Dissertation

RETINA: Real-time analysis and visualization of wearable Eye TrackINg dAta

ANDREAS COSTI

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2018

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

RETINA: Real-time analysis and visualization of wearable Eye TrackINg dAta

Andreas Costi

Supervisor

Prof. Georgios Samaras

Co-Supervisor

Dr. Marios Belk

The individual thesis submitted for partial fulfillment of the requirements for obtaining the degree of Computer Science, Department of Computer Science, University of Cyprus

May 2018

Acknowledgments

I would like to express my sincere thanks to Professor George Samaras, the supervisor of my dissertation, for believing in me and giving me the opportunity to undertake this project.
By giving me this opportunity, during the course of the dissertation, I was able to study in-depth, fields that I am interested in and acquire knowledge that is essential for my academic and professional career.

I place on record, my sincere thank you to Dr. Mario Belk, the co-supervisor of my dissertation, for all the assistance and guidance he provided me during the whole year of my dissertation and for his overall trust in me.

Furthermore, I am also grateful to Argyris Constantinides, for his support and assistance during the development of the dissertation project.

Finally, I would like to express from the depths of my heart a big thank you to my family and friends; their continuous encouragement, support, attention and patience throughout the four years of my studies have been endless and for that I will always be grateful to them.

Abstract

The incremental use of digital devices such as desktops and mobile devices in a multitude of aspects of our daily life, leads these devices to be a necessary tool for every human being. Furthermore, the recent shift towards immersive interaction using Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR) devices suggest that individuals will be interacting constantly with a variety of new systems, devices and technologies.

Consequently, new methods are surfacing in order for researchers to understand the behavior of users and develop interactive systems to visualize complex information in a meaningful way, offering optimal user experience. Eye tracking, is a method used by the researchers in order to understand the visual behavior of users and has become a cornerstone in the development of interactive user-friendly systems by taking into consideration the eye movements of users.

Developing tools for the real-time observation of the visual behavior of the users while interacting with systems and analyzing the raw data received from eye tracking devices with the help of state-of-the-art algorithms, can help usability researchers understand the visual behavior of users and offer them optimal user experience in real-time.

The purpose of this thesis was to create a tool for the real-time analysis and visualization of wearable eye tracking data (RETINA) that can be used by service providers, in order to extract meaningful information regarding the fixations, the fixations' duration and the saccades of users while interacting with a system and visualize this information in a meaningful format with the help of charts and gaze plots. In addition, we aimed to create a Universal Workbench that can work as an extension of RETINA, which can store data regarding not only the visual behavior of a user, but metrics related to other physiological aspects, with the use of wearable smart devices.

By creating a universal workbench platform, researchers and service providers can use it, in order to analyze their users' behavior in real-time and extract meaningful information from this data, with the ultimate purpose of offering their users optimal user experience.

Our Universal Workbench tool was developed and tested by twenty users in terms of its usability, design and attractiveness as well as its visual impact on the user while interacting with it.

Table of Contents

Chapter 1

Introduction

1.1 Thesis Overview

In our modern era, where the technological advancements dictate the world to use desktop or mobile personal digital devices leads these devices to be involved in a multitude of aspects of our daily life such as working, entertainment, financial management, navigation and even healthcare, meaning that they are versatile appliances that store and operate a lot of our data. Consequently, individuals are required to interact daily with a lot of different systems, devices and technologies. However, recently, a shift has been noted from using these traditional ways of interaction based on desktop computers and mobile devices towards immersive interaction using Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR) devices.

As a result, with the increased use of new computing systems and interaction methods, complex information is visualized and presented through visual interfaces as a mean of information presentation, in order for the user to easily understand complex information. However, even though the information might be presented in a simple and understandable format, sometimes the users seem to not be able to process this information properly. This happens due to the fact that users tend to seek, retrieve, process and understand information differently based on the way a user navigates while interacting with a system. Based on all of the above, today, eye tracking has become a cornerstone in research, especially in the context of AR/VR/MR and how researchers can define and offer the

optimal user experience as well as how eye tracking can be used by marketing groups to design effective interactive systems for advertising their products.
Being able to observe and record the visual behavior of the users during the interaction with a system has been occupying researchers for decades and is primarily used to identify where the users are looking, where the users are focusing their attention (fixating) and what they don't like to observe and "rapidly" change their attention to another area of interest.

1.2 Problem Statement

As mentioned above, since eye tracking has grown in the recent years and more affordable and non-intrusive eye tracking technologies exist, developers are faced with the dilemma regarding which approach to follow in order to analyze eye tracking data recorded in a session of interaction between a user and a system. There are two approaches in order to analyze the eye tracking data recorded.
The first approach is to analyze all the data recorded following an offline analysis process, whereas the second approach is to analyze the data during the eye tracking recording process following a real-time analysis process. Both approaches have pros and cons but most of the time the developed applications are following the offline analysis process, as it is easier for the application to be implemented. Such applications will take as input all the raw data of an eye tracking session and visualize the results regarding the visual behavior of a user after passing through state-of-the-art implemented algorithms. This makes the developed applications useful to usability researchers in order to understand the visual behavior of users and develop user-friendly applications and systems in order to offer optimal user experience after the interaction of a user with the system.
Therefore, by developing applications following the offline analysis process, the real-time analysis process of eye tracking data is not used as much, making it a research field where a lot of applications may be developed and find specific usefulness regarding the analysis of wearable eye tracking data in a real-time fashion (e.g. analyze wearable eye tracking data and extract information regarding fixations, fixations duration, saccades etc. and then visualize these analyzed results based on the visual field of the user in real-time while interacting with a system, or visualize them with the help of charts which will help both researchers and the simple users understand their visual behavior).

1.3 Motivation

By taking into consideration all of the above, our aim was to tackle the real-time aspect of the analysis of wearable eye tracking data. In order to do that we decided that it was best to create a tool for the real-time analysis and visualization of wearable eye tracking data.
Currently a lot of companies, both commercial and research, are involved in the research of analyzing and visualizing wearable eye tracking data and developing applications both for the offline analysis as well as the real-time analysis of these data, but as we stated above, the real-time analysis process can find specific fields of applications, in regard to the extraction of useful information in real-time from these raw eye tracking data. Based on this fact, we found that it would be a good opportunity to tackle this problem and present our implementation of a system which is based on the real-time analysis process.
With the rise of the wearable eye trackers, the desktop eye tracking devices as well as the shift towards the immersive interaction between users and systems, the implementation of a tool that would help analyze and visualize the visual behavior of a user in real-time is critical, as it would help researchers create algorithms to understand the raw data received from these devices regarding the eye movements of users and help them personalize in real-time the systems created in order to offer the users optimal user experience.

1.4 Scope of the Thesis

The scope of this thesis is to take all these facts into consideration and based on some related research works which have already tackled the process of analyzing gaze data in real-time, create a tool which essentially will be used as a platform for the real-time analysis and visualization of wearable eye tracking data. By adjusting algorithms based on already presented works and creating this tool, we can make it possible for the development community to shift their focus to the real-time analysis of the eye tracking data as well, in order to create systems that can be adjustable to the users' personal liking based on understanding the users' visual behavior in real-time while interacting with a system.

Furthermore, in the scope of this thesis, by developing this tool to be as extensible as possible, this led us to develop a universal visualization workbench that can support the visualization of data, not only regarding the visual behavior of the user received from wearable eye tracking devices, but for visualizing data received from wearable smart-devices regarding the physiological state of a user.

Chapter 2

Related Works

Introduction

In this chapter I will present different technologies, using different methods for observing the visual behavior of users and how by collecting and analyzing this data either in a real-time or an offline fashion, they can visualize later the behavior of the users and come to conclusions regarding the collected data. Such technologies include wearable headsets with mounted cameras recording the pupils as well as recording the surrounding environment of the individual wearing the headset. Other technologies use desktop cameras that record the visual behavior of the users while interacting with desktop interfaces. Some of the companies using these technologies are Pupil Labs, Tobii and GazePoint and research-based companies such as SensoMotoric Instruments (SMI), EyeLink, Ergoneers etc.

2.1 Pupil Labs

Pupil Labs is a company based in Berlin, offering open-source coding and hackable eye tracking solutions. Pupil Labs uses a wearable headset with mounter cameras for recording eye movements using either a monocular fashion where the gaze is recorded and streamed or binocular to estimate where the individual is looking in 3D as well as recording and streaming the gaze of the individuals with a gaze sampling frequency of either 120Hz or 200Hz. The headset uses a front camera as well, to record the field of view of the participant which helps in the egocentric vision research.

2.1.1 Pupil Labs Fixation Detection Algorithms

The I-DT (dispersion based) algorithm as described in Chapter 3, was adopted by Pupil Labs in their real-time fixation detection algorithm for their eye tracking device but implemented a bit differently.

Their plugin detects fixations based on a dispersion threshold in terms of degrees of visual angle with a minimum duration and a minimum average pupil confidence. The fixations are published as soon as the threshold constraints are met which might lead to a series of overlapping fixations.

The thresholds set are:

Maximum Dispersion (spatial, degree): is the maximum distance between all gaze locations during a fixation.

Minimum Duration (temporal, milliseconds): is the minimum duration in which the dispersion threshold must not be exceeded.

Confidence Threshold: Data with a lower pupil confidence will be disregarded.

Pupil Labs chose to implement a second version of the algorithm as well, for the offline analysis of the gaze data in order to detect fixations. This version detects fixations based on a dispersion threshold in terms of degrees of visual angle within a given duration window. The algorithm tries to maximize the length of the fixations classified within the duration window and this results in fixations not overlapping.

The thresholds set are:

Maximum Dispersion (spatial, degree): is the maximum distance between all gaze locations during a fixation.

Minimum Duration (temporal, milliseconds): is the minimum duration in which the dispersion threshold must not be exceeded.

Maximum Duration (temporal, milliseconds): is the maximum duration in which the dispersion threshold must not be exceeded.

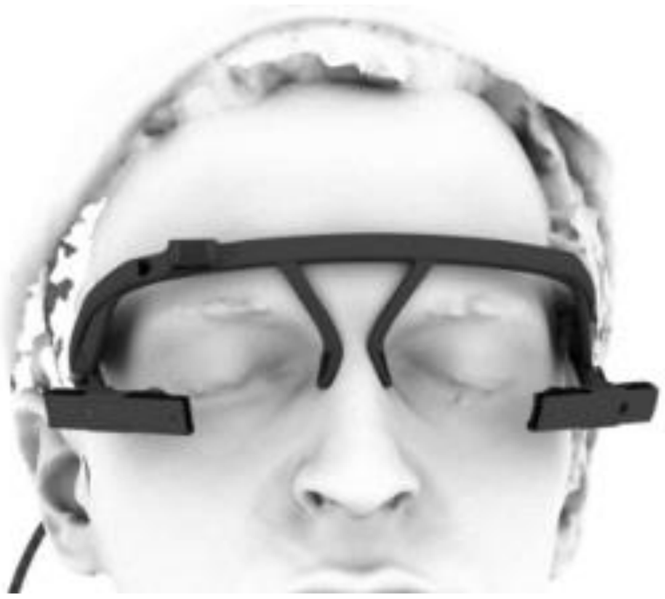


Figure 2.1 - Pupil Labs Eye Tracker

2.2 Tobii

Tobii is a Swedish high-technology company that develops and sells products for eye control and eye tracking. ***Tobii Pro Glasses 2*** is the wearable eye tracking device developed by Tobii. It is designed to capture natural viewing behavior in any real-world environment while ensuring eye tracking robustness and accuracy. It is mentioned by the company that it is possible to be combined with biometric devices for even deeper insight and understanding of human behavior.

In contrast with Pupil Labs, Tobii does not offer their software in an open-source format. Alongside the device comes the ***Tobii Pro Lab*** application, where the gaze data recorded during a session can be exported to, in order for the researchers to gain a deeper insight after analyzing the data. Rather than offering their software in an open-source format, they offer the ***Tobii Pro Glasses 2 API*** which is a free application programming interface for creating wearable eye tracking solutions that can work with the Tobii Pro Glasses 2. They provide a real-time access to the data streamed from the device as well as store this

data for an offline analysis after the session's recording ends. The API offered by Tobii is platform and programming language independent. By providing real-time access to the data streamed from the device they give the opportunity to the developers to develop real-time applications in order to process the raw data to extract meaningful information during an eye tracking session.
Tobii Pro Glasses 2 is a wearable eye tracking headset device with mounted cameras (2 cameras per eye) for recording gaze data based on a 3D eye model either using a gaze sampling frequency of 50Hz or 100Hz. The tracking technique used is based on corneal reflection, using binocular recording and dark pupil tracking is offered. A full HD, wide angle front camera is also used to record the field of view of the participant wearing the eye tracker.

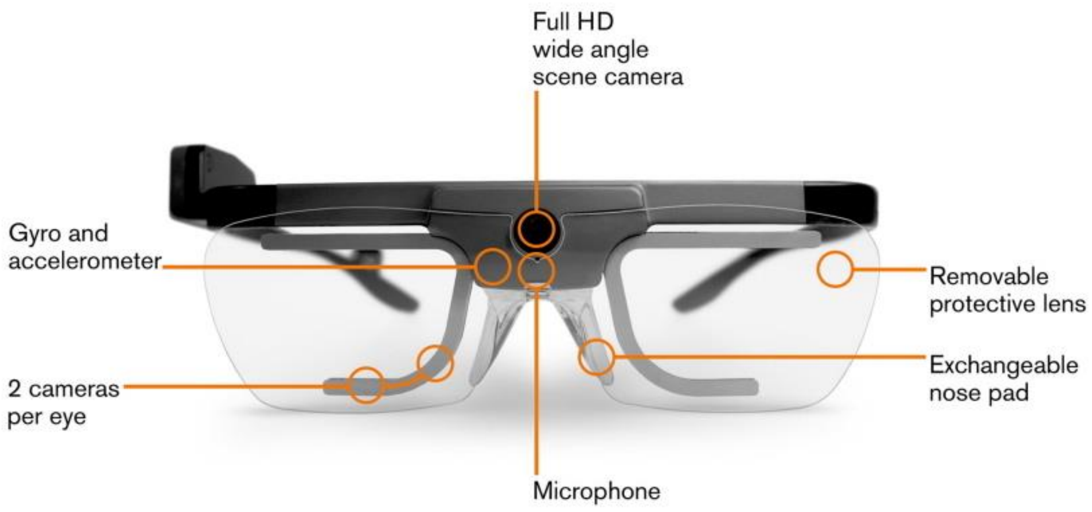


Figure 2.2 - Tobii Pro Glasses 2

2.3 GazePoint

GazePoint is a company offering eye tracking solutions that can be used for usability and academic research or even as an assistive technology to provide people with disabilities new opportunities to interact with the world.

GazePoint's GP3 HD eye tracking device, in contrast with the aforementioned devices, is a desktop research-grade eye tracker utilizing a machine vision camera for its imaging and processing system. It is designed to capture the eye movements of a user while interacting with a desktop interface and a system in order to provide usability researchers insight in how to develop new applications offering users the optimal user experience.
Just like Tobii, GazePoint does not offer their software in an open-source format. However, alongside the desktop eye tracker comes the ***GazePoint Analysis UX Edition*** application, which is the application used for the analysis of the recorded gaze data of a session. It offers functionalities such as the creation of heatmaps, gaze fixation paths, screen capturing, data aggregation, definition of static as well as dynamic Areas of Interest (AOIs), etc. Furthermore, since their code is not open-source, they include an API/SDK which is used for eye tracking software development. Development of applications using the offered ***Open Eye-gaze interface API*** is programming language independent as long as the platform allows TCP/IP communication. The analysis application however is offered only for Windows.
As mentioned above, ***GP3 HD*** is a desktop eye tracker, used for recording the gaze data of a user while interacting with an interface on a desktop display, using a gaze sampling frequency of either 60Hz or 150Hz.

Figure 2.3 - GazePoint GP3 HD

2.4 SensoMotoric Instruments (SMI)

SMI, is a German provider of dedicated computer vision applications with a major focus on eye tracking technology. Its primary focus is in the academic, medical and scientific research, as well as providing eye tracking professional solutions and OEM applications.

Their eye tracking solutions can be combined with other motion tracking systems and can be integrated into virtual environments or simulators.
SMI is tried and tested as experienced providers of eye tracking equipment and was recently acquired by Apple.
Their technology is based on dark pupil and corneal reflection tracking and the cameras installed on the SMI eye tracking devices can detect faces, eyes, pupils as well as the corneal reflections from the infrared light sources. They can calculate eye movements, gaze directions and areas of interest. These devices use a sampling frequency which ranges from 30Hz up to the KHz range.
The company has three main product lines which are the mobile ***Eye Tracking Glasses (ETG), remote eye tracking systems (RED) and the tower-mounted system (Hi-Speed)***. The software offered for experimental design and data analysis is called ***Experiment Suite*** and comes in different packages depending on the user’s research interests which can vary from research in psychology to linguistics and education. This company offers synchronization with biometric sensors as well as the observation of experiments and the annotation of the behavior of a user in real-time.

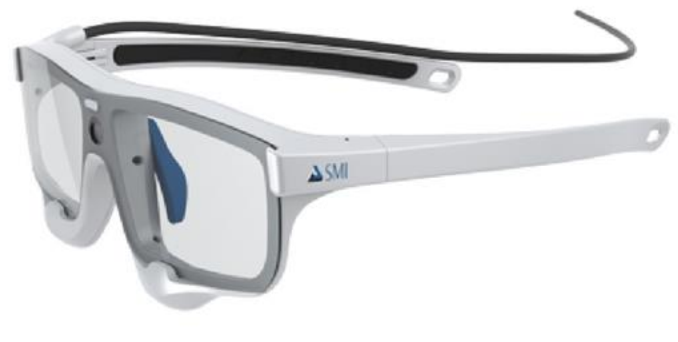


Figure 2.4 - SMI ETG

2.5 EyeLink

EyeLink eye tracking units are made by SR Research, that provides several solutions, including portable and head-mounted systems. SR Research has been exclusively dedicated to supporting and responding to the needs of the eye-movement research community in order to enable the scientific exploration in a variety of domains.
The ***EyeLink 1000 Plus*** is their most precise and accurate video-based eye tracker which uses a sampling frequency of up to 2000Hz. It is flexible and customizable, offering multiple mounting options, interchangeable lenses and multiple head free-to-move remote configurations.
They offer different types of software for the analysis of the data recorded which are the ***Experiment Builder software, the Data Viewer software and the Third-Party Display Software / Hardware Integration***, each one offering different capabilities based on the needs of the user. The Experiment Builder can be used as a drag-and-drop graphical programming environment for creating computer-based psychology and neuroscience experiments. The Data Viewer can be used for viewing, filtering, and processing eye gaze data recorded with the EyeLink eye tracking devices and the Third-Party software and hardware integration software can be used and controlled by a wide range of software and can be integrated with many other recording devices which allows EyeLink systems to be used seamlessly with a range of software products.

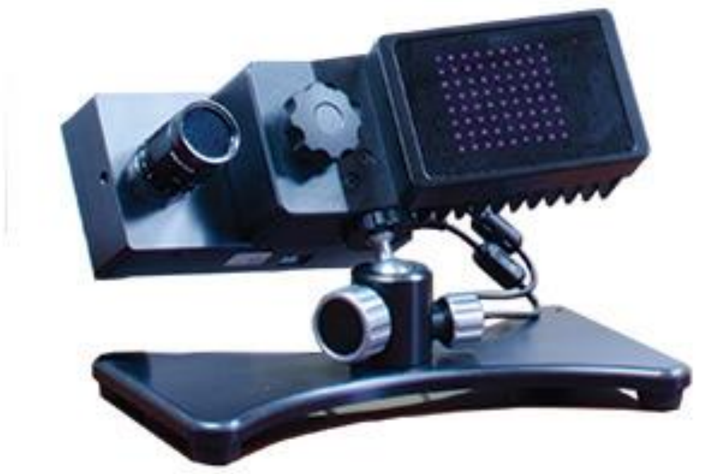


Figure 2.5 - EyeLink Plus 1000

2.6 Ergoneers

Ergoneers was founded in 2005 in Munich, and now provides portable eye tracking solutions as well as large-scale vehicle driving simulation setups. The company's aim is to aid in real-world driving studies with benchmarking vehicles, in order to develop, manufacture and distribute systems for the measurement and analysis of the human behavior in order to optimize the human-machine interaction.

Their primary solution comprises of the ***360-degree D-Lab*** which is an extensive software platform for capturing and analyzing human behavior. Offering different software modules, it can synchronously measure and analyze eye tracking data that can be used for professional eye tracking studies in real world or virtual environments.

Offering a great variety of eye tracking solutions, a wearable eye tracking system provided is the ***Dikablis Glasses 3,*** which enables binocular eye tracking with high measurement accuracy and standardized calculation of characteristic values, using a sampling frequency of 60Hz. The wearable eye tracker is fully supported by the D-Lab measurement and analysis platform which can be used to record and analyze data in several different ways based on the needs of the user.

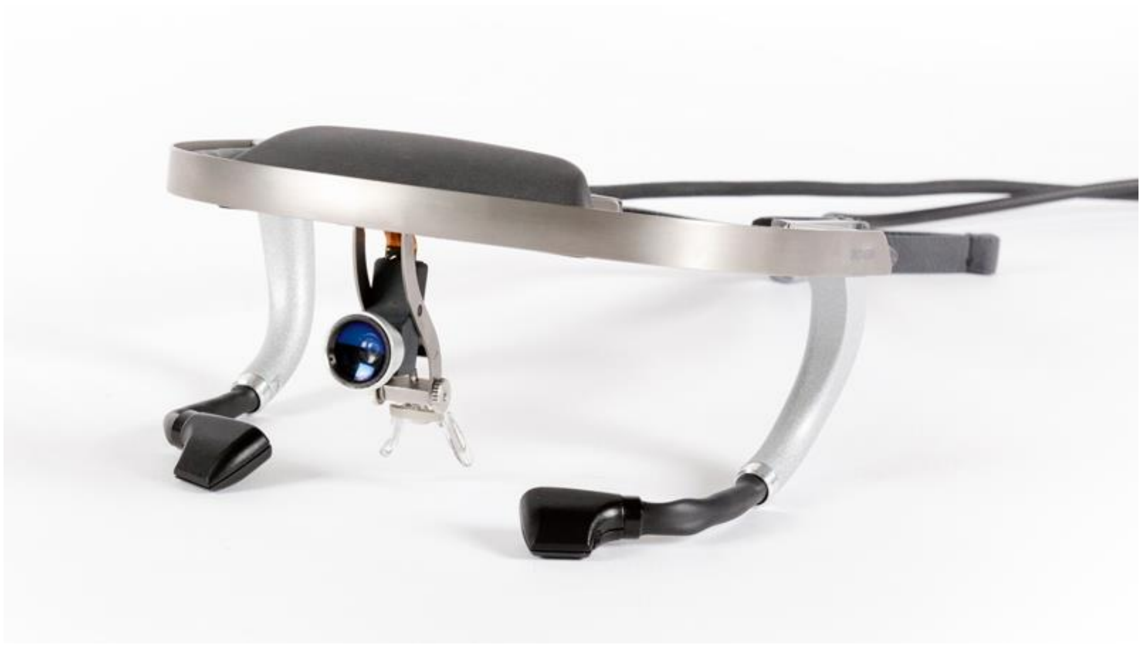


Figure 2.6 - Dikablis Glasses 3

Chapter 3

Background Theory

3.1 Introduction

This chapter presents state-of-the-art eye gaze data analysis algorithms that were used to back up our algorithms regarding the detection of fixations and saccades. The publications containing these algorithms will be reviewed as well as the most important metrics related to eye tracking which we also used in the development of our system. Finally, the pseudocode describing the structure of the algorithms presented in those papers can be found in Appendix A of this thesis. How our algorithms were adjusted based on these algorithms and how they were used for the real-time elicitation of the wearable eye tracking data, regarding fixations, fixations' duration and saccades will be described in Chapter 4 where we will discuss the implementation of our system.

