Final Project Design

EECS 582

Team VI (Phantasos)

Carter Harrod Grant Holmes Sandy Urazayev
Malena Schoeni Rodrigo Figueroa

01/29/2022

1 Project Synopsis

Develop a system which expands the user's interactive capability of interfacing with a computer using multiple sensors.

2 Project Description

The project aims to help develop a new interface that integrates movement from the arms as well as eye tracking hardware and software for user input in a computer system. EMG signals can be collected and analyzed from an arm band. We can also use accelerometer and gyroscope sensors (and possibly other sensors as well) to provide information about the movement and orientation of the arms. Since the forearm muscles control hand and finger movement, we can train a machine learning classifier on the armband EMG data to classify different discrete gestures. Such classification will allow us to directly map intentional hand gestures to traditional key or mouse bindings. We can train an additional machine learning model using the processeed gesture classification, accelerometer, and gyroscope data as inputs to produce more complex, higher level physical directives that include information from the arms in addition to the hands. This project could also be used to help people with limited mobility interact with their environment in addition to creating a new modality of virtual interaction for those with normal mobility. Additionally, eye tracking could be used to increase users productivity by allowing dynamic focus switching of windows and text boxes. Furthermore, eye tracking software could be used

to augment gaming experiences by allowing dynamic mouse acceleration to improve players' performance and accuracy of mouse movements.

3 Project Milestones

- Semester 1
  - Acquire funding, order products, and handle the shipping (Nov. 10th, 2021)
    - Everyone
  - Familiarizing ourselves with the APIs (software interface) (Nov. 30th, 2021)
    - Everyone
  - Set up appropriate workstations with acquired hardware (Dec. 3rd)
    - Everyone
  - Search optimal technical stack for the project (Dec. 10th)
    - Everyone
  - Search optimal machine learning models (Dec. 10th)
    - Everyone
- Semester 2
  - Change focus of windows based on eye tracking (Feb. 1st, 2022)
    - Rodrigo and Carter
  - Develop processing techniques for raw sensor data smoothing (Feb 15th, 2021)
    - Sandy and Malena
  - Work on classification of gestures (Feb. 20th, 2022)
    - Grant and Malena
  - Work on feature extraction algorithms from smoothed signal (Mar. 1st, 2022)
    - Rodrigo and Carter
  - Work on user-oriented API for classification of discrete gestures and tune performance (March 31st, 2022)
    - Everyone
  - Complete GUI to combine functionality of eye tracker and EMG armband (March 31st, 2022)

    - Sandy
- Augment API to allow for combinations of gestures to be recognized as well as provide documentation for API (April 17th, 2022)
    - Everyone

4 Project Budget

Status: Delivered

Device/Product	Description/purpose	Vendor	Estimated Cost
2 x Eye tracker	Track eye movement	Tobii Eye Tracker 5	$600
2 x EMG Armband	Gesture detection	Oymotion	$3000

Table 1: Project Budget

5 Gantt Chart

Description	STATUS	OWNER	START DATE	END DATE	DURATION
Order products	Complete	Everyone	09/12	11/10	43
Tobii eye tracker	Complete	Everyone	09/12	10/14	24
OYMotion arm bands	Complete	Everyone	09/12	11/10	43
Familiarize ourselves with APIs	Complete	Everyone	10/01	11/30	43
Set up appropriate workstations with acquired hardware	Complete	Everyone	11/01	12/03	25
Search optimal machine learning models	Complete	Everyone	11/01	12/10	30
Research optimal tech stack	Complete	Everyone	11/15	12/10	20
Change focus of windows based on eye tracking	In Progress	Rodrigo & Carter	12/01	02/01	45
Develop processing techniques for raw sensor data smoothing	In Progress	Sandy & Malena	11/01	02/15	77
Feature extraction algorithms from smoothed signal	In Progress	Rodrigo & Carter	11/20	03/01	72
Classification of gestures	In Progress	Grant & Malena	01/18	02/20	25
Develop API for classification of discrete gestures and tune performance	Not Started	Everyone	02/15	03/31	33
Complete GUI to combine functionality of eye tracker and EMG armband	In Progress	Sandy	01/15	03/31	54
Augment API to allow for combinations of gestures	Not Started	Everyone	03/01	04/17	35

