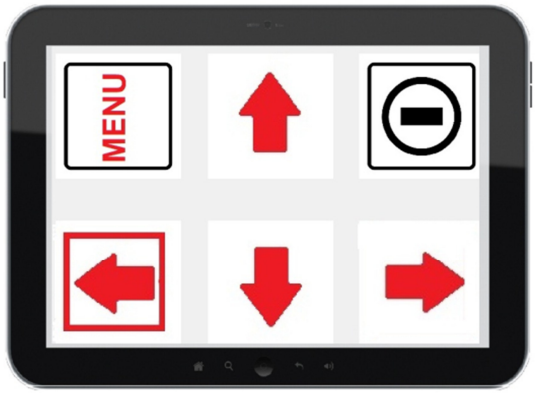

Figure 14 Application for robot control using the graphic user interface.

4 Conclusion

The data acquisition and management module allows connecting Matlab applications with the eye tracker using a Python sub-server. In order to correctly connect them, firstly the eye tracker server should be connected, then the Python sub-server and finally the Matlab application started.

The configuration and calibration of system allow testing and setting-up the optimum configuration for each user, according to disability level, and control motor of eye movement. This module is independent of the eye tracker device and can be used with other eye tracker systems.

The analysis and graphic report module was developed to allow the study of eye tracker data. The graphics can show the variation in time in the focal point. The frequency analysis can show the average attention focus, the superposition graphics show the part of the screen where the user is viewing and the histogram analysis is used to compare different regions of interest in the image.

The graphic user interface (GUI) developed is being used to control device from intelligent environment by people with disabilities, motor intention in robotic walker, study about valence and emotional facial expressions and in recognition of emotions and focus of attention in children with autism.

In addition, the use of eye-tracking can benefit applications requiring to observe and evaluate human attention objectively and non-intrusively including games, operative system navigation, e-books, market research studies, and usability testing.

Acknowledgements

The authors would like to thank Federal University of Espirito Santo (UFES), CNPq, CAPES and FAPES for financial support and scholarships.

References

Calvo, A., Chio, A., Castellina E., Corno F., Farinetti L., Ghiglione P., and Vignola A. (2008). Eye tracking impact on quality-of-life of ALS patients. In 11th International Conference on Computers Helping People with Special Needs, (Linz, Austria, 2008), Springer, pages 70-77.

Card, S. K., Moran, T. P., and Newell. A. (1980). The keystoke-level model for user performance time with interactive systems. Communications of the ACM, 23(7), pages 396-410.

Glenstrup, A., and T. Engell-Nielsen, T. (1995). Eye controlled media: Present and future state. Master's thesis, Institute of Computer Science, University of Copenhagen.

Goulart, C., Rivera, H., Bastos, T., Caldeira, E. (2015) Recognizing Emotions and Focus of Attention in Individuals with ASD Based on Facial Images. Proceedings of the VI Brazilian Conference of Biotechnology, Brasília, Brazil.

Hornof, A. J., and Kieras, D. E. (1999). Cognitive modeling demonstrates how people use anticipated location knowledge of menu items. In Human Factors in Computing Systems: CHI 99 Conference Proceedings, pages 410-417.

Jacob, R. J. (1990). What you look at is what you get: eye movement-based interaction techniques. In Proceedings of the CHI'90, pages 11-18.

Kocejko, T., Bujnowski, A., and Wtorek, J. (2008). Eye Mouse for Disabled. In Conference on Human System Interactions, (Krakow, Poland, 2008), IEEE, pages 199-202.

Lupu, K., Bozomitu, R., Ungureanu, F., and Cehan, V. (2011). Eye Tracking Based Communication System for Patient with Major Neuro-Locomotor Disabilities. In 15th International Conference on System Theory, Control and Computing, (Sinaia, Romania, 2011), IEEE, pages 1-5.

The Eye Tribe. (2014), The Eye Tribe developers guide, University of Copenhagen, http://dev.theeyetribe.com/dev/