To be able to develop our system (RETINA), we firstly had to understand and develop the algorithms that would determine the visual behavior of the user in a real-time analysis process and in particular the metrics we were interested in were the fixations, the fixations' duration, the total number of fixations and the saccades of the users while interacting with our system. Some other important metrics related to eye tracking and the analysis of the users' visual behavior in a real-time analysis process are the gaze transition entropy as well as the stationary entropy. In order for our algorithms to be robust, we had to find publications to base our algorithms on and back up our work. In this chapter the eye gaze metrics that will be presented are the two types of entropy (gaze transition entropy and stationary entropy), fixations and saccades.

Eye Tracking is a very popular technique, which has started gaining popularity over the past decade [3] due to the technological advancements in this period of time. The development of more sophisticated eye tracking equipment has led to a growth of the research [4] in investigating cognitive and visual processes as well as providing insight in understanding the users' visual behavior while interacting with computer systems [1]. Eye trackers are designed to capture gaze information related to fixations and saccades [1]. Fixations occur when a person pauses over informative regions of interest meaning that the individual's vision is focused on a particular point over a period of time [1, 3]. On the other hand, saccades describe a person's rapid movements between fixation points. Alongside these two primary parameters of eye tracking comes fixation duration; the amount of time an individual fixated on an Area of Interest (AOI) and these three factors are used to understand a user's visual behavior while interacting with a system.

3.2 Entropy

There are two leading eye movement metrics for quantifying eye movement transitions [5] between AOIs based on the aforementioned metrics. The first are scan-paths, represented by an ordered sequence of fixations by computing the similarity between those sequences of fixations. The second are heatmaps, represented by Gaussian Mixture Models (GMMs), indicating frequency of fixation localization [5].

$$H_c = -\sum_{i=1}^{n} p_i \sum_{j=1}^{n} p_{i,j} * \log_2 p_{i,j}\,, \quad i \neq j$$

Equation 3.1 - Entropy equation

Entropy [5, 6], provides a statistical dependency in the spatial pattern of fixations represented by the transition matrix and can be used to compare one transition matrix to another (the probability of viewing the *j*th AOI given the previous viewing of the *i*th AOI).

There are two types of entropy [5]. The first type is *gaze transition entropy* which is analogous to the entropy and is calculated for individual subjects' transition matrices and the second type is *stationary entropy* which is calculated for individual subjects' stationary distributions. Having these two types of entropy allows achieving prediction of AOI transitions and overall distributions of eye movements [5]. Higher value of stationary entropy means that the visual attention of the user is spread evenly among AOIs whereas a lower value of stationary entropy suggests the user's fixations were concentrated on certain AOIs. On the other hand, higher transition entropy suggests more randomness in eye movements and more frequent switching between AOIs.

3.3 Fixations

As mentioned above, fixations occur when an individual's gaze stops moving, lingering long enough to process the information seen in a certain AOI. The analysis of fixations requires some kind of identification, therefore the existence of an algorithm able to translate the raw eye movement data points to fixation locations on the visual display recorded by the eye tracking device. Fixation identification is a convenient method of minimizing the complexity of the raw data received from the eye tracker while retaining the most essential characteristics related to the recorded fixations in order to understand the cognitive and visual processing behavior of the user.

Our fixation detection algorithm is based on [3] in which the authors define different categories of fixation detection algorithms and compare them in terms of *accuracy, speed, robustness, how easy it is to be implemented and the number of parameters* which the algorithm requires as input [3].

One of them describes *dispersion-based algorithms (I-DT).* I-DT is based on the fact that due to the low velocity of the fixation points they tend to cluster closely together and identifies fixations as groups of consecutive points within a particular dispersion or maximum separation.

Fixations tend to have a duration of at least 100ms; however, dispersion-based identification techniques employ a minimum duration threshold of 100-200ms.

As explained in this paper the I-DT uses a moving window that spans consecutive data points checking for potential fixations based on the dispersion threshold as well as the duration threshold [3]. I-DT then checks the dispersion of the points in the window by summing the differences between the points' max and min *x* and *y* values; which leads to dispersion being equal to:

$$D = [\max(x) - \min(x)] + [\max(y) - \min(y)]$$

Equation 3.2 - Dispersion equation

If the dispersion is greater than the dispersion threshold, the spanned window does not represent a fixation thus expanding the window by one point until the dispersion is lower than the dispersion threshold in order to register the centroid of the window points with the time and duration as a fixation.

The I-DT algorithm for identifying fixations in a real-time process, as described in this paper, is fairly simple, straightforward and fully explained which makes it easy to implement.
The pseudocode describing the I-DT algorithm presented by the authors of this paper [3] can be found in Appendix A.

3.4 Saccades

Saccades, as already mentioned above, describe an individual's rapid eye movements between fixation points [3]. By employing an algorithm to identify fixations as the one described previously, in order to translate the raw eye movement data points to fixation locations [3], means the significant reduction of size and complexity of the data recorded and the removal of raw saccadic data points. Identifying saccades is based on the fact that saccadic movements have high velocities; thus, the term rapid eye movement.
Our real-time analysis tool to identify and quantify saccades based on the velocity of the eye movements is based on [7] in which the authors suggest that in order to provide saccade data from the raw coordinate data acquired by mobile eye-tracking devices an algorithm is required making the velocity-based saccade identification the simplest method to extract and understand this kind of data [7].

The separation of the saccadic movements and the fixations is based on their point-to-point differences of their velocities. In order to identify an eye movement as a saccade the point-to-point velocity has to typically be > *300º / sec*. whereas identifying an eye movement as a fixation the velocity usually is < *100º / sec*. making the identification process quite simple and robust [7].

The algorithm presented in this paper is split into steps:

Step 1:

The algorithm begins by calculating the point-to-point position change of the *x* and *y* coordinates for each frame in the raw data which calculates a distance in pixels between 2 consecutive frames. The calculated distance is based on the following equation:

$$Distance = \sqrt{(x_{t1} - x_{t2})^2 + (y_{t1} - y_{t2})^2}$$

Equation 3.3 - Distance equation

Then the velocities and accelerations are calculated from one frame to the next or previous; velocity depends on the distance and acceleration depends on the calculated velocities of the 2 frames and are given by the following equations:

$$Velocity = \left(\frac{Distance}{Time}\right)$$

Equation 3.4 - Velocity equation

$$Acceleration = \left(\frac{Velocity_{t1} - Velocity_{t2}}{Time}\right)$$

Equation 3.5 - Acceleration equation

Step 2:

The raw *x* and *y* coordinate data represented in pixels are converted to degrees based on a pixel to degree conversion ratio of 1:0.31 [7].

After the conversion, data with velocities over *1.000º / sec*. or acceleration over *100.000º / sec²*. were disregarded and considered as flickers or blinks.

Step 3:

After calculating the velocities and accelerations for each frame of the raw data recorded, based on a certain velocity threshold (> *240^o / sec*. (*5^o*)), the algorithm classified the point as a candidate saccade.

After the comparison with the acceleration threshold (> *3.000^o / sec^2*.) and the duration threshold (< *100ms*), the event would be classified as a saccade, otherwise as a fixation.

The algorithm suggested in this paper is quite easy to implement, straightforward, fully explained and by taking into account the velocity difference between classifying a saccade and a fixation makes it robust and accurate.
The pseudocode describing the identification and classification algorithm of saccades presented by the authors of this paper [7] can be found in Appendix A.

Chapter 4

Design and Development of RETINA

In this chapter I will present the RETINA system which is the tool for the real-time analysis and visualization of wearable eye tracking data. I will also explain the design of the algorithms based on the algorithms presented in Chapter 3 and how we adjusted them to fit our needs, for the real-time analysis of the wearable eye tracking data. Furthermore, I will explain the architecture of the system, the user interface of the system and the different functionalities offered, the architecture of our database as well as the different technologies and tools used for the development.

4.1 System Architecture

The system we chose to implement was a web application in order to be platform-independent, so an individual having the proper devices required could use it and interact with it as easily as possible.

For this reason, the architecture we chose to use for our system was the MVC architectural pattern. The technologies and tools used to implement the system will be described at a later stage.

Model-View-Controller

MVC is an architectural pattern which has been widely adopted in the development of many systems and mainly in the development of web applications. MVC stands for *Model-View-Controller* which is commonly used for developing an application and dividing it into three interconnected parts. This is basically done to separate internal representations of information from the ways information is presented to and accepted from the user. As said above, the main components of the MVC architectural pattern is the *Model*, the *View* and the *Controller*.
The *model* is the central component of the pattern. It expresses the application's behavior in terms of the problem domain which is independent of the user interface. It directly manages the data, logic and the rules of the application, as well as receives input from the controller based on the user's input.
A *view* can be any output representation of the information encapsulated in the model in a graphical format, understandable to a simple user who interacts with the system and just wishes to see the information in a meaningful way, usually represented in tables, charts, diagrams etc.
The *controller* is responsible for receiving any kind of input, usually the user's input regarding an action that needs to be performed in a view or the model. The controller responds to the user's requests and performs interactions on the data model objects.

The main goal of using the MVC architectural pattern for the development of applications is that it allows developers to tackle some major problems regarding efficiency in code reuse as well as parallel development.
By creating components that are independent of each other, developers are able to create these components in an abstract way, so they can be reused in the development of other applications quickly and efficiently with minimal effort in the integration process based on the requirements of the new application. An example of how efficient this reuse can be, is that a view of one application can be refactored for another application with

different data because the view is simply handling how the data will be presented to the end user.
Using the MVC architectural pattern, which decouples the various components of the application, developers can work in parallel on different components without blocking the work of each other. The simplest example is the separation of the front-end and the back-end development. The front-end development team will be able to design and test the layout of the application and the different views to be presented to the end users whereas the back-end development team will be able to design the structure of the data and how the users will be interacting with it without the user interfaces being available.
By using the MVC architectural pattern properly many advantages are offered, some of them described above, such as parallel development and code reusability. MVC also offers *high cohesion* enabling logical grouping of related actions on a controller together and *low coupling* amongst models, views and controllers.

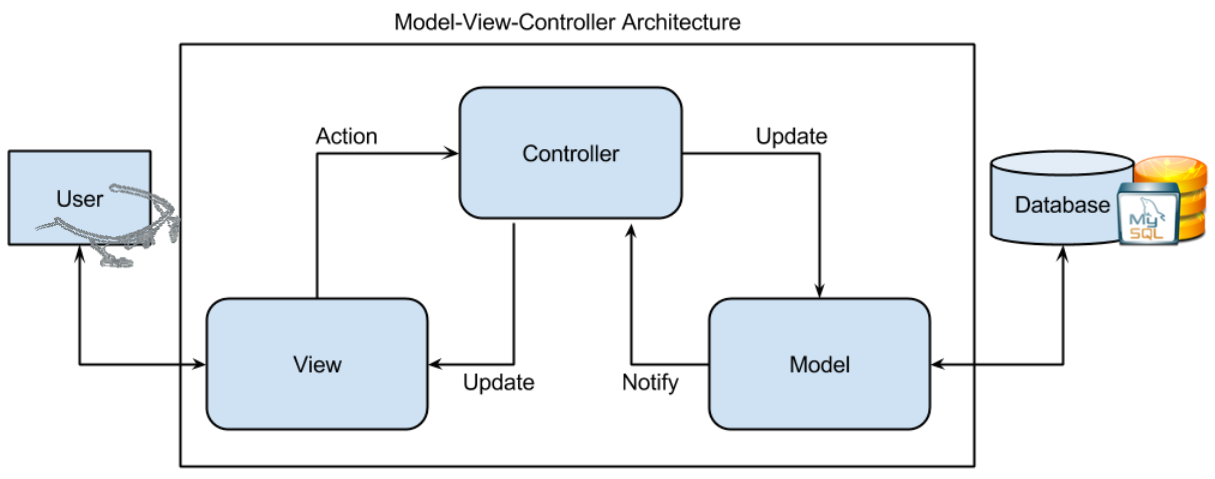


Figure 4.1 - Model-View-Controller Architecture

4.2 System Requirements

In the process to define our system’s requirements, firstly we had to split them into *functional* and *non-functional* requirements.

Functional requirements, which are used to define a function of a system or its components, may be calculations, technical details, data manipulation and processing and other specific functionalities that define what a system is supposed to accomplish.
Functional requirements are supported by *non-functional* requirements, which state the constraints on the design or the implementation of the system. Such *non-functional* requirements are performance, security, reliability and extensibility requirements.

4.2.1 Functional Requirements

As stated above, functional requirements express the way a system must work, the functionalities supported by this system and offered to the end-user.
Some of these functional requirements are:

- An individual must be able to register to the web application.
- An individual must be able to login to the web application after successfully creating an account.
- An individual must be able to follow and understand the steps presented after successfully logging in to the web application.
- An individual must be able to start the capture process by wearing the Pupil Labs eye tracking device and by calibrating the device to have a strong confidence in pupil detection (*~0.6* or greater).
- An individual must be able to view the charts regarding the metrics recorded in an eye tracking session.
- An individual must be able to view the fixations gaze plot after the end of a session.
- An individual must be able to view the fixations' duration gaze plot after the end of a session.
- An individual must be able to upload the results of a session to the online database for further analysis in our universal workbench tool offered which offers more functionalities regarding data manipulation and visualization.

4.2.2 Non-functional Requirements

As stated above, non-functional requirements specify criteria that can be used to judge the operation of a system as well as state constraints on the design or implementation of the system.

Since we wanted to design and implement a real-time analysis and visualization tool for wearable eye tracking data, our system had to ensure some basic non-functional requirements, in terms of performance, efficiency, reliability and since this tool will communicate with another online version for further analysis of the data recorded, extensibility had to be ensured.

Some of these non-functional requirements are:

- *Performance:* The tool has to offer stable performance and complete the assigned work in short periods of time in order to maintain the real-time aspect of the system we wanted to implement without taking into consideration the workload assigned.
- *Extensibility:* The tool has to be extensible, meaning that new features can be added easily with minimal effort from the developers' point of view.
- *Availability:* Since the system is developed as a web application, it must be available to the end-user at any given time and be committed to complete the work assigned.
- *Security:* The tool has to be secure from either cyber or physical attacks from people who potentially want to harm the system. Maximum security for the personal data of the users has to be ensured.

4.3 Flow of Information

In order for an individual to use our tool, RETINA, for the real-time analysis and visualization of the wearable eye tracking data, the Pupil Labs eye tracking device must be acquired. Relying on this, the eye tracking device offers particular basic functionalities based on the cameras installed for capturing the pupils' gaze positions and the camera installed for recording the visual field of the user. The algorithms developed by the company are used to detect the gaze of the user wearing the eye tracking device and process this information, find fixations, fixations' duration and other useful metrics in regard to the visual behavior of the user by using the ***"Pupil Capture"*** application. However, a lot of this processing and the exporting of the useful data is done in an offline fashion after the end of a session by using a separate program called ***"Pupil Player".***

What we wanted to do was, to receive the data recorded from the eye tracking device, in a real-time fashion, and also in a real-time process analyze the data to find the fixations, fixations' duration, fixation count and saccades, in order to be able to visualize them using charts and gaze plots.

The process followed by a user in order to be able to operate RETINA is that the user must login into the system – register first if no account exists – and follow the on-screen instructions in order to start the interaction with the system.

The on-screen instructions are split into three steps which are explanatory, so the user can understand why these steps are important for the eye tracking data recording process; (i) The first step is that the clock of the eye tracking device has to be synchronized with the system's clock, so the timestamps received are in *UNIX epoch* format. (ii) The second step is that in order to receive correct gaze data from the eye cameras and the confidence level of the pupil detection algorithm is passing the set threshold, a proper calibration has to be made so the eyes of the user are recognized by the eye cameras. When the eyes are found by the cameras and the confidence level is high enough the calibration process starts so the eye tracking device can map correctly where the user is looking in the visual field. (iii) The third step of the instructions is notifying the user that after clicking the button presented on the screen the capture of the raw data will start and the algorithms will start working in order to classify fixations and saccades, find the fixations' duration and fixation count. While the algorithms are running and when a classification occurs the useful data regarding the classified metric is saved in our database and the visualization of the data occurs in a real-time process on charts. We decided that in order for our system to perform better, the refresh rate of the charts presenting the data in real-time should be set at ***5 seconds.***

When the session ends, the user has the choice to view a fixations gaze plot which maps the fixations recorded by our algorithms on the visual field of the user with an ***'X'***, as well as view a fixation duration gaze plot which maps the fixations recorded by our algorithms on the visual field of the user with a circle based on the duration of each recorded fixation;

if a fixation occurred for a longer period of time the mapped fixation's circle will have a bigger radius than a fixation with lower duration.
The user also has the possibility to send the data recorded to our online database and visualize them in our RETINA Universal Workbench which offers more functionalities regarding data manipulation and visualization which will be described at a later stage.

4.4 RETINA Web Application

A fundamental tool that was developed in the scope of my Bachelor Thesis was the system responsible for the analysis and visualization of the wearable eye tracking data. The RETINA web application serves as a mediator for the user-friendly interaction between the end-user and the Pupil Labs eye tracking device.
Every person who utilizes the web application has the opportunity to observe the metrics received from the eye tracking device, after passing through the algorithms we designed, and the end-user has the chance to perceive the fixations' duration and the fixation count in real-time. The user also has access to view a fixations gaze plot as well as a fixations' duration gaze plot based on the interaction with the system.
It is worth mentioning that this web application is a link to transmit the processed data regarding the wearable eye tracking metrics received to the RETINA Universal Workbench, if the user wishes to, for further processing and data visualization and manipulation functionalities.

4.5 System Design – User Interface

Firstly, it is a requirement that the Pupil Labs eye tracking device is connected to the desktop or laptop personal computer in order for the system's algorithms to be able to receive the metrics in real-time. In order to validate that the eye tracking device is connected, the second step presented in the on-screen instructions requires to connect the device for the calibration process. Once connected and after the calibration process, the device is ready to receive and transmit the raw data.

First time users are required to create an account, in order for the received metrics to be stored in our database based on the user's id. The registration form that the user must fill, contains the *name, surname, gender, date of birth, telephone number, address, postal code, e-mail address, username* and *password* which must be confirmed. All of the user's personal information filled in this form are stored safely in our MySQL database. The registration process is required in order to offer users the chance to be able to connect later if one wishes to, to the Universal Workbench to view further their metrics.

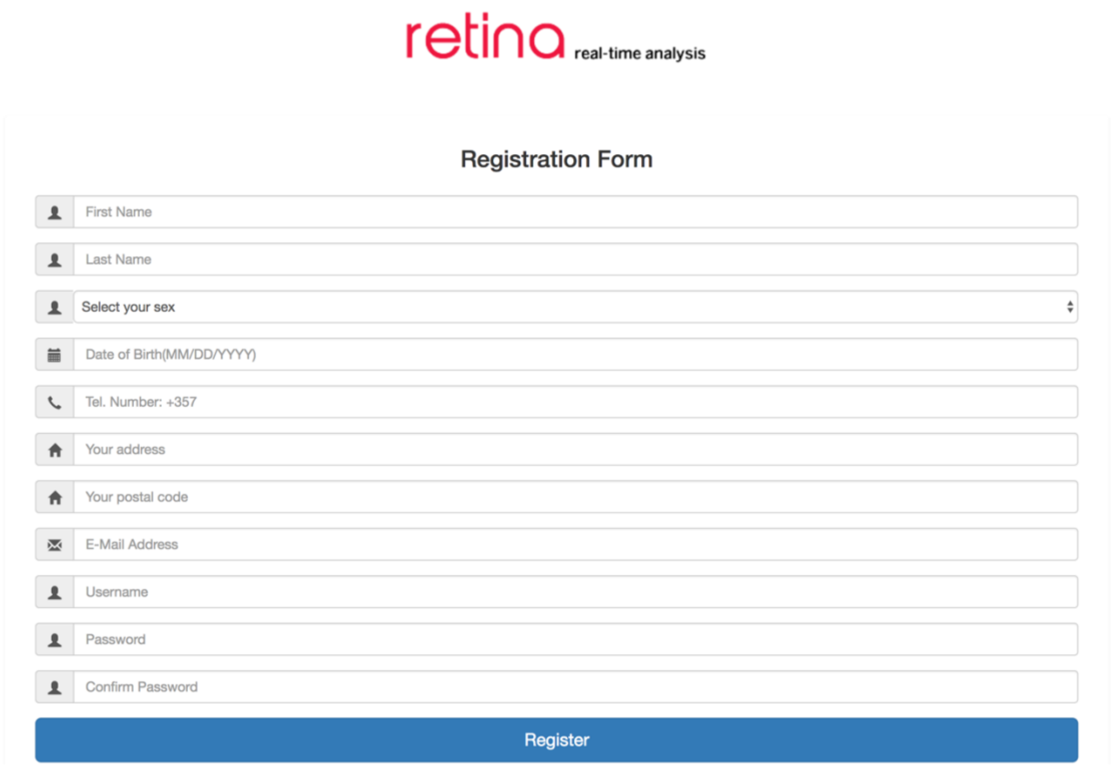


Figure 4.2 - Registration form

After successfully creating an account the user can navigate the rest of the web application by simply logging in. The form consists of the *username* and the *password* set by the user.