6 Final Project Design

Our software is going to try to offer an improved human-computer interactive experience by integrating devices that record and analyze data wirelessly, efficiently and in real time. Our software is going to use special equipment that we recently acquired, which includes a sensor for eye tracking and eye control, and two armbands that record the signal from the muscles. The Tobii Eye Tracker 5 is a 13-inch sensor that has to be attached at the bottom of a screen or a monitor and it gets connected to the computer by a USB cable. It tracks head and eye movements and generates a stream of data that is later decoded and interpreted by the desktop application. The way we are going to interact with this device is by accessing the data stream and decode it ourselves through its API and then later assign specific rules or events according to the type of movement or action executed by the user. The gForcePro+ is an armband that uses Electromyography (EMG), which is a technique for recording and evaluating electrical activity produced by skeletal muscles, and produces real time signals that can be further processed. The armband supports up to 16 different customizable gestures. The only limitation on the gestures you can use is that they must produce distinct EMG signals.

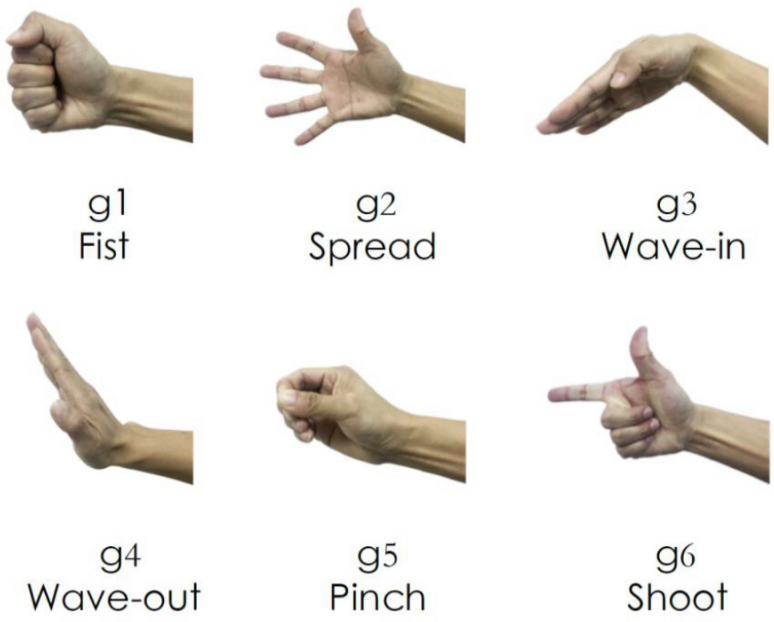

Figure 1: Example of customizable gestures that can be used on the gForcePro+ [1]

For both devices, a real time data stream is recorded and parsed for our uses. For the arm band we are getting raw EMG data for classification of gestures. The data needs to

be processed in a special way to decode which signals belong to specific movements. The data we receive from the eye tracker is the location on the screen. Even though this data is also considered raw data, it is less abstract and more human readable. That is why we are going to spend more time learning and understanding EMG signals in order to develop accurate software.