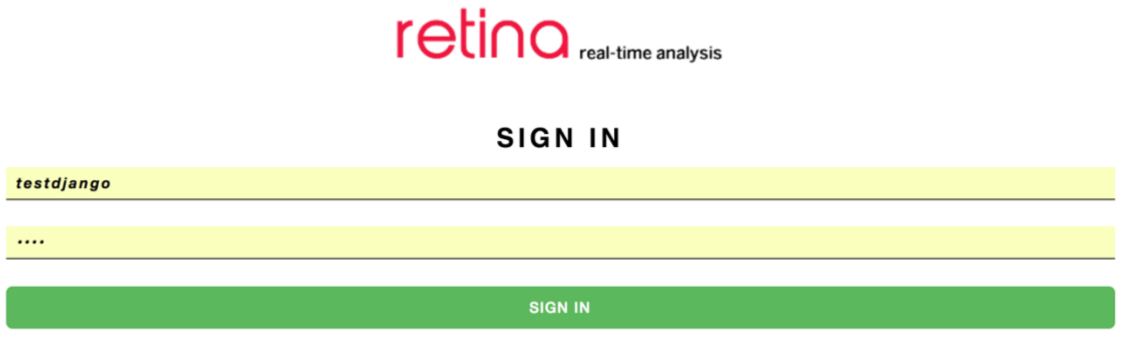
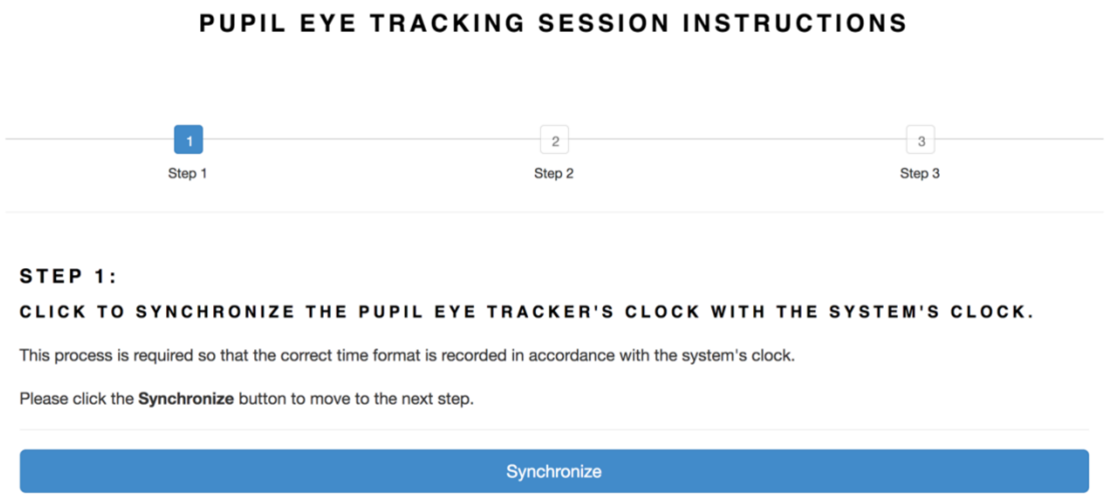


Figure 4.3 - User login form

After a user logs in, the eye tracking session instructions are shown so the user can understand why the various steps are required for the correct recording of the eye tracking data. The three required steps were mentioned and explained previously.


PUPIL EYE TRACKING SESSION INSTRUCTIONS

STEP 1:

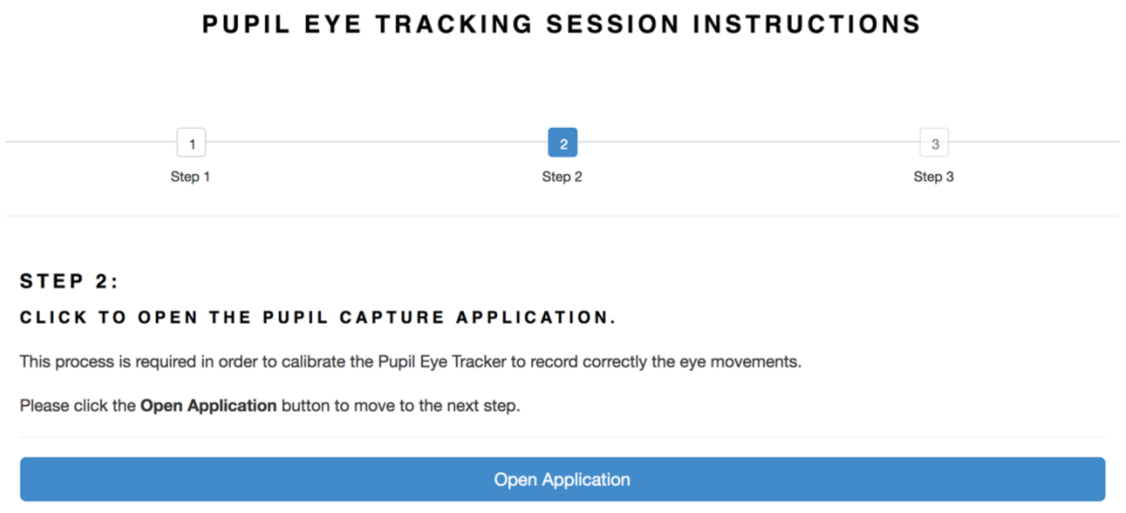
CLICK TO SYNCHRONIZE THE PUPIL EYE TRACKER'S CLOCK WITH THE SYSTEM'S CLOCK.

This process is required so that the correct time format is recorded in accordance with the system's clock.

Please click the **Synchronize** button to move to the next step.

Figure 4.4 - First step of Eye Tracking Session

The first step, as we mentioned above is required in order to synchronize the eye tracker's internal clock with the system's clock, in order to receive the timestamps of the data recorded in *UNIX* epoch format. This will help us present the data recorded in relation to the real time a fixation occurred, so the chart is visualized in a meaningful manner by correlating the correct timestamp to a fixation.



PUPIL EYE TRACKING SESSION INSTRUCTIONS

STEP 2:

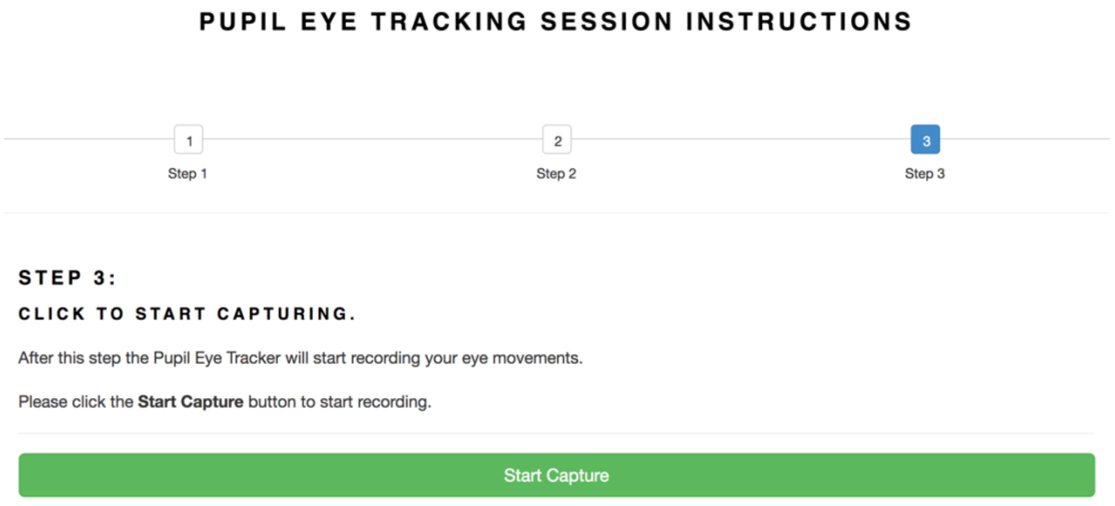
CLICK TO OPEN THE PUPIL CAPTURE APPLICATION.

This process is required in order to calibrate the Pupil Eye Tracker to record correctly the eye movements.

Please click the **Open Application** button to move to the next step.

Figure 4.5 - Second step of Eye Tracking Session

The second step of this process, as we mentioned above, is required in order to receive correct gaze data from the eye cameras. In order to be able to do that the confidence level of the pupil detection algorithm must pass the set algorithm threshold thus requiring a proper calibration so the eyes of the user are recognized. When the eyes are found by the cameras and the confidence level is high enough the calibration process starts so the eye tracking device can map correctly where the user is looking in the visual field recorded by the world camera. Furthermore, in this step the surface tracking process is also running in order to define our Area of Interest (AOI) which at a later stage will help us map the gaze information regarding fixations on the set AOI. This process requires that the Pupil Capture application is started where the calibration process and the surface tracking process occurs.


PUPIL EYE TRACKING SESSION INSTRUCTIONS

STEP 3:

CLICK TO START CAPTURING.

After this step the Pupil Eye Tracker will start recording your eye movements.

Please click the **Start Capture** button to start recording.

Figure 4.6 - Third step of Eye Tracking Session

The third step of the instructions is notifying the user that after clicking the button presented on the screen the capture of the raw data will start and the algorithms will start working in order to classify fixations and saccades, find the fixations' duration and fixation count.

By completing the steps successfully, the capture of the eye tracking data begins, the raw data are being processed by the algorithms we created, they are classified, stored and shown in real-time in charts related to fixations' duration and fixation count.

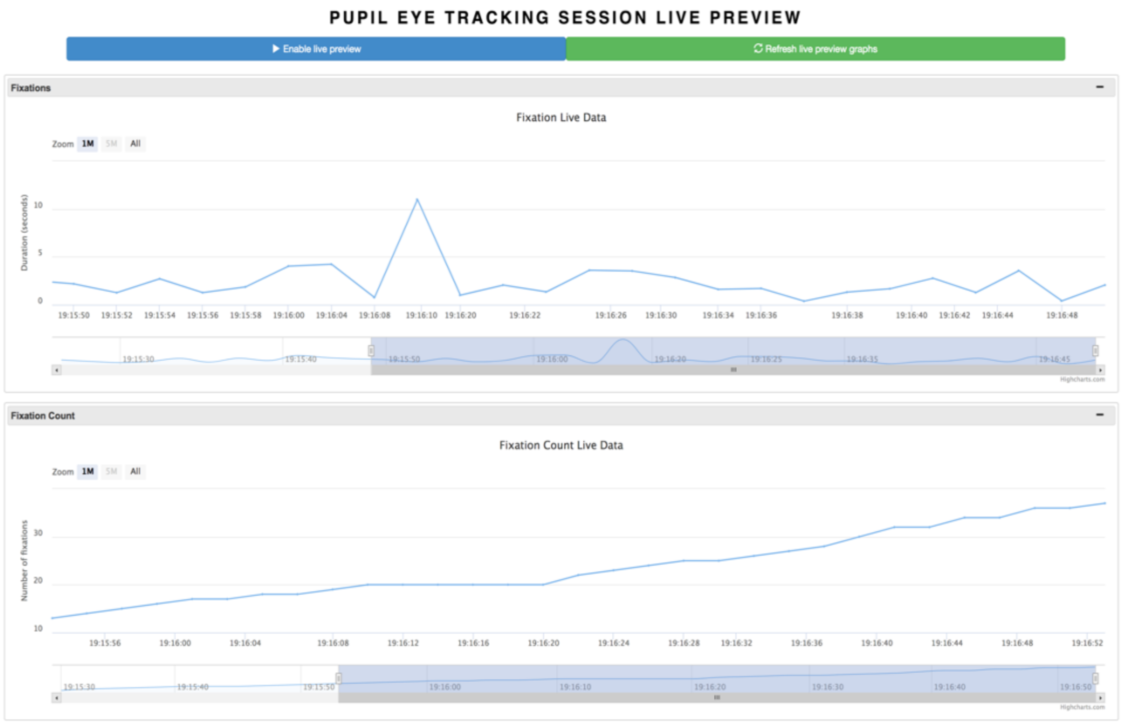


Figure 4.7 - Fixations and Fixation count live preview

When the user ends with the eye tracking session and the recording of the data is completed, a fixations gaze plot and a fixations' duration gaze plot become available which represents the visual behavior of the user in that session.

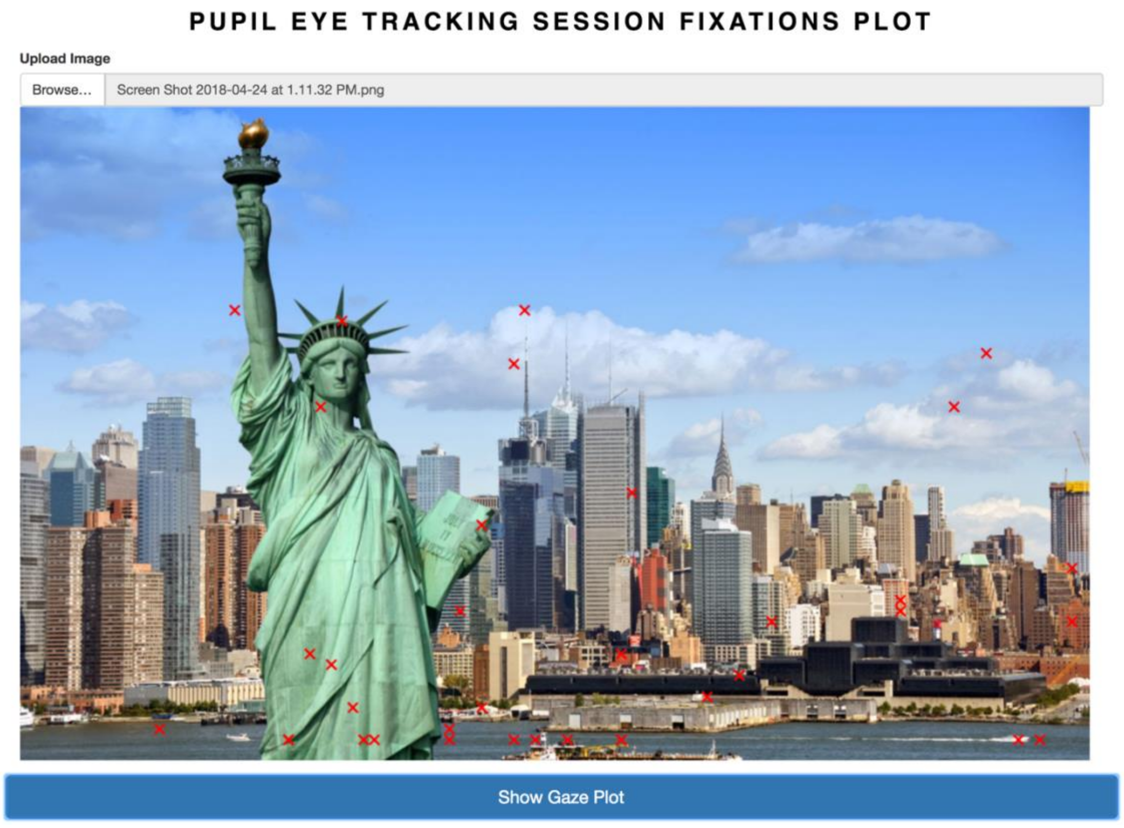

PUPIL EYE TRACKING SESSION FIXATIONS PLOT
Upload Image
Browse...
Screen Shot 2018-04-24 at 1.11.32 PM.png
Show Gaze Plot

Figure 4.8 - Fixations Gaze Plot

As we mentioned previously, each fixation's duration and the fixations of the user are mapped on a gaze plot based on the surface defined in the second step of the process required to use this real-time visualization tool. When the session of the user ends, and all the data are processed and stored, we map them on these gaze plots based on the user's visual field at the moment of recording. The fixations gaze plot maps the positions of a fixation occurred with a red ***'X',*** whereas the fixation's duration gaze plot maps the positions of a fixation occurred with a green circle based on the duration of the fixation. The longer the duration the bigger the circle.

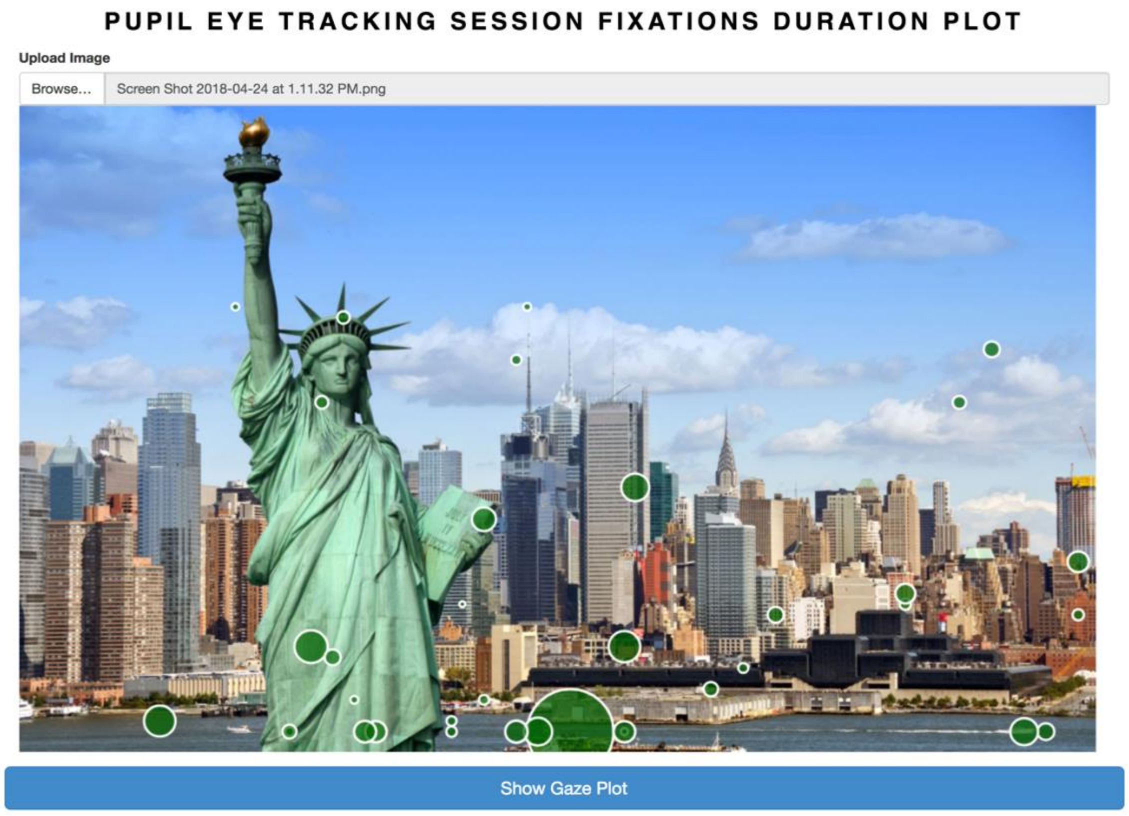
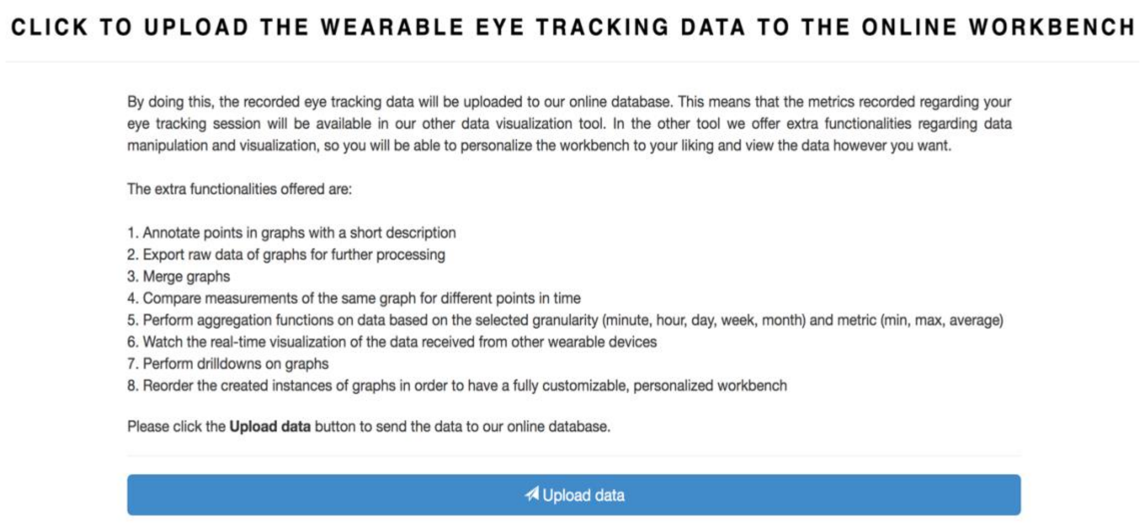

PUPIL EYE TRACKING SESSION FIXATIONS DURATION PLOT
Upload Image
Browse...
Screen Shot 2018-04-24 at 1.11.32 PM.png
Show Gaze Plot

Figure 4.9 - Fixation Duration Gaze Plot

Finally, the choice to upload the session's data to our online database is presented to the user. Then at any time the user can log in to the RETINA Universal Workbench with the same credentials and view the data regarding all of the user's sessions and visualize them again in charts; however more functionalities are offered.


CLICK TO UPLOAD THE WEARABLE EYE TRACKING DATA TO THE ONLINE WORKBENCH

By doing this, the recorded eye tracking data will be uploaded to our online database. This means that the metrics recorded regarding your eye tracking session will be available in our other data visualization tool. In the other tool we offer extra functionalities regarding data manipulation and visualization, so you will be able to personalize the workbench to your liking and view the data however you want.

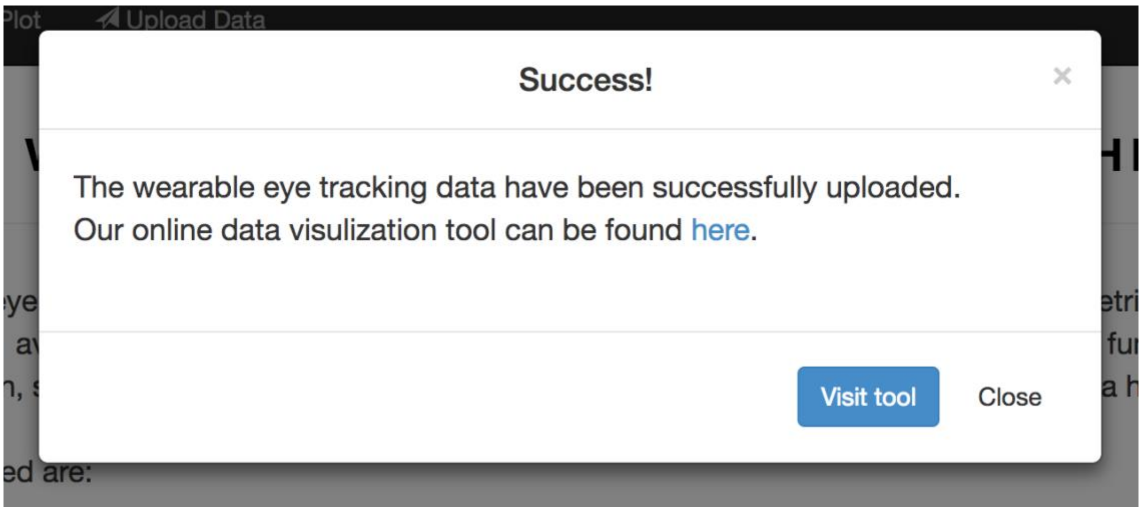
The extra functionalities offered are:

1. Annotate points in graphs with a short description
2. Export raw data of graphs for further processing
3. Merge graphs
4. Compare measurements of the same graph for different points in time
5. Perform aggregation functions on data based on the selected granularity (minute, hour, day, week, month) and metric (min, max, average)
6. Watch the real-time visualization of the data received from other wearable devices
7. Perform drilldowns on graphs
8. Reorder the created instances of graphs in order to have a fully customizable, personalized workbench

Please click the **Upload data** button to send the data to our online database.

Figure 4.10 - Upload the wearable eye tracking data to the Universal Workbench

When the upload is successful, the user is alerted and given the choice to visit the online tool in order to manipulate and visualize the eye tracking data by using the extra functionalities offered there.


Success!

The wearable eye tracking data have been successfully uploaded.
Our online data visulization tool can be found here.

Figure 4.11 - Success message

4.6 Fixations and Saccades Algorithms Design

As mentioned above, the algorithms for the detection of fixations as well as the detection of saccades already exist. We have also explained in detail how these algorithms work in previous chapters. However, what we wanted to do was change these algorithms in order to serve our purpose, which was the analysis of the eye tracking data in a real-time process.

Our system executes two algorithms; one to detect fixations and one to detect saccades. In order to be able to classify the metrics we wanted in real-time, we had to adjust the existing algorithms to fit our needs. Below I will explain the steps followed in order to receive the raw data from the eye tracking device and the pseudocode representing how the algorithms work based on the raw metrics received from the eye tracking device can be found in Appendix A.

(i) It is worth mentioning that what we were interested in was a specified area in the visual field of the user wearing the eye tracking device which was the display of the computer the user was using while interacting with the system. In order to specify that we were interested in part of the visual field of the user – called surface – we had to define it at run-time by using markers. By defining these surfaces and by making sure that the fiducial markers are recognized properly, the detected surfaces as well as the recorded gaze positions which are relative to those surfaces are broadcasted under the topic **`surfaces`**.

By taking into consideration the above, ultimately what we ended up recording based on our algorithm was the fixations on the defined surfaces.

(ii) Furthermore, in order to be able to receive the raw data from the eye tracking device in a real-time fashion, we had to study the way **`Pupil Capture`** (application responsible for reading the video streams coming from the world camera and the eye cameras and use this video streams for pupil detection, gaze tracking, surfaces detection, recording videos and stream the data in real-time) and **`Pupil Service`** (application designed to run in the background and be controlled by network commands) communicate with each other. Based on the documentation the Pupil Labs offers, these two applications communicate via a message bus internally and the networking between the applications is based on the **`ZeroMQ`** (ZMQ) network library and by using a

`PUB-SUB`, one-to-many communication socket type. Pupil Labs calls this messaging bus the `IPC Backbone,` which runs as a thread in the main process and functions as a big message relay station, where actors can push messages into it and subscribe into other actors' messages. Therefore, the `IPC Backbone` is the most important part of all communication from, to and within `Pupil Capture` and `Pupil Service`. Also, in order for us to be able to tap to the `IPC Backbone,` we need the IP address which is 127.0.0.1 (localhost) as well as the session unique port. In order to receive the session's unique port, we have to "talk" to `Pupil Remote` which serves as an entry point to the `IPC Backbone` from the fixed port `54613.` After successfully taping to the `IPC Backbone,` the requester talks to `Pupil Remote` to receive the session's unique `SUB URL` and `port`. When successfully receiving the session's port we use it to subscribe to receive the pupils' data message and in particular what we were interested in as mentioned above was the `surfaces` topic which contains the raw data regarding the fixations on the surfaces and the `pupil` topic which contains all the raw data regarding pupil positions, gaze tracking etc.

(iii) After successfully subscribing to the topic that interests us, the message received is a message with *topic* and *payload*. *Topic* is a ***utf-8*** encoded string, which is returned as a unicode object. *Payload* is a *msgpack* serialized dictionary which is returned as a Python dictionary. This dictionary contains the raw data requested from the eye tracking device based on ***"key-value"*** pairs.

It is important to state that apart from point (i) which is only taken into consideration in the algorithm for the detection of fixations, the other two points are implemented in both algorithms.

4.6.1 Fixations Detection Algorithm

As mentioned above, our algorithm for the classification of fixations differs from the algorithm used by Pupil Labs. A summary of how the algorithm works will be presented below and the pseudocode representing our algorithm can be found in Appendix A. Please note that the steps for tapping to the IPC Backbone will not be referred, as it is not part of the algorithm for fixations classification but part of the connection to the eye tracking device to receive the raw data.

Our algorithm is based on the fact that only 70 frames per second are recorded and also that the message received from the `IPC Backbone` contains the information about fixations in a raw representation. What we did was receive the whole message and if the fixation occurred on the surface that we set in the visual field of the user we stored its id, its timestamp and the normalized position (x, y) where the fixation occurred. We continued to receive messages and while the id of the fixations occurring were the same we did nothing as it meant that the new fixation occurring is part of the previous fixation. When a message received had a different fixation id that meant that the user fixated elsewhere, and the previous fixation ended. Then we calculated the duration of the fixation by subtracting the starting timestamp of the previous fixation from the timestamp of the new fixation and we stored the information regarding this fixation (duration, position, starting timestamp) alongside the user id. Furthermore, every 100 frames recorded we stored the fixation count.

4.6.2 Saccades Detection Algorithm

We have already explained an algorithm for saccade detection in our background theory. We found that algorithm quite good, straightforward and easy to implement, so we tried to implement that algorithm based on the raw data received from the Pupil Labs eye tracking device. Our algorithm differs a bit in the implementation as well as the values we set for the thresholds but the idea behind the algorithm is the same as the one described previously. Below our variation of the saccade classification algorithm will be presented and the pseudocode representing our algorithm can be found in Appendix A. Again, please note that the steps for tapping to the `IPC Backbone` will not be referred to.

Our saccade classification algorithm is based on the fact that the message received from the `IPC Backbone` contains all the information about the eye movements in a raw representation. What we did was receive the whole message and find only the information that was of interest to us. That information was the *id* of the pupil that represents from which eye the data were received, the *timestamp* representing those data, the *theta* and *phi* values or the *norm_pos* values and the *confidence* of the pupil when the data were received.

The *confidence* represents the average pupil confidence, which is an assessment of how sure we can be about this measurement. The confidence is between 0 and 1, where a value of 0 indicates no confidence and 1 perfect confidence.

The *norm_pos* values represent the (x, y) positions in the image frame in normalized positions where (0, 0) is at the bottom left of the visual field and (1, 1) at the top right of the visual field and are received only if the 3D model is not followed but the recording happens using the 2D model.

The *theta* and *phi* values are polar angles representing the 3D eye model's vector in spherical coordinates and are received only if the 3D model is followed.

For having a robust algorithm and classifying saccades successfully we chose to set our thresholds to:

duration threshold: *0.1 (100 ms)*

confidence threshold: *0.75*

velocity threshold: *3^o*

Finally, every 100 frames we store the saccade count alongside the user id and the timestamp.

4.7 Technologies and Tools used

As mentioned at the beginning of this chapter our system was implemented using the ***MVC architectural pattern***. In order to apply the MVC architecture to our system we decided to use the ***Django MVC Framework***.

(i) ***Django*** is a free and open source web application framework, written in Python which follows the MVC architectural pattern. Python is used throughout the framework and the main reason we chose to implement our system's MVC architecture using the Django Framework was that Pupil Labs also uses Python, for writing the modules responsible for manipulating the data of the eye tracking device. Based on this fact, we were able to write our algorithms also in Python and deploy them to the eye tracking device as well as deploy them on our system effortlessly as there were no compatibility issues. Furthermore, by relying our whole system on Python we were able to use Pupil Labs helper modules for the implementation of the connection to the eye tracking device through the **`IPC Backbone`** and the receiving of the raw data in real-time.

(ii) ***Python*** is the main programming language used to develop not only the MVC architecture of our system but our algorithms as well. It is an interpreted high-level programming language for general-purpose programming. It provides constructs that enable clear programming even on large-scale applications. In our system it is widely used to create the models that represent the data of our database, it is used to implement the functionalities of the controller and it is even used in the views presented to the user. Furthermore, it is used to create the structure of the web application and declare the settings used by the system after its creation.

(iii) ***JavaScript*** is a dynamic programming language, which can be embedded in HTML pages, enabling interactive web pages and thus is an essential part in the development of web applications. The vast majority of web applications use it and all major web browsers have a dedicated JavaScript engine to execute it. Our whole system depends on the execution of JavaScript functions not only to provide a nicer user interface but also for implementing key functionalities in the visualization of our gaze plots.

(iv) ***Cascading Style Sheet (CSS)*** is a style sheet language used for describing the presentation of a document written in in a markup language. It is commonly used for styling the user interfaces in HTML. CSS adjusts different styles and methods in the same page and can display or resize the screen depending on the device. In our system CSS complements the JavaScript.

(v) ***HyperText Markup Language (HTML)*** is the basic markup language used to create the user interfaces of a web application. It provides a means to design composition documents by structural semantics.

(vi) ***MySQL Workbench*** is a visual database management tool that integrates SQL development, administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system. In order to use this tool our database is implemented using MySQL, which is an open-source relational database management system. It is used to store and retrieve our data over the network in order to be represented by our models and view them to the users in a graphical, meaningful way. It serves as a link between our system's back-end and the front-end. The system plays a dominant role in allowing the information received from the database to flow from one component to the other with the ultimate purpose of viewing the data to the user. An important aspect is that the database keeps the data secure and safe by storing them in tables. The tables include the users' personal information as well as their data regarding eye tracking sessions based on their unique ids. Therefore, whenever a user logs into the system only the appropriate data will be shown.

(vii) ***Sublime Text*** is the main editor used to write our code. It is a proprietary cross-platform source code editor with a Python API. It natively supports many programming languages and markup languages, and functions can be added easily with the use of plugins.

(viii) ***Highcharts*** is an open-source library used in the development of web applications in order to set up interactive charts in the web pages to present data in a more meaningful way.

4.8 Database Architecture

Our database was implemented using MySQL. MySQL is an open-source relational database management system based on SQL (Structured Query Language). It is used to store and retrieve our data over the network in order to be represented by our models and view them to the users in a graphical, meaningful way. It serves as a link between our system's back-end and the front-end. An important aspect is that the database keeps the data secure and safe by storing them in tables. The tables include the users' personal information specifically the users' IDs as well as their data regarding eye tracking sessions based on their unique IDs. Therefore, whenever a user logs into the system only the appropriate data will be shown.

Below the architecture of the database will be presented. As we can see by the database diagram, we only have two tables in order to store the data regarding the processed eye tracking data. The one table stores the factors of the system which are represented by an

ID and a ***name.*** The second table stores the values regarding those factors which are represented by a ***unique ID***, the ***user ID***, the ***factor ID***, the ***factor value***, the ***date recorded***, the ***x-position*** a fixation occurred on a surface and the ***y-position*** a fixation occurred on a surface. The relationship between the tables is 1-N based on the ***factor ID.*** The ***user ID*** is stored based on the current user that is logged in. Since our system is part of a larger data visualization and manipulation tool which is the RETINA Universal Workbench, it is required for this database to contain information regarding the users which are stored in our online database. Specifically, when a user logs in, the username and password given are authenticated based on the data stored in the online database. What is returned is the user's ID and username in order to keep track of a user's session. By choosing to implement this architecture so the two systems have a common table which stores, retrieves and authenticates information regarding a user, we give the advantage to the user to be able to use the two systems with the same credentials, since this system is part of the RETINA Universal Workbench visualization tool.

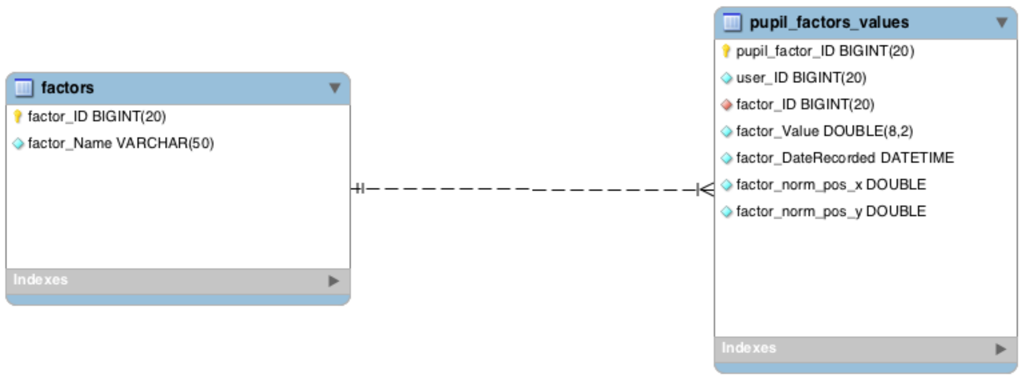


Figure 4.12 - RETINA System Database Architecture

Chapter 5

Design and Development of RETINA Universal Workbench

This chapter will talk about the extended tool that was developed, called ***RETINA Universal Workbench***, in order to work as a universal platform to preview the different metrics recorded during eye tracking sessions from the RETINA system we discussed in Chapter 4 as well as visualizing metrics received from other wearable smart devices regarding the analysis of physiological sensory data of users. The architecture of the system will be discussed as well as the extra functionalities offered in comparison with the RETINA web application we presented in Chapter 4. Furthermore, I will present how the two systems communicate and how the integration was made in order to store the analyzed wearable eye tracking data to our online database. Finally, the system design, the flow of information and the architecture of the database will be explained as well as the tools and technologies used for the development of this tool.

5.1 RETINA Universal Workbench Web Application

The functionalities that will be described later as well as the design of the system including the user interfaces and how the information flows while interacting with this web application are part of the ***RETINA Universal Workbench*** web application which was also implemented in the scope of my Bachelor Thesis. This tool is considered to be a fundamental tool for my Thesis as it is a universal tool which extends the architecture of the RETINA system we described in Chapter 4, however, offering more functionalities for data manipulation and visualization.

By extending RETINA, we showed that it is possible to be used in different applications, as we do use it, by combining the wearable eye tracking data and the data regarding the physiological state of a user. ***RETINA Universal Workbench,*** the extension of RETINA is the core component of our complete system as it serves as the end-point to store the data received from the eye tracking device as well as the wearable smart devices.

The process which receives, stores and sends the wearable eye tracking data has been described in Chapter 4.

In order to receive, store and send the data received from the wearable smart devices regarding the physiological state of a user, the metrics are recorded through the sensors of wearable smart devices. These metrics are sent to an Android application which implements state-of-the-art algorithms regarding the extraction of useful results relevant to the emotional state of the user.

Both the data extracted from the Android application and the RETINA system are sent to web services implemented in a controller of the Universal Workbench in order to store them in our online database.

5.2 System Architecture

Again, for developing this tool, we decided that the best path to follow was to implement it as a web application, in order to be platform independent so anyone with access to a web browser and the Internet could use our tool and interact with it as easily as possible. For this reason, following the development process of the RETINA system we talked about in the previous chapter, the architecture we chose to use for our tool was the MVC architectural pattern. The technologies and tools used to implement the system will be described at a later stage.

The benefits of using the MVC architectural pattern were discussed in the previous chapter so we will not analyze them again.

5.3 Integration of Universal Workbench and RETINA – Extended System Architecture

As we have already mentioned in the previous chapter and in this chapter as well, RETINA Universal Workbench is an extension of the RETINA system presented in Chapter 4. By extending our architecture to include both the RETINA system which analyzes, visualizes and stores wearable eye tracking data as well as include the system which analyzes, visualizes and stores wearable data regarding the physiological state of a user we managed to create an extensible web application tool where different metrics regarding different factors can be presented to the user.

By creating a universal platform for the visualization of different factors, regarding different aspects of the daily lives of individuals, such as their visual behavior while interacting with a system or the physiological analysis of users in their daily life, we can extract meaningful conclusions in real-time regarding their user experience while using different technologies every day.

The integration of the systems we presented was succeeded by using web services implemented in a controller of the Universal Workbench web application. This controller serves as a communication mediator between RETINA which is responsible for sending the processed data regarding the wearable eye tracking data of users and the Android application which is responsible for sending the processed data regarding the physiological state of users to the Universal Workbench database.

These web services were developed in order to be accessed by methods implemented in each of the system which communicates with the Universal Workbench controller. They are always available and when a system wishes to contact the appropriate web service a request is made from the method of each system.

Below the extended system architecture of the RETINA Universal Workbench system will be presented as well as the integration between the systems alongside their respective technologies.

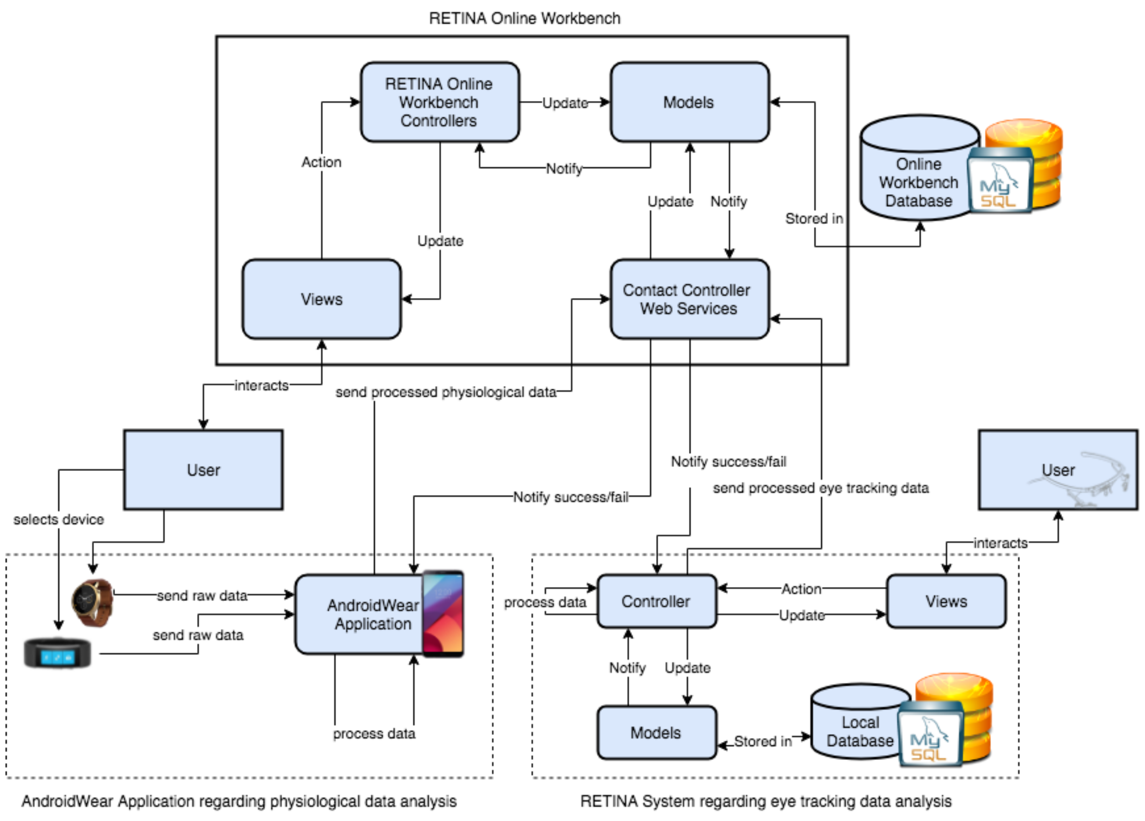


Figure 5.1 - Extended System Architecture and integration of systems

5.4 System Requirements

As discussed in the previous chapter, in order to define our system's requirements, we had to split them into *functional* and *non-functional* requirements.

5.4.1 Functional Requirements

As stated above, functional requirements express the way a system must work, the functionalities supported by this system and offered to the end-user.

Some of these functional requirements are:

- An individual must be able to register to the universal workbench.
- An individual must be able to login to the universal workbench after successfully creating an account.

- An individual must be able to access the personalized *Workbench* in order to view the already created charts regarding specific metrics.
- An individual must be able to create new instances of charts regarding metrics through the workbench.
- An individual must be able to access the *Comparisons* interface in order to view already created charts containing comparison information of a specific metric for different periods of time.
- An individual must be able to perform all the functionalities offered by the tool on the created charts and their metrics. These functionalities are as follows:
    - View different measurements in charts.
    - Annotate points in charts with a short description.
    - Export raw data of charts for further processing.
    - Merge charts.
    - Compare measurements of same chart for different points in time.
    - Perform aggregation functions on data based on the selected granularity (minute, hour, day, week, month) and metric (minimum, maximum, average).
    - Watch the real-time visualization of metrics received from wearable devices.
    - Perform drilldowns on data of charts.
    - Delete an instance of an already created chart.
    - Reorder the already created charts in order to have a fully customizable, personalized workbench.
- An individual must be able to change his/her personal information through an easily accessible form.

5.4.2 Non-functional Requirements

As stated above, non-functional requirements specify criteria that can be used to judge the operation of a system as well as state constraints on the design or implementation of the system.

In the previous chapter we mentioned that the system would communicate with a universal visualization tool for further processing the data recorded from the wearable eye tracker and other wearable devices. Since the design of the universal tool we are talking about in this chapter still requires a real-time analysis and visualization of the data, our universal tool had to ensure some basic non-functional requirements, in terms of performance, efficiency, reliability and extensibility since the core of this tool is based on the tool described in the previous chapter.

Some of these non-functional requirements are:

- *Performance:* The tool has to offer stable performance and complete the assigned work in short periods of time in order to maintain the real-time aspect of the system we wanted to implement without taking into consideration the workload assigned.
- *Extensibility:* The tool has to be extensible, meaning that new features can be added easily with minimal effort from the developers' point of view.
- *Availability:* Since the system is developed as a web application, it must be available to the end-user at any given time and be committed to complete the work assigned.
- *Security:* The tool has to be secure from either cyber or physical attacks from people who potentially want to harm the system. Maximum security for the personal data of the users has to be ensured.