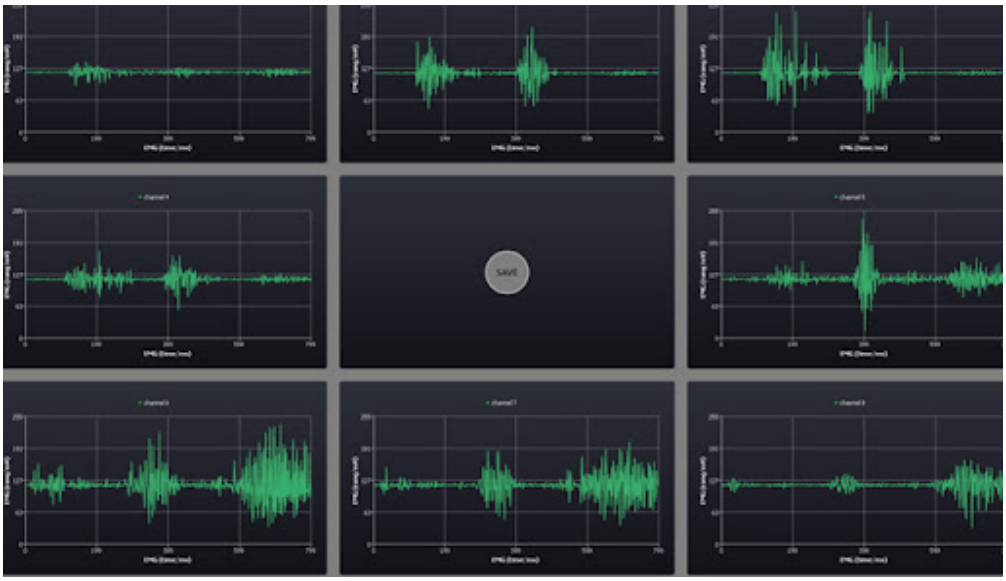

Figure 2: Example of raw EMG data from the eight channels on the EMG armband [2]

As there are the data streams from our devices, there has to be a way for us to properly receive the collected observation readings. Tobii, the manufacturer and proprietor of the eye tracker hardware we use has provided us with their Software Development Kit (SDK), which itself is a collection of C header files and a compiled shared object of the header file symbols' actual implementation. That is the general API we will be working with. Their SDK does support wrappers in various other languages, such as: Python, .NET, and Java. However, for the purposes of reliability, good language ecosystem, and expertise of the team, we decided to build the core logic of our collecting and processing infrastructure in Golang.

Golang (Go) is a statically-typed, garbage-collected, imperative programming language developed by Google'. We chose Go as our main target language thanks to its simplicity of onboarding programmers with the language. Golang is able to be compiled and run on any operating system and chip architecture, which makes our software extremely portable. Compared to Python with its dynamic typing, Go's static type checking during compilation gives us a better guarantee that the program would not fail due to some type errors. Finally, Go's standard library provides a special package that is able to directly

read C header files for a symbol lookup. As Tobii doesn't provide an official API library wrapper in Go, our first implementation step as a team is to write one ourselves. We would implement a wrapper in Go that includes all the C-bindings to read observations from the eye tracker. This is a small constraint to our project, as we have to implement our own tools and API wrappers.

Go supports powerful parallelization and concurrency techniques, such as launching green threads with minimal overhead; this would allow it to process data not only in a single stream, but serialize it across multiple subroutines and threads. As we have outlined in the initial description of the project, we plan to utilize classification-based machine learning tools to differentiate between discrete gestures. However, the overwhelming majority of machine learning algorithms and processes are actively developed and maintained in Python's ecosystem, not Go's. Our solution to this is implement an inter-process communication infrastructure (message passing through REST or similar), such that the code written in Go can extract streams from our devices, later passing the data to a Python process for a possible data cleanup (sometimes preprocessing is done in hardware) and classification, and finally receiving back the decisions and pushing them further in the data pipeline, such as applying made decisions in a game or any user's running application.