5.5 Flow of Information – System Design

In order for an individual to be able to use our universal workbench tool called ***RETINA Universal Workbench***, a supported device has to be acquired (either wearable or eye tracking), in order for the tool to be able to visualize the recorded metrics from these devices. As we mentioned at the beginning of this chapter, this version of the tool is an extended implementation of RETINA, the real-time analysis and visualization tool we presented in Chapter 4. The Universal Workbench works as a universal platform, offering more functionalities and also supports other wearable smart devices as well. Such devices are the Motorola's *Moto 360* and the Microsoft's *Microsoft Band* which are used to receive sensory data regarding the physiological state of a user. The process required in order for a user to be able to interact with the system is that an account has to be created so the user will be able to login to the system.

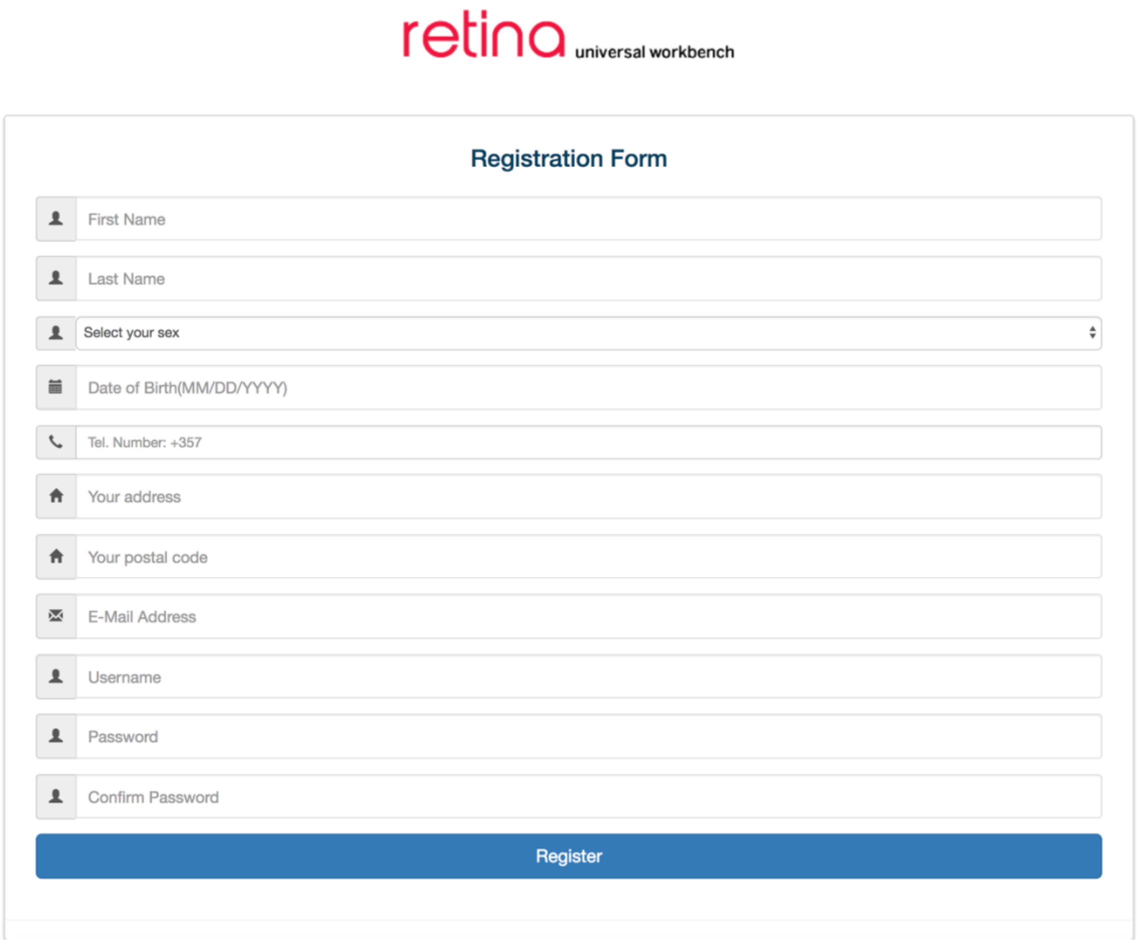


Figure 5.2 – RETINA Universal Workbench Registration Form

After logging in, the user is redirected to the personalized workbench where there the charts containing the different metrics regarding different factors will be presented.

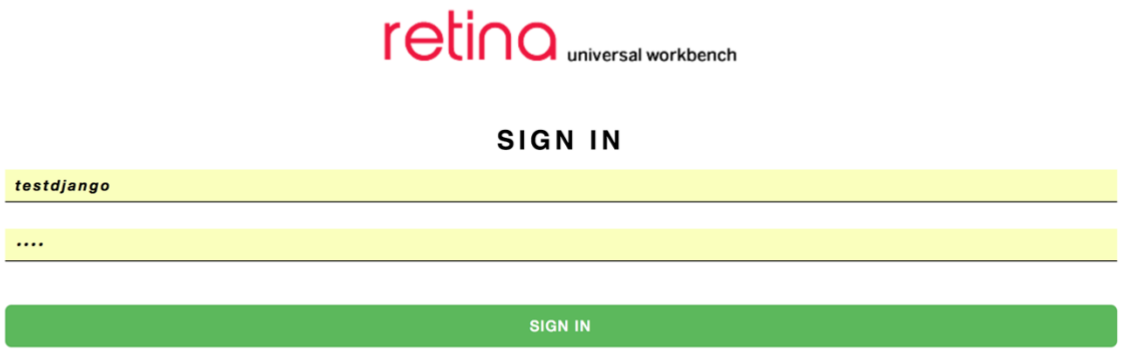


Figure 5.3 – RETINA Universal Workbench Login form

If a user hasn’t used the system before, nothing will be visualized in the charts as data regarding the user’s emotional state and visual behavior do not exist.

The factors we are able to visualize in charts, so the user can extract meaningful information regarding the metrics shown are split in two categories.

The first category represents the ***visual behavior*** of the user and the metrics received in a session by using the system we described in the previous chapter where the user has the ability to send those metrics to this universal system. Those metrics are the ***fixations*** and the ***fixation count*** of a user during all of his eye tracking sessions. The calculation of those metrics, are based on the algorithms we described in the previous chapter.

The second category represents the ***emotional state*** of the user and the metrics received while wearing the wearable devices we mentioned above. Those metrics are the ***Heart Rate***, ***RR Interval***, three different algorithms for the ***Stress*** calculation, the ***Physical activity*** of the user as well as the ***Accelerometer (x, y, z)*** metrics received from the wearable device regarding the hand movement of the user while wearing the device. The calculation of those metrics, are based on already implemented algorithms from previous thesis.

Assuming that a user has successfully logged in and there are data to be visualized in charts regarding the visual behavior as well as the emotional state, the user can choose to visualize a factor from the ones we mentioned above.

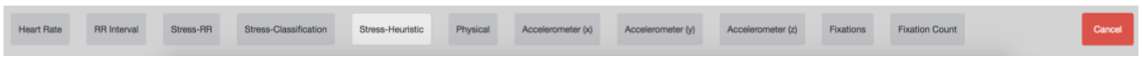


Figure 5.4 - Available factors

After choosing to visualize a factor the metrics will be visualized in a chart where the x-Axis represents the time and the y-Axis represents the proper value for each type of chart (e.g. for the Heart Rate chart the y-Axis value is a number whereas for the Physical activity chart the y-Axis is represented in a state; sitting, standing, walking, running). Furthermore, in some charts there are set thresholds in order for a user to be able to understand the meaning of the values regarding the chart (e.g. for the RR Interval and the three types of stress visualization charts the thresholds are Relaxed, Neutral and Very Stressed which are set to certain values by the administrator of the system).

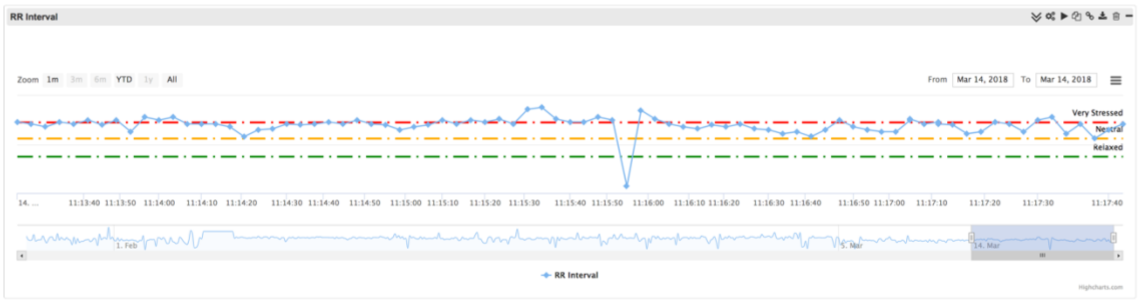
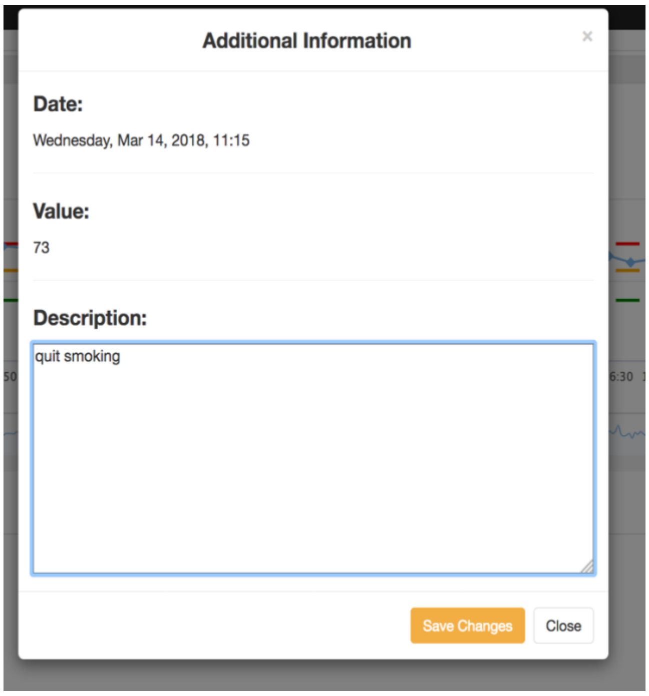


Figure 5.5 - RR Interval chart with the set thresholds

After successfully visualizing a chart the user can perform the functionalities we mentioned above in the functional requirements regarding the data manipulation and visualization.

(i) The user can choose to annotate a point with a short description by clicking on it and a window will open in order to write the description.



Additional Information

Date:

Wednesday, Mar 14, 2018, 11:15

Value:

73

Description:

quit smoking

Figure 5.6 - Annotate a point with a short description

(ii) The user can perform drilldown functions on the data of one chart. By selecting a period of time, the monthly average of the metric will be viewed in a bar chart and the user has the choice to drill down into a certain month and view the weekly average of the selected month again in a bar chart.

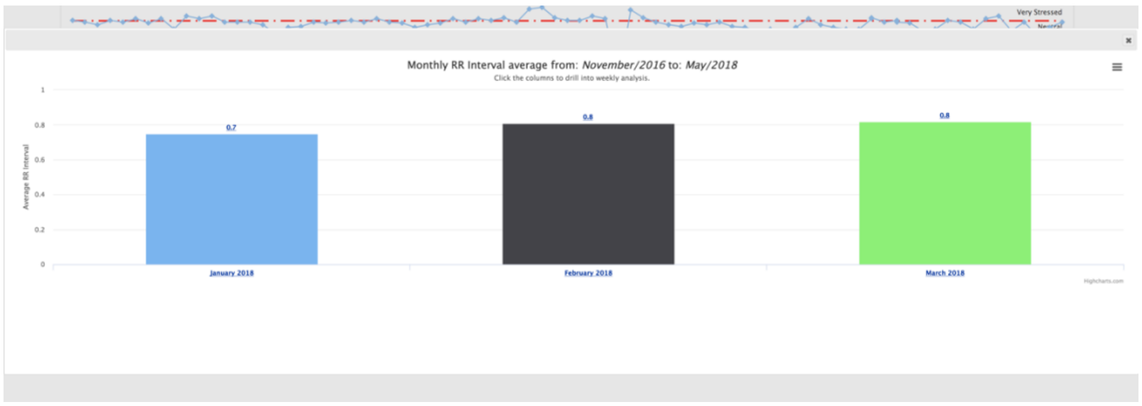


Figure 5.7 - Monthly Drilldown of RR Interval data

(iii) The user can perform aggregation functions on the data by selecting the granularity value (minute, hour, day, week, month) and the metric (minimum, maximum, average). This will result in changing the selected chart and visualize it based on the user's preferences (e.g. If a user chooses ***daily*** and ***maximum*** for the ***Heart Rate*** chart). The user has the ability to reset the chart.

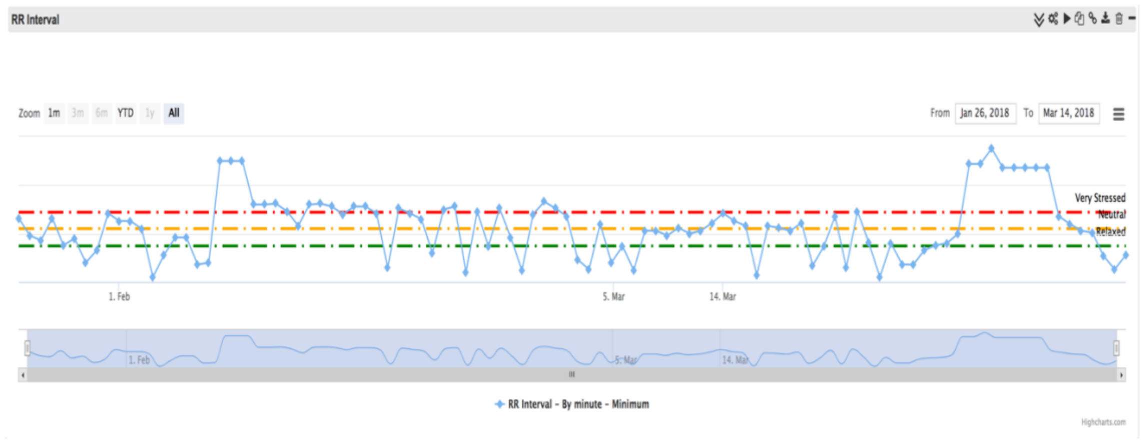


Figure 5.8 - Aggregation of RR Interval data by minute and minimum value

(iv) If a user wears a wearable device and new metrics are recorded the functionality of the real-time visualization of the newly recorded metrics is given to the user.

(v) The user can perform comparisons between data of the same chart for different points in time. By selecting this functionality, the user can select a point and choose n points before the selected point in order to compare with the n points after the selected point. After the selection is made a comparison chart is created in the Comparison tab of the web application where the user is redirected in order to view it.

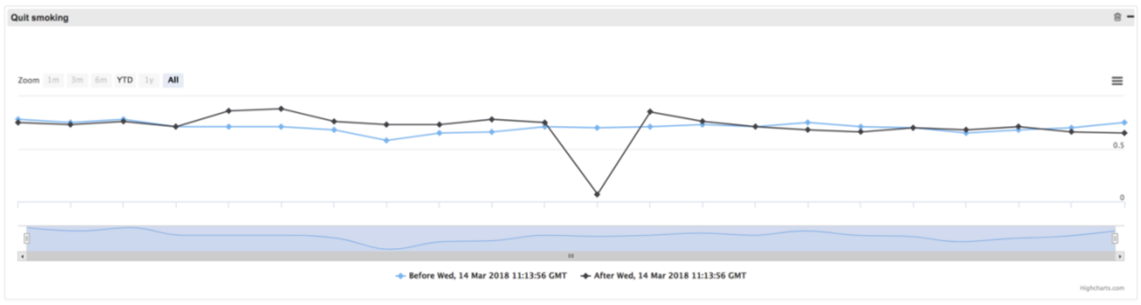


Figure 5.9 - Comparison of 22 points for RR Interval data

(vi) The user has the option to merge different charts into one and view the different measurements overlapping one another. The user has the ability to unmerge the charts if he no longer wishes to have them merged.

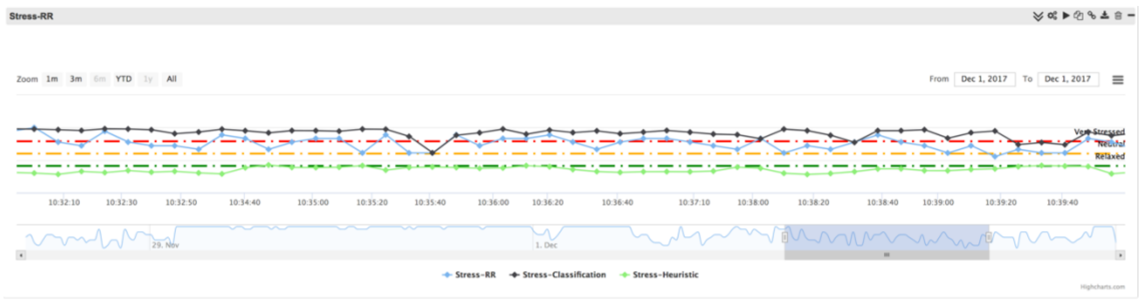
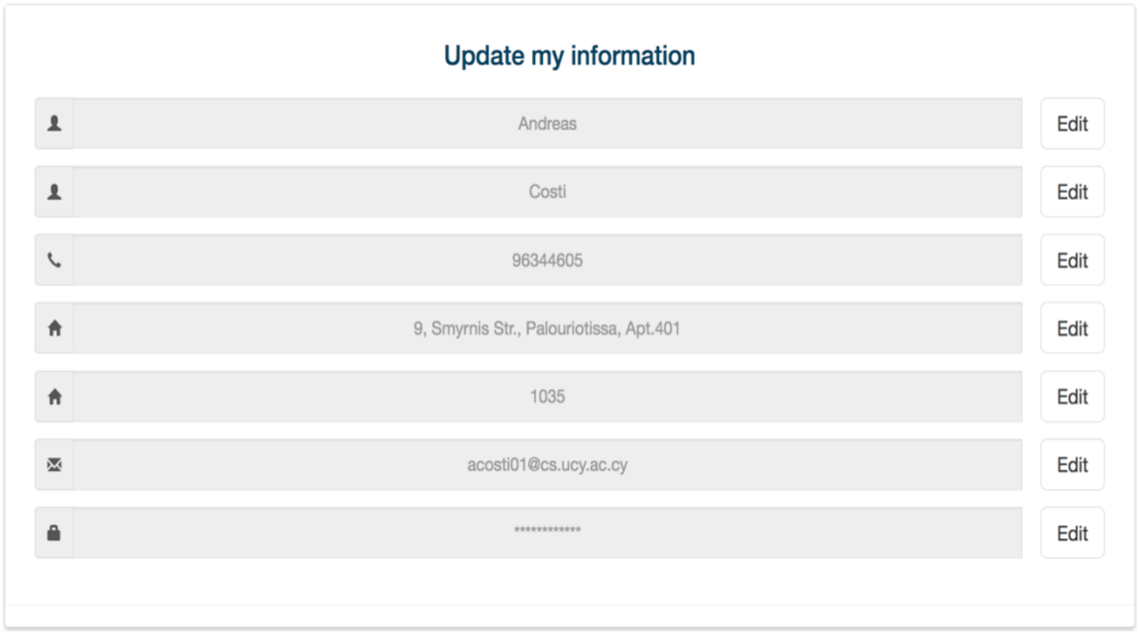


Figure 5.10 - Merged charts regarding the 3 types of Stress algorithms

(vii) The user has the ability to export the raw data regarding a chart into a csv file for further processing offline.

(viii) Lastly, the user has the ability to delete an instance of a chart where a confirmation dialog will be shown in order to verify that the user does indeed want to delete the selected instance.

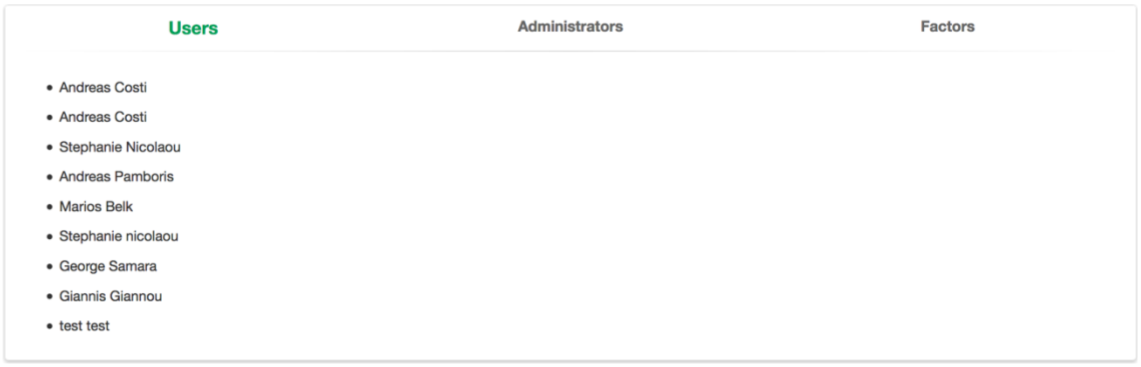
Other functionalities offered by this system is that the user has the ability by clicking on the Profile section to change the personal information regarding the name, surname, telephone number, home address and postal code, email and password. If a user wishes to change the password, the old password and the new password has to be given, as well as confirming the new password.


Update my information

👤	Andreas	Edit
👤	Costi	Edit
📞	96344605	Edit
🏠	9, Smyrnis Str., Palouriotissa, Apt.401	Edit
🏠	1035	Edit
✉	acosti01@cs.ucy.ac.cy	Edit
🔒	************	Edit

Figure 5.11 - Update personal information

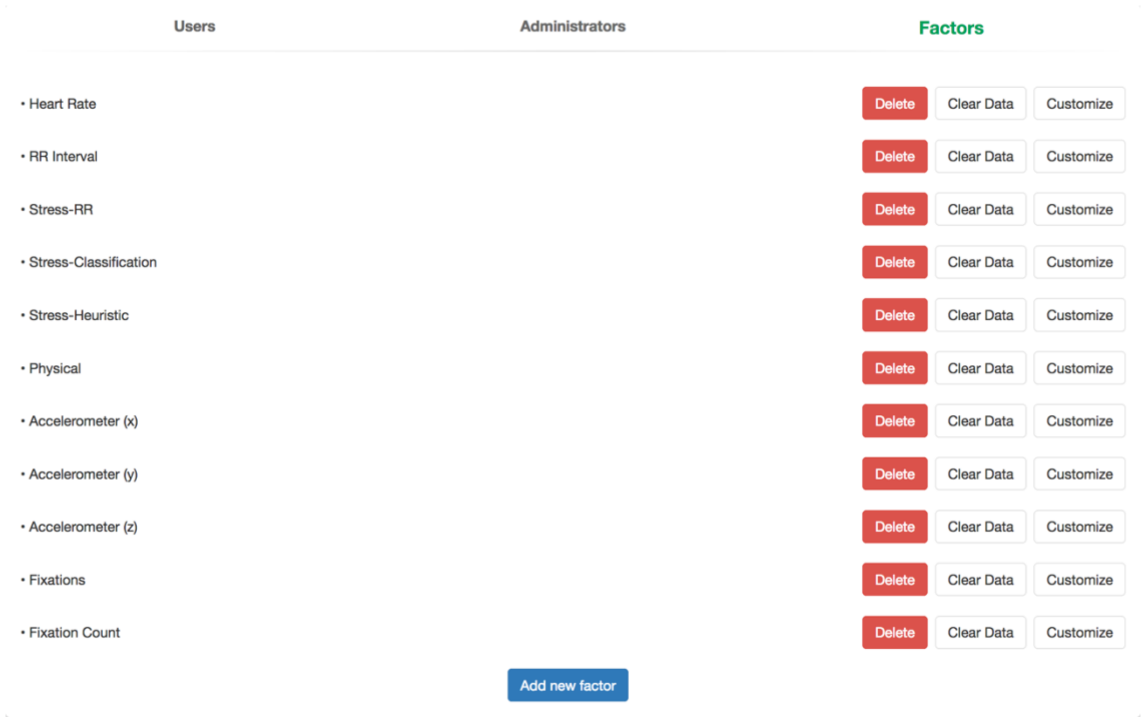
Finally, a user who is also an administrator of the system has the ability from the Website Settings to view the users and the administrators of the system, as well as view the factors supported by the system which the users can visualize.



- Andreas Costi
- Andreas Costi
- Stephanie Nicolaou
- Andreas Pamboris
- Marios Belk
- Stephanie nicolaou
- George Samara
- Giannis Giannou
- test test

Figure 5.12 - Administrative view of Web Application's Settings

The administrator has the ability to extend the system and add a new factor to visualize its metrics. Furthermore, the administrator has the ability to clear the data of a factor for all the users or delete one factor from the system. Lastly, the administrator has the ability to customize the threshold values of each chart.



- Heart Rate
- RR Interval
- Stress-RR
- Stress-Classification
- Stress-Heuristic
- Physical
- Accelerometer (x)
- Accelerometer (y)
- Accelerometer (z)
- Fixations
- Fixation Count

Figure 5.13 - Factors settings (add, delete, clear, customize)

5.6 Technologies and Tools used

As mentioned at the beginning of this chapter our system was implemented using the MVC architectural pattern. In order to apply the MVC architecture to our system we decided to use the ***ASP.NET Core MVC*** offered in ***Microsoft Visual Studio 2017***. In this section we will describe only the different technologies used in this system as most of the technologies and the tools used such as ***JavaScript***, ***CSS***, ***HTML***, ***MySQL*** and ***Highcharts*** are the same with the system we described in the previous chapter.

(i) The ***ASP.NET Core MVC*** is a web application framework developed by Microsoft, which follows the MVC architectural pattern. It is written in C# and VB.NET and the controller methods are developed using the C# programming language. The views of the web application are developed using

CSHTML rather than plain HTML in order to offer the developers the ability to include C# commands in the HTML file.

(ii) ***Microsoft Visual Studio*** is an integrated set of development tools for building ASP.NET Web applications. It supports several programming languages natively such as C# and C++ and gives the opportunities to the developers to download and integrate support for more languages. It can be used to develop web-sites, web applications, web services, mobile applications and HoloLens native applications. Finally, it gives the opportunity natively to manage a Microsoft SQL Server but with the help again of packages, support for other types of Database systems can be integrated. In our case we downloaded the ***MySQL connector for Visual Studio 2017***.

(iii) ***C#*** is an object-oriented programming language, developed by Microsoft. C# is the programming language used in the development of the MVC applications using the Visual Studio. By using the C# programming language, the implementation of a web application is made easy because no extra packages or external libraries are required in order for the correct communication between Models, Views and Controllers.

5.7 Database Architecture

The database of RETINA Universal Workbench was implemented using MySQL as well. We described MySQL previously when we talked about the database architecture of RETINA.

RETINA Universal Workbench is the extended system, as we have already mentioned, which can contain information regarding eye tracking data as well as data regarding the emotional state of a user. However, more importantly, in this database we store the personal information of the users when they create a new account.

Universal Workbench's database consists of seven tables:

(i) The ***users*** table, which contains information regarding the users' personal information such as name, surname, date of birth, gender, telephone number, address, postal code, email, username, password and if a user has administrative privileges.

(ii) The ***factors*** table, which contains information regarding the factors supported by our system in relation to the eye tracking data to be stored and the emotional data of the users. This table contains the factor name and the thresholds set for each factor by the administrator. Such factors are fixations, heart rate, etc.

(iii) The ***useremotions*** table, which contains information regarding the data recorded for each user in relation to their eye tracking sessions as well as their emotional data recorded from the wearable devices. This table contains the user id and the factor id which are foreign keys from the tables users and factors respectively with a 1-N relation. It also contains the value recorded as well as the date recorded of each value.

(iv) The ***saved_reports*** table, which contains information about the created charts of the users on their workbench based on their personalization preferences, in order to be able to view them whenever they login without having to recreate them. This table contains the user id and the factor id which are foreign keys from the tables users and factors respectively with a 1-N relation. It also contains the report priority which represents the order in which the charts will be presented, the report name which essentially is the factor's name, the report's granularity and the report's metric. If granularity and metric are not selected by the user from the choice to aggregate the data of a chart, the values default to ***'Normal'.***

(v) The ***mergedgraphs*** table, which contains information regarding the charts which a user merged. This table contains the report id which is a foreign key from the saved_reports table with a 1-N relation and the chart id which is a foreign key from the factors table with a 1-N relation. Both of these fields create the primary key of the table.

(vi) The ***saved_comparisons*** table, which contains information regarding the comparison charts created by a user for a factor. This table contains the user id and the factor id which are foreign keys from the users table and the factors table respectively with a 1-N relation. It also contains the start date and the end date of the comparison chart as well as the chart title which will be the description given to the first selected point during the comparison process or the factor name if no description is available.

(vii) The ***factorsvaluedescriptions*** table, which contains information regarding the descriptions given when a user annotates a point. It contains the factor id, the user id and the report id which are foreign keys from the saved_reports table. The relations between the fields mentioned are all 1-N. It also contains the description and the date recorded of a point.

Based on the analysis presented and how the database was designed, whenever a user logs into the system only the appropriate data will be shown. Below the architecture of the database will be presented. The users table is the table used by RETINA as well for registering and authenticating the users.

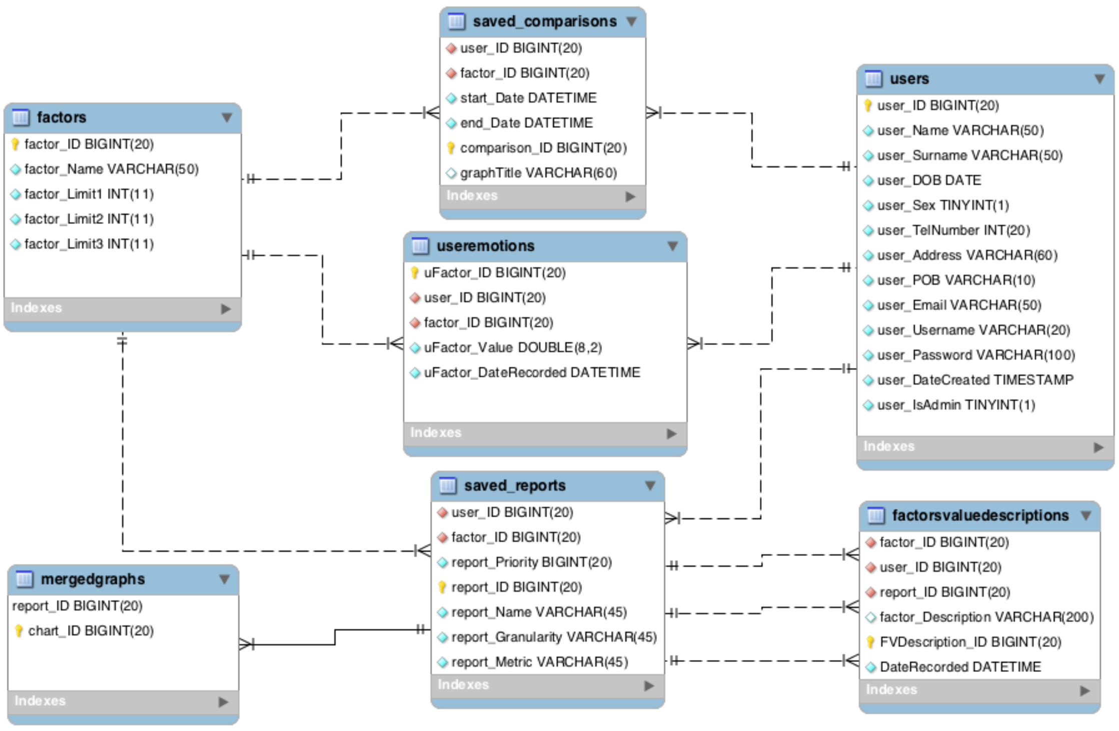


Figure 5.14 – RETINA Universal Workbench System Database Architecture

Chapter 6

Evaluation

This chapter will talk about how we evaluated our Universal Workbench system, for the visualization of metrics regarding the analysis of wearable eye tracking data as well as data regarding the physiological state of users and in particular their emotions. Twenty people participated for the system's evaluation following a scenario we presented. Our aim was to present the users with a scenario where they had to interact with our system executing the tasks presented and after finishing all those tasks answer two questionnaires in order to approve the usability, design and functionality of our system as well as how they perceive the aesthetics of our Universal Workbench.

6.1 Scenario of Evaluation

In order to be able to apply an evaluation process to our Universal Workbench system, we had to present the user with a real-world scenario of interaction with this system.
Firstly, we explained to the users the purpose of RETINA Universal Workbench and how it can be used to visualize data regarding different aspects of the users' daily lives, such as their visual behavior received from eye tracking devices or the analysis of physiological data regarding their emotional state received from wearable smart devices.
Then, we explained how the two main sub-systems which comprise the RETINA Universal Workbench work and how they interact with the Universal Workbench.
Finally, we presented the available functionalities offered by the Universal Workbench regarding data manipulation and visualization.

After explaining the technical aspects of this system, we presented the users with the following scenario in order for them to understand their role while interacting with the system and what they should have in mind during the interaction process. We told the users that in order to evaluate the system they should consider that they used previously a wearable smart device from which physiological metrics were received regarding their emotional state, in real-time, and they were stored in our online database. After using the wearable device, they wished to access the RETINA Universal Workbench in order to see how these metrics are visualized and presented to them with the use of charts.
The interaction between the user and the system comprises of six steps; each step with its corresponding tasks.

(i) Firstly, the user had to access the system, by; (i) visiting the URL where the RETINA Universal Workbench is hosted and (ii) access the system with the credentials we equipped them.

(ii) Secondly, the user had to create a chart and visualize the data by; (i) creating a chart by pressing the green **'+'** button which presents the supported factors that can be visualized in charts; (ii) selecting the Heart Rate choice, which would create an instance of a chart visualizing the metrics related to the Heart Rate during the use of the wearable device and (iii) from within the chart created and by using the Zoom functionality which is located at the top left corner of the chart, press the ***"All"*** button which will visualize all the data regarding the Heart Rate metrics recorded.

(iii) Furthermore, the user had to assess some basic functionalities of the system by; (i) selecting a point of the chart that represents a value of the Heart Rate and annotate the selected point with a short description; (ii) selecting the Drill-down button which is located at the top-right part of the chart and define the dates required **(From: 07/01/2017, To: 05/25/2018),** in order for the data to be presented with the use of bar charts that will represent the monthly average Heart Rate from July 2017 until today and (iii) navigate through the chart by pressing a bar which will drill into the month and show in detail the weekly average heart rate of the selected month.

(iv) Additionally, the user had to use the aggregation functionality offered, by; (i) selecting the aggregation button which is located at the top-right part of the chart and define the Granularity to **"By minute",** the Metric to **"Maximum"** and by pressing the "**Update Graph**" button aggregate the metrics of the Heart Rate to visualize their by-minute maximum Heart Rate value recorded from the wearable device and (ii) press the aggregation button again and select the **"Reset to raw metrics"** button which will reset the chart to the real values recorded.

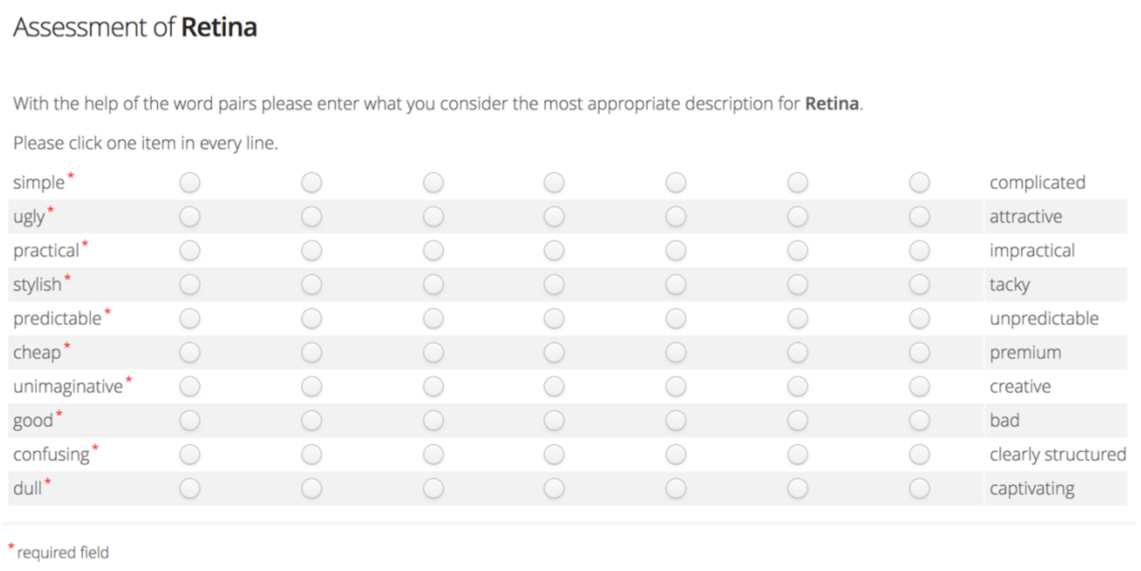
(v) In addition, the user had to use the merging functionality offered, by; (i) selecting the merge button which is located at the top-right part of the chart, tick the "Stress-RR" choice and press submit in order to merge the chart of Heart Rate with the chart of Stress-RR and visualize their metrics overlaid.

(vi) Concluding, the user had to: (i) create a chart of the "Physical" factor which represents metrics relevant to the physical movement of the user while using the wearable device and the metrics are explained by using four states (Sitting, Standing, Walking, Running); (ii) Drag-and-drop the Heart Rate chart to change its position with the Physical chart in order to customize the workbench; (iii) delete the instance of the Physical chart by clicking the bin icon located at the top-right part of the chart and confirming this choice and (iv) Logout from the system.

6.2 RETINA Universal Workbench Evaluation Process

This experiment was carried out with the objective to decide how satisfied were the users while interacting with our Universal Workbench following the scenario presented. In order to measure the usability, the attractiveness as well as how the users perceived the aesthetics of our interactive Universal Workbench, we decided to use two questionnaires. The first questionnaire was AttrakDiff [33], which can be used to understand how users personally rate the usability and design of an interactive product.

By using the short version of this questionnaire which is comprised of ten questions, we could apply an evaluation method that records both the perceived pragmatic quality, the hedonic quality as well as the attractiveness of our system. In order to quantify the attractiveness, this questionnaire applies an instrument of measurement in the format of semantic differentials, which consists of ten seven-step items whose poles are opposite adjectives (e.g. "confusing" – "clear", "good" – "bad" etc.). Each set of adjectives is ordered into a scale of intensity.


Assessment of **Retina**

With the help of the word pairs please enter what you consider the most appropriate description for **Retina**.

Please click one item in every line.

simple*	○	○	○	○	○	○	○	complicated
ugly*	○	○	○	○	○	○	○	attractive
practical*	○	○	○	○	○	○	○	impractical
stylish*	○	○	○	○	○	○	○	tacky
predictable*	○	○	○	○	○	○	○	unpredictable
cheap*	○	○	○	○	○	○	○	premium
unimaginative*	○	○	○	○	○	○	○	creative
good*	○	○	○	○	○	○	○	bad
confusing*	○	○	○	○	○	○	○	clearly structured
dull*	○	○	○	○	○	○	○	captivating

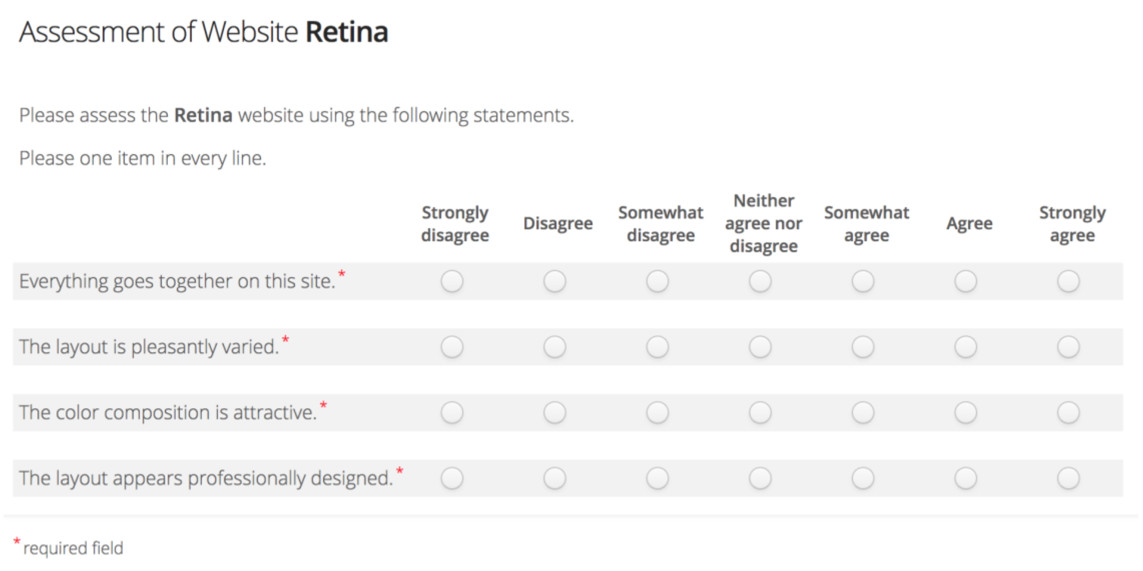
* required field

Figure 6.1 - AttrakDiff Questionnaire

The second questionnaire was VisAWI [34] (Visual Aesthetics of Websites Inventory), which can be used to show us how users subjectively perceive the visual aesthetics of our graphical interface.

In order to record the users' subjective evaluation, they are presented with statements on the interface's design features. Then the users can agree or disagree with these statements. VisAWI records four core aspects of the system's aesthetics based on the user's point of view. These aspects are simplicity, diversity, colorfulness and craftmanship.

Simplicity comprises the clarity and the structure of the system's layout. Diversity reflects aspects such as the creativity and the dynamics of the design, while colorfulness deals with the aesthetic evaluation based on the placement as well as of how the colors are combined. Craftmanship evaluated whether the design is up-to-date and whether it was developed with skill and care, showing the professionalism and the attention to details of the developer.



Assessment of Website **Retina**

Please assess the **Retina** website using the following statements.

Please one item in every line.

	Strongly disagree	Disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Agree	Strongly agree
Everything goes together on this site. *	○	○	○	○	○	○	○
The layout is pleasantly varied. *	○	○	○	○	○	○	○
The color composition is attractive. *	○	○	○	○	○	○	○
The layout appears professionally designed. *	○	○	○	○	○	○	○

* required field

Figure 6.2 - VisAWI Questionnaire

6.3 Analysis of Evaluation

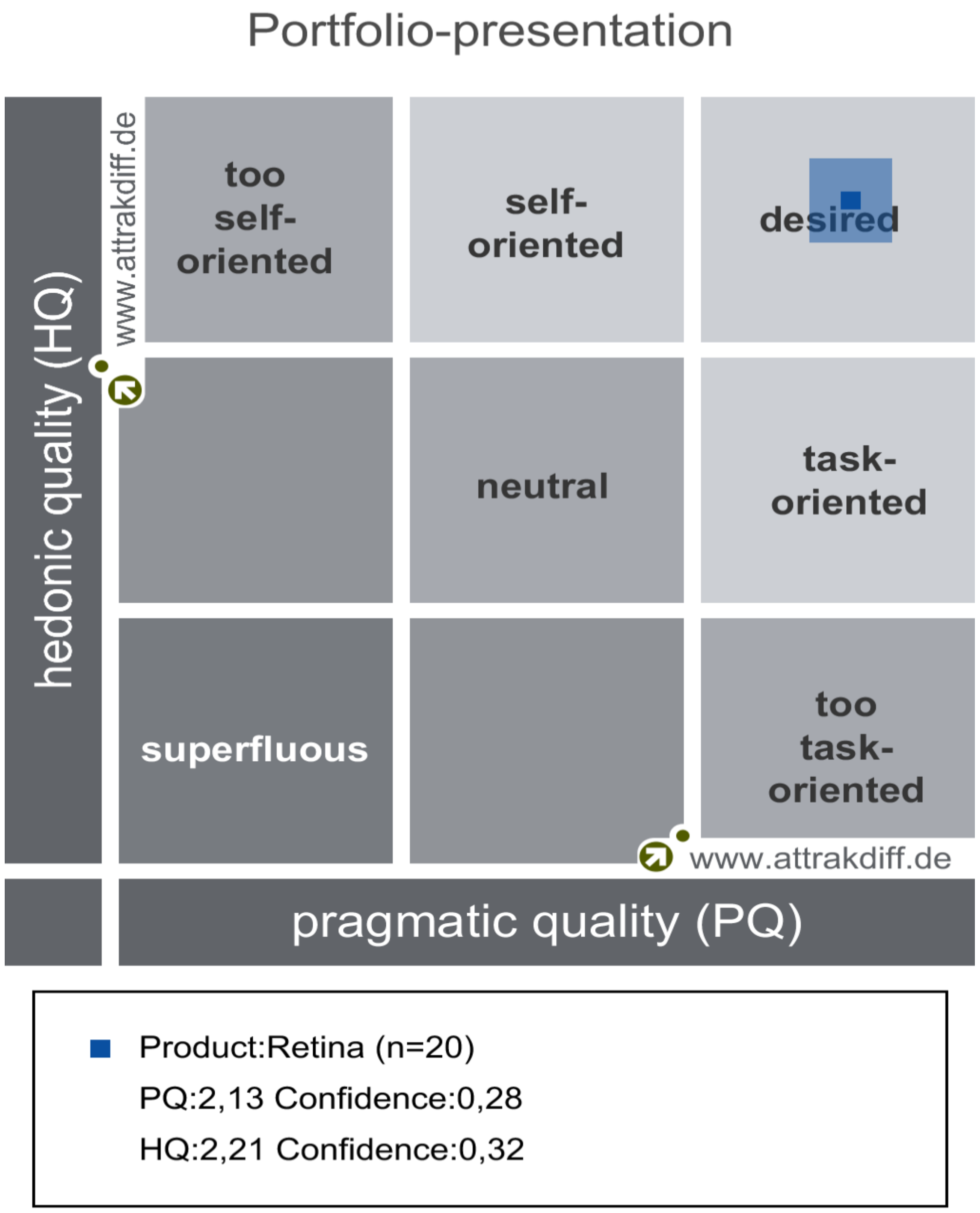
After the twenty users tested our system and answered the two questionnaires we mentioned above, the service provider of these questionnaires provided us with the representation of the results. Based on this fact, the analysis of these results allowed us to draw conclusions in regard to the usability, design, attractiveness and the aesthetics of our system.