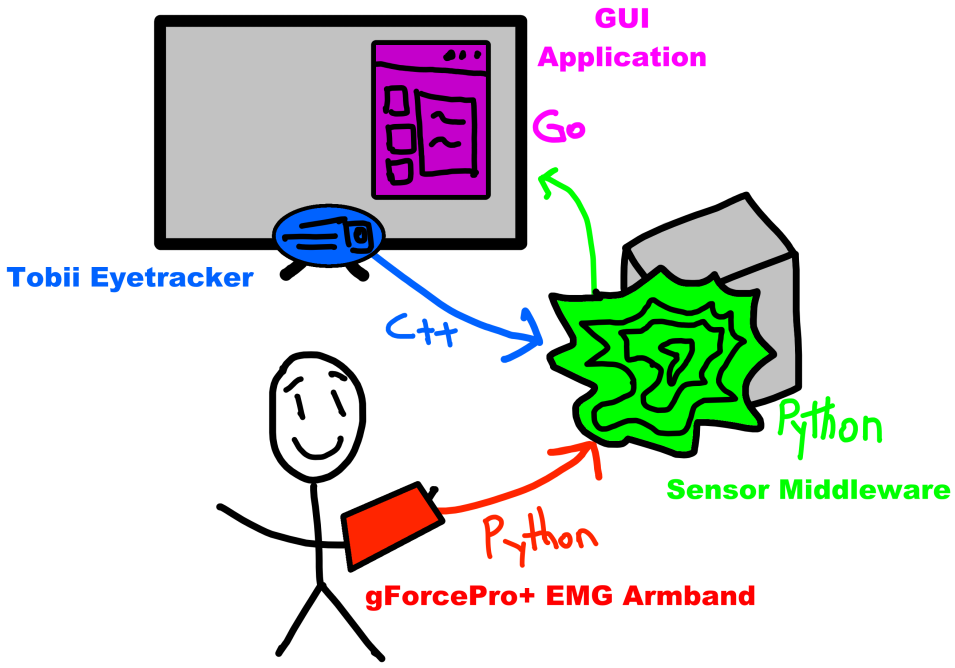

Figure 3: Interface Implementation Diagram

For the final product our aim is to improve the ability of users to interact with their computers. In particular for gaming, we are trying to improve the user inputs that they are already providing. As of right now the goal is to provide a unified interface with both of the devices in Golang. Additionally we want to have Go communicate with a python machine learning program for classification of gestures. Using the inputted data from these interfaces is then used to improve players mouse and keyboard inputs. The eye tracker data can be used to augment the gamer's mouse inputs to make the mouse move in a weighted way towards where they are looking. This will improve reaction time as well as mouse accuracy. The arm band will provide context into what the user is feeling. If the user is in a high stress scenario, we will increase the weight of the eye movements so that their accuracy improves. In a scenario where the user has less stressful inputs, the weight would be lessened to allow for more natural mouse movements. These scenarios can be distinguished by the EMG's reading of tension in the arm. Lastly, due to the nature of the EMG as well as the eye movements, we can try to verify and classify proper inputs from improper inputs, such as misclicks. Depending on the accuracy of the EMG, we may be able to disregard and eliminate accidental input. This could allow for better performance overall in gaming scenarios.

7 Design Constraints

Among the technical constraints we have, the main difficulty is that due to the low level raw data, classification via machine learning is necessary to properly distinguish gestures. Another hard technical constraint is the sensor sample rate. Fortunately, the devices we ordered have a max sample rate of 500 Hz, which will be more than fast enough for our purposes, and on the other hand the eye tracker has a much higher level API. However, We will still need to wrap their SDK implementations, so their SDK serves as another technical constraint.

We face several notable business-oriented constraints. The primary constraint is our deadline. We only have until the end of next semester to complete this project, so we will need to stay organized and stick to our schedule. We do not anticipate facing any prohibitive budget-related constraints, as we have already successfully requested and placed orders for the primary hardware components our project will rely on. We are also constrained by our team size. There are both advantages and disadvantages to having a smaller or larger team, but we will need to consider our team size constraint to best parallelize the development of our project.

8 Ethical issues

Since our project is dealing with signals directly from a person, such as the EMG signals in someone's arm, or the movement of their gaze, there are multiple ethical issues around the collection and privacy of data.