Firstly, ***AttrakDiff*** as we mentioned above, measures the usability, design and attractiveness of our system.

It provided us with a ***Portfolio*** (presented below) based on these results which displays the hedonic quality (bottom = low extend) and the pragmatic quality (left = low extend) of our system.

Hedonic quality is based on the aspects of a user interface that appeal to a person's desire of pleasure and avoidance of boredom or discomfort. The aspects that are fun, original, interesting and engaging which will result to a positive subjective experience of the user. Pragmatic quality is calculated based on the usefulness and usability of the system.

Furthermore, depending on the dimension values our system will lie in one or more "character-regions". The bigger the confidence rectangle the less sure one can be to which region it belongs. A small confidence rectangle on the other hand, suggest that the investigation results are more reliable, accurate and less coincidental.



■ Product:Retina (n=20)
PQ:2,13 Confidence:0,28
HQ:2,21 Confidence:0,32

Figure 6.3 - Portfolio of Results

Based on what we mentioned above, and the Portfolio diagram presented the results for the hedonic and pragmatic quality of our Universal Workbench were as follows:

1. Our system was rated well in hedonic and pragmatic quality showing that the users had a positive experience using the system and found it to be useful and usable.
2. The confidence rectangle shows that according to the users, the hedonic quality is greater than the pragmatic quality (PQ = 2.13, HQ = 2.21). For our system the confidence rectangle lies only in the desired region and both pragmatic and hedonic confidence values (PQ = 0.28, HQ = 0.32) are low which suggest that the results of this evaluation are reliable and not coincidental.

In addition to the Portfolio diagram, the service provider, provided us with a ***diagram of average values*** which is presented below. In this diagram hedonic quality distinguishes between the aspects of stimulation and identity. Furthermore, the rating of attractiveness is presented.

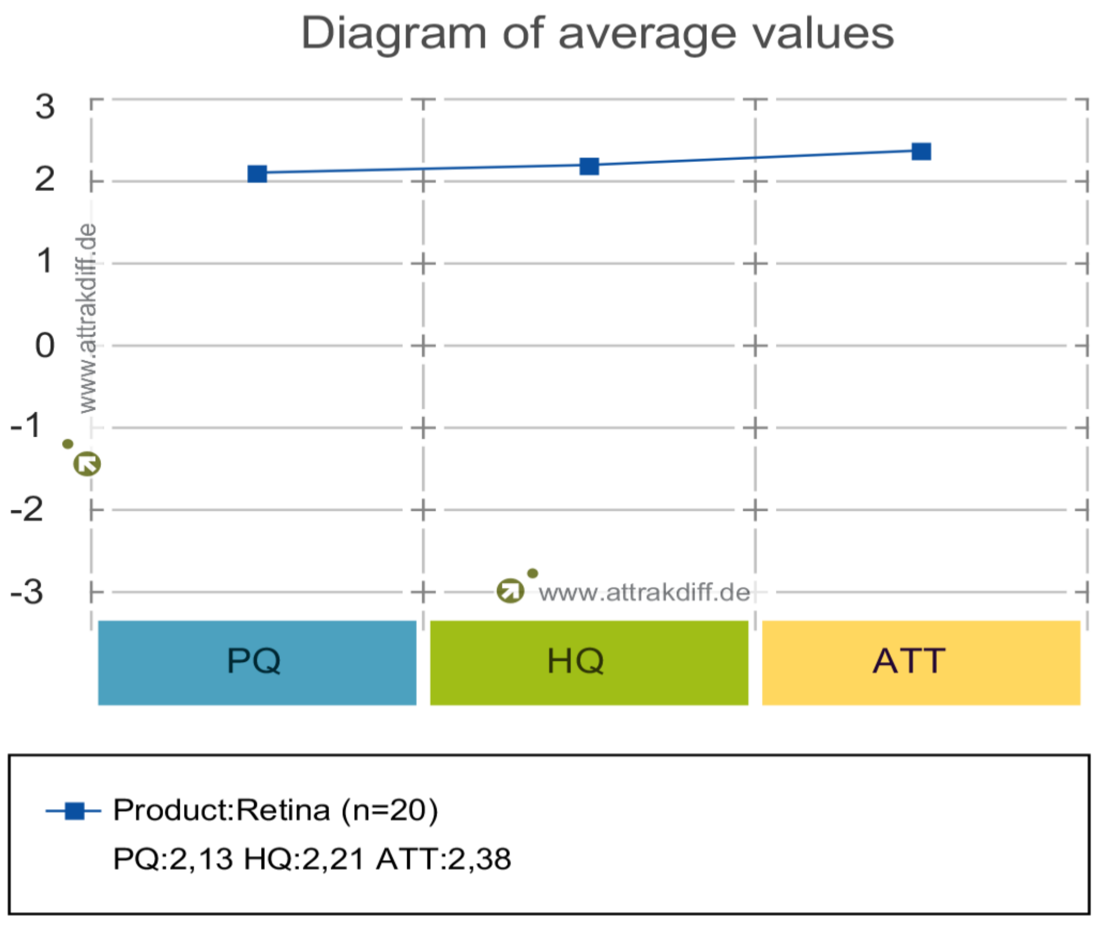


Figure 6.4 - Diagram of average values

This diagram depicts how our system was rated on average in terms of its pragmatic quality (PQ = 2.13), its hedonic quality (HQ = 2.21) and its attractiveness (ATT = 2.38). This led us to the conclusion as we mentioned above that the users had a positive experience while interacting with our system and found its design to be attractive.

Concluding, we were presented with a ***diagram based on the description of word-pairs*** that were used in the AttrakDiff questionnaire.

In the following diagram the mean values of the word pairs are presented. It is worthwhile to mention that of particular interest are the extreme values which show characteristics which are particularly critical or well-resolved.

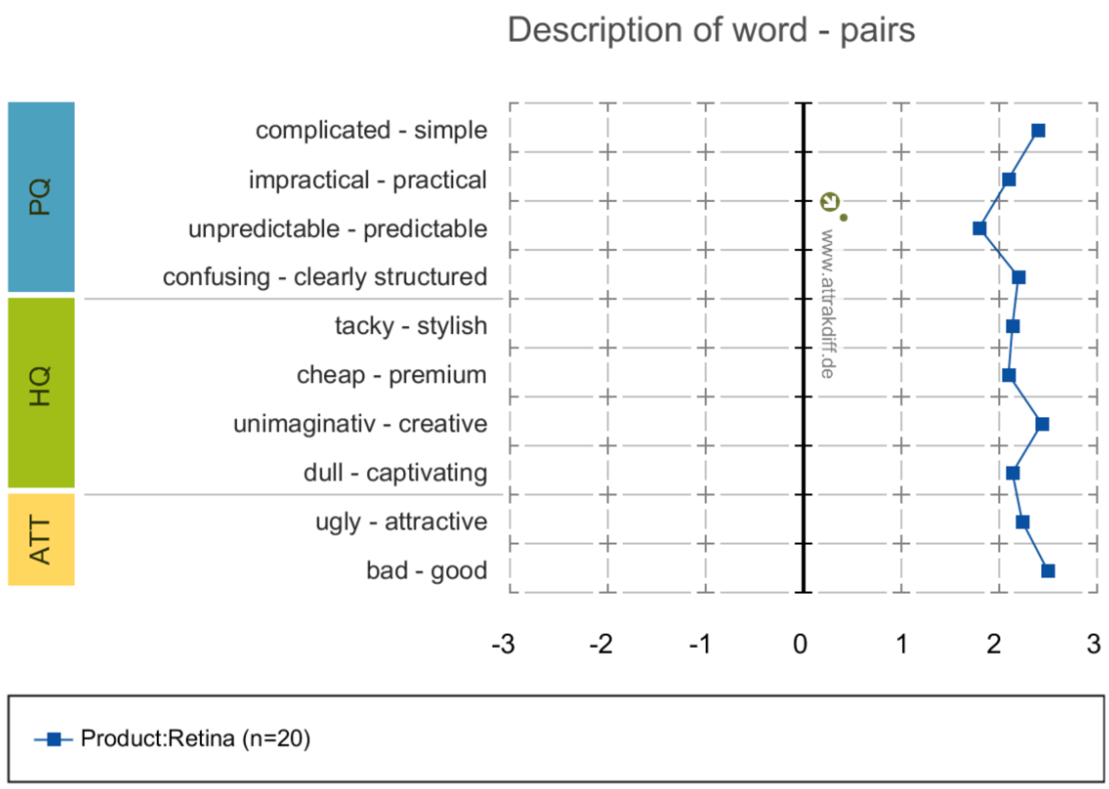


Figure 6.5 - Description of word-pairs

As we can see by the results our system rated well in terms of all the positive words describing the system. The only word-pair with a mean value lower than 2 was the “unpredictable – predictable” pair suggesting that our system’s functionalities were not as clear as the users might have wanted.

Secondly, ***VisAWI***, as we mentioned above, showed us how the users subjectively perceived the visual aesthetics of our graphical interface.

The results were exported and presented with the use of a ***diagram of average values*** presented below which rated our system based on the four core aspects of our system's aesthetics (Simplicity, Diversity, Colorfulness and Craftmanship) as well as provided us with a General rating.

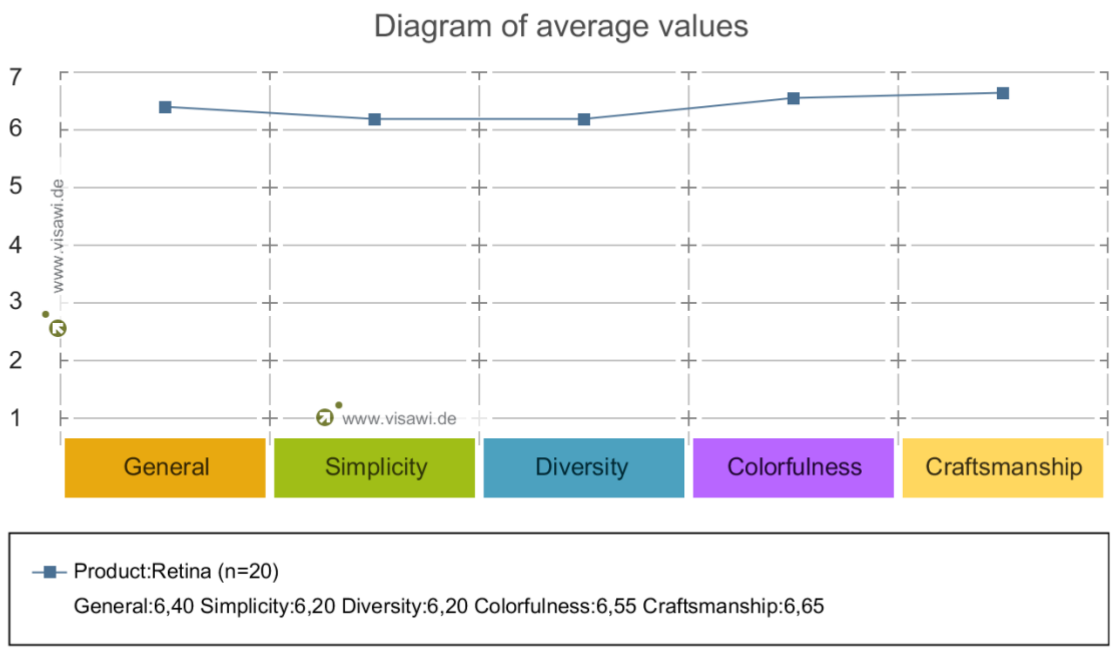


Figure 6.6 - VisAWI Diagram of average values

As we can see by the results, our system scored a General average value of 6.40 suggesting that the visual impact of the system to the users was excellent. In terms of the four core aspects of our system's aesthetics, Craftmanship rated higher with a value of 6.65 followed by Colorfulness with a rating of 6.55 and then Simplicity and Diversity with a score of 6.20.

This leads us to the conclusion that our system's visual impact on the users were extremely good suggesting that the aesthetics of the system were designed properly.

In addition to the diagram of average values, we were presented with a ***box plot diagram*** as well, presented below, which rated again our system based on the aforementioned aspects. In this diagram however, the maximum, minimum and median values were presented.

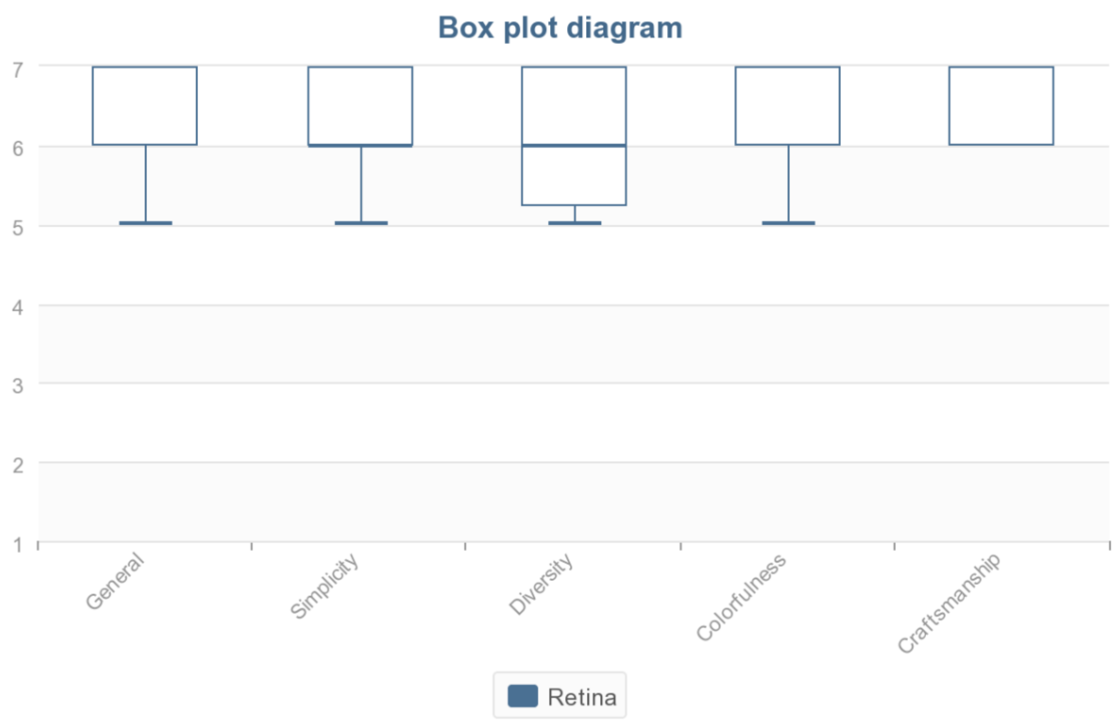


Figure 6.7 - Box plot diagram

The values presented in the above box plot diagram are the extended analysis of the diagram of the average values in terms of the four core aspects of our system's aesthetics as well as the General rating showing the maximum, minimum and median ratings based on the users' answers of the VisAWI questionnaire.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The purpose of this dissertation was the design and implementation of two systems; RETINA, a tool for the real-time analysis and visualization of wearable eye tracking data; and RETINA Universal Workbench, an extensible tool where users can visualize data received from different wearable devices, regarding not only their visual behavior recorded by the first tool using eye tracking devices, but visualize physiological data received from wearable smart devices as well. By tackling the aspect of analyzing wearable eye tracking data in real-time using existing algorithms and refactoring them in order to fit our needs, we showed that it is possible to create a tool for the analysis and visualization of this eye tracking data regarding fixations, fixations' duration, saccades, etc. in real-time, with the help of charts and gaze plots in order to visualize the visual behavior of a user during a session of interaction with a system. Furthermore, by creating a universal workbench that can visualize metrics based on different aspects in the daily lives of the users, we showed that it is possible to create an extensible tool to store data in real-time, and by using the functionalities offered, extract useful information regarding the users' experience while interacting with different technologies in their day to day lives.

By measuring the usability, design and aesthetics of our Universal Workbench by testing it with twenty users, we evaluated how significant the use of such a system would be. By adopting such systems, service providers can understand the different metrics recorded in real-time regarding different aspect of the daily lives of their users, and by understanding

this data, develop interactive systems with the ultimate purpose of offering the users the optimal user experience while interacting with the developed systems.
The analysis of our evaluation showed that our system was found desired by the users (n=20), rating high values in both pragmatic and hedonic quality and attractiveness (PQ = 2.13, HQ = 2.21, ATT = 2.38), as well as showed high ratings in terms of the four aspects of our system’s visual aesthetics (General = 6.40, Simplicity = 6.20, Diversity = 6.20, Colorfulness = 6.55 and Craftmanship = 6.65).
Concluding, as we have already mentioned, eye tracking recently became a cornerstone in research, and by analyzing this data in real-time usability researchers can extract meaningful information regarding the visual behavior of users, locate common visual behavior patterns, again with the purpose of developing user-friendly interactive systems based on those patterns.

7.2 Limitations

During the evaluation of our system and based on the comments received from the users who tested our ***Universal Workbench*** system, we identified some limitations regarding the ***deployability*** as well as the ***performance*** of our system.
Some limitations were that in certain cases our system was ***browser-incompatible (specifically Google Chrome),*** as it presented problems regarding the correct visualization of data and the correct execution of certain basic functionalities. In order for the users to properly interact with those functionalities they had to change their client.
Furthermore, our system is not ***mature*** yet. Our system was implemented and deployed for evaluation purposes and tested only by twenty users, which means that we do not know yet how it would perform on a large-scale concurrent evaluation process.
Concluding, the limitations presented while testing the ***RETINA*** real-time analysis and visualization tool of wearable eye tracking data, were in regard to the technologies and the knowledge required in order to setup the system and for it work properly. Some problems were presented while installing the technologies required due to the errors presented in the updated version of the dependencies we tried to install. Based on this fact, certain functionalities that were critical for the eye tracking process stopped working, which led us to use out-of-date dependencies in order for our system to work properly.

7.3 Future Work

As we have mentioned, our motivation was the implementation of a tool for the real-time analysis and visualization of wearable eye tracking data, as well as implementing a universal workbench tool for visualizing different metrics based on the user's recorded data from wearable devices.

The expectations of the ***RETINA Universal Workbench*** are to turn it into a more efficient and attractive system for everyone who uses it in order to offer optimal user experience, so the users will want to use it to preview their metrics. For this reason, the RETINA Online Workbench has room for improvement, in regard to the functionalities offered regarding data manipulation and visualization as well as the supported devices and the factors that can be visualized. By supporting more wearable devices and extending our architecture and implementation further, to support the visualization and analysis of more factors, the ***Online Workbench*** can be constituted as the backbone system used by service providers, to understand the human behavior in general; in contrast to the current support which is the visualization of the visual behavior and the visualization of physiological data of users.

Furthermore, it is worthwhile to mention that by developing the ***RETINA*** (real-time analysis and visualization of wearable eye tracking data) tool, in order to classify fixations and saccades in real-time and find the duration of fixations as well as the fixation count of users while interacting with a system and present them in charts and gaze plots; RETINA can be used as a tool by service providers in order to understand the visual behavior of the users during the interaction with their system. Based on this fact, RETINA can be used as an evaluation tool in specific fields, such as analyzing and visualizing the visual behavior of the users in real-time in a graphical user authentication process both in desktop environments as well as VR/AR/MR environments.

By enabling service providers to observe their users' visual behavior from wearable eye tracking devices as well as observing their physiological data received from wearable smart devices while they create a graphical password, the service providers will be able to generalize visual behavior patterns and locate common patterns and trends across the visual behavior of the users and develop policies and tools such as how emotions affect users in a graphical password creation process or create a password strength meter to measure the security/complexity of a graphical password in real-time and suggest users to change their passwords in certain cases.

Bibliography and References

[1] Raptis, George & Fidas, Christos & Avouris, Nikolaos. (2016). Using Eye Tracking to Identify Cognitive Differences: A Brief Literature Review.

[2] H. A. Witkin, C. A. Moore, D. R. Goodenough, and P. W. Cox. Field-dependent and field-independent cognitive styles and their educational implications. ETS Research Bulletin Series, 1975(2):1–64, 1975. doi:10.1002/j.2333-8504.1975.tb01065.x.