One ethical issue with the eye tracker is that tracking eye movement can possibly reveal medical conditions of those who use our products. For example, if someone has an eye movement disorder such as nystagmus and uses our product, our software could detect a medical condition about the person based on their visual movement that they did not disclose. Medical information and diagnosis are personal private data and fall under HIPPA rules. Furthermore, our software might not be very accurate for people with eye movement disorders which could lead to false categorization or analysis by our software and hardware. This is an ethical issue because it is not accessible to all, and can misrepresent the behavior of those with eye movement impairments.

Another ethical issue with the eye tracker is the privacy and sensitivity of eye tracking data. When using a computer and browsing web pages, eye movement can accidentally reveal personal preferences based on attention and other visual behaviors. Even though we would be using eye tracking to improve mouse capabilities or multitask between web pages and applications, there is still a potential to gain personal data from the eye tracking. Eye tracking information, for example, could be used to make smarter and more targeted ads. Like how social media uses scrolling speeds to determine a user's tastes, eye tracking information could be used in a similar way to collect information about the user. Therefore, it is important that our eye tracking data just be used internally and even when being integrated with other apps and not let it be exposed or sold to third parties.

Furthermore, the eye tracker is something that attaches to the computer or monitor. It is possible for someone to be tracked without their knowledge or consent which would be a breach of their privacy. The eye tracker does utilize a camera and so other visual data could be collected from the background. Our devices should only be used in an environment where all parties are aware of the presence of the eye-tracking software and should be used only in non-sensitive areas.

Likewise, the EMG armbands also pose similar ethical concerns regarding the confidentiality and privacy of biomedical information. The armbands directly measure electrical feedback from the muscles which can be classified to determine gestures and movement of the arms. It is important that the raw data is kept private.

Although the OYMotion armbands open up the potential for a user-interface for those with limited mobility, it does require the wearer to perform hand gestures such as making

a fist, spreading your fingers, and pinching your fingers. This could potentially exclude people who have poor motor control. The armbands could classify gestures incorrectly for people in these groups, or not respond at all.

9 Intellectual Property Issues

The first concern for intellectual property and using the hardware was ensuring that our use case was valid for the Tobii Eye tracking hardware. There were initial concerns about whether using their gaming product for development software was valid under their use case. We reviewed their terms of sales and in the summary the product could be used for "development purposes or for your private, noncommercial use only." Because of our use case, we can use the eye tracker to develop software for their eye tracker. Along with the Terms of sales, their SDK also had a license attached. There were three types of licenses, getting started, commercial, and research. Initially, we wanted to develop under the getting started license due to us not initially having plans for commercial use. This license allows us to develop applications for private, non-commercial use. If we later want to provide a commercial product we can apply later for their commercial license for distribution.

For the arm band, we had similar concerns. However, the SDK and hardware were offered under a much less restrictive license. Due to the hardware essentially being a sensor, it is assumed that the use case is for application development and as such is perfectly viable for our use.

10 Change Log

- Project budget
  - Changed status from ordered and awaiting deliveries to delivered.
- Project Milestones
  - Added milestone for the eye tracker because we didn't have any explicit milestone with that device.
  - Added milestone for the GUI to combine the efforts of the armband and eye-tracker capabilities.

  - Changed the people responsible for the classification of gestures from Grant and Rodrigo to Grant and Malena to reflect who is actually working with the armbands.
  - Changed the people responsible for working on user-oriented API for classification of gestures from Malena and Carter to everyone. We are all working on an API for classification.
  - Pushed dates for some key milestones to semester 2 due to unforseen obstacles we faced in semester 1. Added milestones to semester 1 to reflect things we accomplished instead.

- Gantt chart
  - Updated the roles, tasks and status in the gantt chart to reflect our current status and updated project milestones.

- Design constraints
  - Consolidated our design constraints into their own dedicated section.

References

[1] OYMotion, "Oymotion development," 2021.

[2] OYMotion, "Oymotion emgs," 2021.