[3] Dario D. Salvucci and Joseph H. Goldberg. 2000. Identifying fixations and saccades in eye-tracking protocols. In Proceedings of the 2000 symposium on Eye tracking research & applications (ETRA '00). ACM, New York, NY, USA, 71-78. DOI=http://dx.doi.org/10.1145/355017.355028

[4] Gidlöf, K., Wallin, A., Dewhurst, R., & Holmqvist, K. (2013). Using Eye Tracking to Trace a Cognitive Process: Gaze Behaviour During Decision Making in a Natural Environment. *Journal Of Eye Movement Research, 6*(1). doi:http://dx.doi.org/10.16910/jemr.6.1.3

[5] Krzysztof Krejtz, Andrew Duchowski, Tomasz Szmidt, Izabela Krejtz, Fernando Gonza´lez Perilli, Ana Pires, Anna Vilaro, and Natalia Villalobos. 2015. Gaze transition entropy. ACM Trans. Appl. Percept. 13, 1, Article 4 (December 2015), 20 pages.
DOI: http://dx.doi.org/10.1145/2834121

[6] Stephen R. Ellis and Lawrence Stark. 1986. Statistical dependency in visual scanning. Human Factors 28, 4, 421–438.
Miquel Feixas, Esteve del Acebo, Philippe Bekaert, and Mateu Sbert. 1999. An information theory framework for the analysis of scene complexity. In EuroGraphics, P. Brunet and R. Scopigno (Eds.), Vol. 18. EuroGraphics, 95–106.

[7] Stuart, Samuel & Galna, Brook & Lord, Sue & Rochester, Lynn & Godfrey, Alan. (2014). Quantifying Saccades While Walking: Validity of a Novel Velocity-Based Algorithm for Mobile Eye Tracking. 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014. 2014. 10.1109/EMBC.2014.6944931.

[8] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman, Of passwords and people: Measuring the effect of password-composition policies, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (New York, NY, USA), CHI ’11, ACM, 2011, pp. 2595–2604.

[9] Christian Winkler, Jan Gugenheimer, Alexander De Luca, Gabriel Haas, Philipp Speidel, David Dobbelstein, and Enrico Rukzio, Glass unlock: Enhancing security of smartphone unlocking through leveraging a private near-eye display, Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (New York, NY, USA), CHI ’15, ACM, 2015, pp. 1407–1410.

[10] Le Ngu Nguyen and Stephan Sigg, Personalized image-based user authentication using wearable cameras, CoRR abs/1612.06209 (2016).

[11] George E. Raptis, Christos A. Fidas, and Nikolaos M. Avouris, On implicit elicitation of cognitive strategies using gaze transition entropies in pattern recognition tasks, Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (New York, NY, USA), CHI EA ’17, ACM, 2017, pp. 1993–2000.

[12] Marios Belk, Christos Fidas, Panagiotis Germanakos, and George Samaras, The interplay between humans, technology and user authentication: A cognitive processing perspective, Computers in Human Behavior 76 (2017), 184 – 200.

[13] George E. Raptis, Christina Katsini, Marios Belk, Christos Fidas, George Samaras, and Nikolaos Avouris, Using eye gaze data and visual activities to infer human cognitive styles: Method and feasibility studies, Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization (New York, NY, USA), UMAP ’17, ACM, 2017, pp. 164–173.

[14] Kuno Kurzhals, Marcel Hlawatsch, Christof Seeger, and Daniel Weiskopf. 2017. Visual Analytics for Mobile Eye Tracking. IEEE Transactions on Visualization and Computer Graphics 23, 1 (January 2017), 301-310. DOI: https://doi.org/10.1109/TVCG.2016.2598695

[15] Emanuel von Zezschwitz, Alexander De Luca, Bruno Brunkow, and Heinrich Hussmann. 2015. SwiPIN: Fast and Secure PIN-Entry on Smartphones. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 1403-1406. DOI: https://doi.org/10.1145/2702123.2702212

[16] Alpern, M. (1962). Type of movement. In H. Davson (Ed.), The Eye, Vol. 3. (pp. 63-151). New York: Academic Press.

[17] DenBuurman, R., Boersma, T., & Gerrisen, J. F. (1981). Eye movements and the perceptual span in reading. Reading Research Quarterly, 16, 227-235.

[18] Ditchburn, R. W. (1980). The function of small saccades. Vision Research, 20, 271-272.

[19] Erkelens, C. J., & Vogels, I. M. L. C. (1995). The initial direction and landing position of saccades. In J. M. Findlay, R. Walker, & R. W. Kentridge (Eds.), Eye Movement Research: Mechanisms, Processes, and Applications (pp. 133-144). New York: Elsevier.

[20] Just, M. A., & Carpenter, P. A. (1984). Using eye fixations to study reading comprehension. In D. E. Kieras & M. A. Just (Eds.), New Methods in Reading Comprehension Research (pp. 151-182). Hillsdale, NJ: Erlbaum.

[21] Karsh, R., & Breitenbach, F. W. (1983). Looking at looking: The amorphous fixation measure. In R. Groner, C. Menz, D. F. Fisher, & R. A. Monty (Eds.), Eye

Movements and Psychological Functions: International Views (pp. 53-64). Hillsdale, NJ: Erlbaum.

[22] Noton, D., & Stark, L. (1971). Scanpaths in saccadic eye movements while viewing and recognizing patterns. Vision Research, 11, 929-942.

[23] Rayner, K. (1995). Eye movements and cognitive processes in reading, visual search, and scene perception. In J. M. Findlay, R. Walker, & R. W. Kentridge (Eds.), Eye Movement Research: Mechanisms, Processes, and Applications (pp. 3-21). New York: Elsevier.

[24] iMotions Blog (2015, July 9). 7 Most Used Eye Tracking Metrics and Terms, Retrieved from https://imotions.com/blog/7-terms-metrics-eye-tracking/

[25] Tobii Pro Glasses 2 Theory. Retrieved from https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/

[26] Tam J. (2016, November 16). Gazepoint Launches the GP3 HD 150 Hz Eye Tracker, Retrieved from http://www.prweb.com/releases/2016/10/prweb13788227.htm

[27] From Wikipedia, Model-View-Controller. Retrieved from https://en.wikipedia.org/wiki/Model–view–controller

[28] From Microsoft, ASP.NET MVC Overview. Retrieved from https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx

[29] From thedjangoBook, The Django Model-View-Controller Design Pattern, Retrieved from https://djangobook.com/model-view-controller-design-pattern/

[30] Pupil Labs Documentation, Retrieved from https://docs.pupil-labs.com/

[31] Moritz Kassner, William Patera, and Andreas Bulling. 2014. Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp '14 Adjunct). ACM, New York, NY, USA, 1151-1160. DOI: https://doi.org/10.1145/2638728.2641695

[32] L. Chukoskie *et al*., "Quantifying Gaze Behavior during Real World Interactions using Automated Object, Face, and Fixation Detection," in *IEEE Transactions on Cognitive and Developmental Systems*. doi: 10.1109/TCDS.2018.2821566 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8328848&isnumber=7422051

[33] AttrakDiff Questionnaire, Retrieved from: http://attrakdiff.de/index-en.html#tab-einzelausw

[34] VisAWI Questionnaire, Retrieved from: http://visawi.uid.com/

Appendix A

Appendix A consists of important code represented in pseudocode which was written during the implementation of RETINA, our tool for the real-time analysis and visualization of wearable eye tracking data. It also contains the pseudocode of the algorithms presented in our background theory.

I-DT fixation classification algorithm

```
I-DT (dispersion threshold, duration threshold)
      While there are still points
            Initialize window over first points to cover the duration threshold
            If dispersion of window points <= dispersion threshold
                  Add one point to the window until dispersion > dispersion threshold
                  Note a fixation at the centroid of the window points
            Mark onset time
            Mark duration
            Remove window points from points
      Else
            Remove first point from points
Return fixations
```

Quantification and classification of saccades

```
Calculate Saccades (velocity threshold, acceleration threshold, duration threshold)
      Receive raw X and Y coordinates
      Calculate point-to-point position change for X and Y
      Calculate point-to-point velocity
      Calculate point-to-point acceleration
      Convert data from pixels to degrees
      If velocity > 1.000°/sec or acceleration > 100.000°/sec$^2$
                  Disregard data
            Else
            If velocity > 240°/sec, acceleration > 3.000°/sec$^2$, duration < 100ms
                  Increase number of saccades
            Else
                  Increase number of fixations
      Return number of saccades
```

Adjusted algorithm for the real-time classification of fixations on surfaces

```
Fixations Classification (user id)
first frame = True, fixation count = 0, frame count = 1
while true
      If frame count mod 100 is 0
            ending_frame_time = current system time ()
            Store fixation count (user id, fixation count,
            ending_frame_time)
      pupil message = receive message from IPC Backbone
      surfaces = pupil message ['fixations_on_srf']
      If the length of the surfaces metrics is greater than 0
      and If the event happened on a surface
            If not first frame
                  previous id = current id
                  previous timestamp = current timestamp
            current id = surfaces [0] ['base_data'] ['id']
            current timestamp = surfaces [0] ['timestamp']
            fixation position = surfaces [0] ['norm_pos']
            if first frame is True
                  starting timestamp = current timestamp
            if not first frame and if current id is not previous id
                  if current timestamp is not previous timestamp
                        increase fixation count
                        duration of fixation = current timestamp –
                        previous timestamp
                        Store fixation (user id, starting timestamp,
                        duration, fixation position)
                        starting timestamp = current timestamp
            first frame = False
            increase frame count
```

Adjusted algorithm for the real-time classification of saccades

```
Saccades Classification (user id, duration threshold, velocity threshold, confidence threshold)
saccade count = 0, frame count = 1
history = deque (max length = 2), deque (max length = 2)
while true
    if frame count mod 100 is 0
        ending_frame_time = current system time ()
        Store saccade count (user id, saccade count, ending_frame_time)
    pupil message = receive message from IPC Backbone
    if pupil message ['confidence'] < confidence threshold
        // do nothing
    else
        eye_id = pupil message ['id']
        append pupil message to history [eye_id]
        curr timestamp = history [eye_id] [0] ['timestamp']
        prev timestamp = history [eye_id] [-1] ['timestamp']
        if the length of history [eye_id] < 2 or (prev timestamp – curr timestamp > duration threshold)
                // do nothing
            else
            3d method = find all occurrences in history [eye_id] where 'method' is '3d c++'
                if 3d method is not empty
                    find 'theta', 'phi' values in history [eye_id]
                else
                    find 'norm_pos' values in history [eye_id]
                calculate distance
                convert distance to degrees
                if distance > velocity threshold
                    increase saccade count
    increase frame count
```

Appendix B

Appendix B consists of the process required in order to set up correctly the Pupil Eye Tracking device [30] as well as the applications offered by Pupil Labs on a MacOS operating system [30]. This process is required in order for a user to be able to use the RETINA application which executes the algorithm for communicating with the eye tracker as well as the execution of the algorithms for the classification of fixations and saccades. In order for the RETINA system to operate correctly the installation of certain dependencies [30] is required by the Pupil Labs eye tracking device.

Introduction

Firstly, it is required to download the Pupil Labs applications which can be found in their Github repository here: https://github.com/pupil-labs/pupil/releases/tag/v1.7.
The MacOS zip has to be downloaded which contains the Pupil Capture, Pupil Player and Pupil Service applications. For the RETINA system to work properly we only require the Pupil Capture application to be installed.

Pupil Capture Application

Pupil Capture is the software used with the Pupil Headset. The software reads the video streams coming in from the world camera and the eye cameras. Pupil Capture uses the video streams in order to detect your pupil, track your gaze, detect and track markers in your environment, record video and events, and stream data in real-time.
The Capture Window which is the main window opening when loading the Capture application, is the main control center for Pupil Capture. It displays live video feed from the pupil headset including the world camera and the eye cameras.

Pupil Detection

Pupil Labs algorithms automatically detect the pupils of an individual wearing the headset. With the 3D detection and mapping mode, Pupil uses a 3D model of the eyes that constantly updates based on the observations of the eyes. This enables the system to compensate for movements of the headset.

Calibration

Pupil uses three cameras. Two cameras record an individual’s eye movements which are called eye cameras. The third camera records an individual’s field of vision which is called world camera. In order to know where someone is looking, we must find parameters to a function that correlates these two streams of information.

In order for the calibration process to be successful we have to make sure that the user’s pupils are properly tracked and detected by the eye cameras and also that the world camera is in focus.

There are different methods of calibration but the one followed by us is based on the screen marker calibration which is the default method. It is a quick method and it is best suited for close range eye tracking in a narrow field of view. This is suitable for our application as we implemented a web application which requires the user to interact with the system on a laptop display.
In order to follow this method of calibration these steps have to be followed:

1. Select **Screen Marker Calibration**
2. If more than one monitor is connected select the main monitor
3. Disable **Use fullscreen** mode if your Mac has a retina display
4. Press ‘c’ on your keyboard or click the blue circular ‘C’ button in the left-hand side of the capture window to start calibration
5. Follow the marker on the screen with your eyes. Try to keep your head still during the calibration process
6. The calibration window will close when the calibration process is complete

In the advanced sub-menu, the sample duration can be set as well as set parameters that are used to debug and detect the circular marker on the screen. This is not recommended for an inexperienced user.

Network Plugins

As mentioned above in the design and development of the RETINA application, a lot of the Pupil Labs plugins are used for the correct communication of the eye tracker and the methods developed to receive the raw data and pass them through our algorithms.

Pupil Labs has a built-in data broadcast functionality. It is based on the network library ZeroMQ and follows the PUB-SUB pattern. Data is published with an affiliated topic. Clients need to subscribe to their topic of interest in order to receive the respective data. To reduce network traffic, only data with at least one subscription is transferred.

Pupil Remote

This is the plugin that functions as the entry point to the broadcast infrastructure. It also provides a high-level interface to control Pupil Capture over the network (in our case start stop the recording and the sending of the raw data). However, we decided not to use the high-level interface to control the Pupil Remote but written a Python module which does that.

Pupil Time Sync

In order to receive the data from the eye tracking device and have the system's timestamp and not the PUPIL epoch timestamp it is important to synchronize the clock of the sensor. For that we use the Pupil Time Sync plugin which is used to synchronize the time of the device with the system's clock. The Pupil Time Sync plugin can be found here:
https://github.com/pupil-labs/pupil-helpers/tree/master/network_time_sync

Surface Tracking

The Surface Tracker plugin allows to define surfaces within your environment and track surfaces in real-time using NxN square markers. Markers can be used by printing them on paper, stickers or displayed on the screen digitally. A surface can be defined by one or more markers. Surfaces can be defined with Pupil Capture in real-time, or offline with Pupil Player. In our system we define them in a real-time process using the Pupil Capture application where we also edit the defined surface in order to adjust the bounds of the visual field of the user, so the metrics received are robust and as accurate as possible.

In our system we chose to use 4 fiducial markers to define a surface in order to create a rectangle to represent the visual field of the user that we were interested in recording and receiving the data regarding this defined surface. By using the Pupil Capture application, we register this surface, name it and edit the bounds of the surface.

The created surface is streamed over the network with its corresponding gaze positions under the topic ***'surfaces'***. This helps us receive the data relative to this surface and based on the raw data of this topic calculate the fixations occurred on the defined surface.

Pupil Service

Pupil Service is like Pupil Capture except it does not have a world video feed or a GUI. It is intended to be used with VR and AR tracking setups, however we decided to use it in order to control the eye tracking device over the network, send commands and receive the raw data from the device based on those commands.

In order to use this service and since no interface is offered, methods have to implemented in Python with the use of external libraries and dependencies. The dependencies that have to be installed in order to be able for an individual to use the RETINA system will be referred later.

A sample code demonstrating how you can listen to all the notifications from the Pupil Service.

```
from zmq_tools import *
ctx = zmq.Context()
requester = ctx.socket(zmq.REQ)
requester.connect('tcp://localhost:50020') #change ip if using remote
machine
requester.send('SUB_PORT')
ipc_sub_port = requester.recv()
monitor =
Msg_Receiver(ctx,'tcp://localhost:%s'%ipc_sub_port,topics=('notify.',))
#change ip if using remote machine
while True:
    print(monitor.recv())
```

Pupil Data Format

The data format for Pupil recordings is 100% open and can be accessed and processed in any way a user wants.

Coordinate Systems

As mentioned in Chapter 4 Pupil Labs uses a normalized coordinate system with the origin (0, 0) at the bottom left and (1, 1) at the top right.

This is an OpenGL convention and what is found by the Pupil Labs development team to be an intuitive representation. Vectors using this coordinate system are broadcasting their values with a **norm** prefix or suffix.

Timestamps

As we have already mentioned, all indexed data (still frames from the world camera, still frames from the eye cameras, gaze coordinates, pupil coordinates, etc.) have timestamps associated for synchronization purposes. The timestamps are derived from the CLOCK-MONOTONIC on Linux and MacOS.

The time at which the clock starts counting is called PUPIL EPOCH which is adjustable through Pupil Remote and Pupil Time Sync as we mentioned previously.

Timestamps are recorded for each sensor separately. Eye and world cameras may be capturing at very different rates and correlation of eye and world can be done after the fact by using the timestamps.

Some other important metrics which are used by the RETINA system are:

1. **topic:** Field which defines the broadcasted message
2. **norm_pos:** Normalized position of the gaze of the user in the visual field
3. **confidence:** is an assessment by the pupil detector of how sure we can be on this measurement. A value of 0 indicates no confidence and 1 indicates perfect confidence.
4. **method:** String that indicates what detection method is used to detect the pupil.
5. **theta and phi:** 3D model described in spherical coordinates.
6. **id:** Specifies if a fixation has the same id as the previous fixation in order to understand if they belong to the same set of fixations.

All the data recorded are transmitted in the form of a Python dictionary which can look like the following message regarding the pupil datum.

```
{  # pupil datum
    'topic': 'pupil',
    'method': '3d c++',
    'norm_pos': [0.5, 0.5],  # norm space, [0, 1]
    'diameter': 0.0,  # 2D image space, unit: pixel
    'timestamp': 535741.715303987,  # time, unit: seconds
    'confidence': 0.0,  # [0, 1]
    # 2D ellipse of the pupil in image coordinates
    'ellipse': {  # image space, unit: pixel
        'angle': 90.0,  # unit: degrees
        'center': [320.0, 240.0],
        'axes': [0.0, 0.0]},
    'id': 0,  # eye id, 0 or 1
    # 3D model data
    'model_birth_timestamp': -1.0,  # -1 means that the model is building
up and has not finished fitting
    'model_confidence': 0.0,
    'model_id': 1
    # pupil polar coordinates on 3D eye model. The model assumes a fixed
    # eye ball size. Therefore there is no `radius` key
    'theta': 0,
    'phi': 0,
    # 3D pupil ellipse
    'circle_3d': {  # 3D space, unit: mm
        'normal': [0.0, -0.0, 0.0],
        'radius': 0.0,
        'center': [0.0, -0.0, 0.0]},
    'diameter_3d': 0.0,  # 3D space, unit: mm
    # 3D eye ball sphere
    'sphere': {  # 3D space, unit: mm
        'radius': 0.0,
        'center': [0.0, -0.0, 0.0]},
    'projected_sphere': {  # image space, unit: pixel
        'angle': 90.0,
        'center': [0, 0],
       'axes': [0, 0]}}
```

MacOS Dependencies

As we mentioned above, in order for the RETINA system to work properly, the Pupil Labs dependencies have to be installed.
Firstly, it is required to install the Homebrew terminal plugin.

```
ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Then Python >= v. 3.6 has to be installed and exported to our PATH.

```
brew install python3
```

```
export PATH=/usr/local/bin:/usr/local/sbin:$PATH

export PYTHONPATH=/usr/local/lib/python3.6/site-packages:$PYTHONPATH
```

Then the dependencies for the eye tracking device has to be installed in order for us to be able to run Pupil from source code and let it interact with our system.

```
brew install pkg-config

brew install scipy

brew install libjpeg-turbo

brew install libusb

brew install portaudio

brew install opencv

brew install glew

brew install glfw3

brew install boost

brew install boost-python3

brew install ceres-solver
```

Then libuvc is required to be installed which is a cross-platform library for USB video devices.

```
git clone https://github.com/pupil-labs/libuvc

cd libuvc

mkdir build

cd build

cmake ..

make && make install
```

After that, some Python packages which are required for the correct compilation of the source code of Pupil are installed using the pip command.

```
pip3 install PyOpenGL

pip3 install pyzmq

pip3 install numexpr

pip3 install cython

pip3 install psutil

pip3 install msgpack_python

pip3 install pyaudio

pip3 install git+https://github.com/zeromq/pyre

pip3 install git+https://github.com/pupil-labs/PyAV

pip3 install git+https://github.com/pupil-labs/pyuvc

pip3 install git+https://github.com/pupil-labs/pyndsi

pip3 install git+https://github.com/pupil-labs/pyglui
```

These are all the required dependencies in order to run the RETINA system after an individual obtained the Pupil Labs eye tracking device.

Inter-process and Network Communication

In order for the Pupil Capture and the Pupil Service applications to communicate internally as we mentioned previously ZeroMQ library is required. Pupil Labs developed a messaging bus called IPC Backbone and it is used as a message relay station where actors can push their messages and subscribe to other actors' messages. This makes it the backbone of all communication from, to, and within Pupil Capture and Pupil Service.

Pupil Remote

In order for someone to be able to use this message relay and tap to the IPC Backbone, Pupil Remote has to be used. It is required in order to receive the session's unique port. It is the entry point to the IPC Backbone for external applications.
Sample code demonstrating how to 'talk' to Pupil Remote in order to receive the session's unique port.

```
import zmq
ctx = zmq.Context()
# The requester talks to Pupil remote and receives the session unique
IPC SUB PORT
requester = ctx.socket(zmq.REQ)
ip = 'localhost' #If you talk to a different machine use its IP.
port = 50020 #The port defaults to 50020 but can be set in the GUI of
Pupil Capture.
requester.connect('tcp://%s:%s'%(ip, port))
requester.send_string('SUB_PORT')
sub_port = requester.recv_string()
```

By using the Pupil Remote to tap to the IPC Backbone, then it is possible to interact with the use of commands with the Pupil Capture application in order to redirect the subscribed metrics received to our implemented methods for the analysis of the raw data based on the specified topic.

By taking all of the above into consideration, we implemented the RETINA system which uses the IPC backbone in order to communicate with the eye tracking device and receive the raw data representing the gaze information of a user. While receiving these data and based on the fact that we subscribed to the desired topics, we pass the data through our algorithms in a real-time process and export the fixations, fixation count and saccades and store them in our local database. These data are represented in real-time on charts and after the session of a user ends, we depict them on gaze plots related to the fixations and the fixations' duration.

To recapitalize, the process followed in order to able to run successfully the RETINA system with the Pupil Eye Tracking device is:

1. Install the Pupil Labs applications
2. Install the MacOS dependencies
3. Connect the Pupil Labs Eye tracker
4. Use the RETINA application

The intermediate steps presented both in Chapter 4 and previously in this Appendix are only required for the development of applications to receive the raw metrics in a real-time fashion and process them to extract meaningful information. The RETINA system was implemented in order to analyze and visualize in real-time the fixations and saccades of a user.