PHYS3888 Project Report

Sahir Nagpal- 470433247
Oliver Ashton - 480362302

School of Physics, The University of Sydney, NSW 2006, Australia

June 18, 2020

Abstract

Camera's can be difficult to operate, with a wide range of functions without a button to go along with each one. Here we present the culmination of the work from the past semester in order to provide a solution to this problem. Our team worked on device that allows for autonomous control of a camera, by just moving your eyes. This idea was encouraged as it provides additional degrees of freedom while handling a camera, which in turn provides necessary capabilities for those with disabilities to use cameras more often. This is done by detecting and assigning specific functions of the camera to eye movements and blinks. This is possible due to the dipole nature of the eye, and unique signs that are given upon these evens.

1 Introduction

The human eye, an imperfect marvel of evolution, hides some interesting physics behind its complex biology. In particular, the light - sensitive layer of the eye, the *retina*, is comprised of 10 layers of cells. As the light strikes the retina, a graded potential is generated by the photoreceptive cells, rods and cones, which is further amplified by various other tissues such as the bipolar cells. As this occurs, small dipoles are formed throughout the retina. Once the graded potential reaches the end of the Ganglion cells which make up one of the last layers of the retina and connects to the optic nerve, the stimulus triggers an action potential which is then sent to the optic nerve and processed in the brain[1]. Rotation of the eye (movement) causes the dipoles to rotate, causing a change in the flux of the electric field. A similar potential is seen during blinking which however, arises from the front of the skull[2].

Utilizing this dipole nature of the eye, our project aims to provide additional degrees of freedom whilst using a camera. Different events such as eye movements or blinks produce unique signals which were measured using the Backyard Brains (BYB) spiker box[3]. We processed the data obtained from the spiker box in Python3 where we performed noise reduction using a *Fast Fourier Transform* (FFT); frequency decomposition of a certain sinusoidal wave, and a lowpass filter; a filter that rejects frequencies outside a certain range. We then analysed the data, looking for any events using a threshold classifier; a piece of software that determines the event that occurred. We then took the information found by the classifier, and applied it to a webcam.

The events that we aimed to classify and binding to our camera in the project were: a *left* eye movement, defined as a quick look to the left; a *right* a eye movement, defined as a sharp look to the right; a *double blink*, defined as two blinka in a quick succession; and alpha waves, defined as the resonant frequency of the brain when there is no visual stimulus, producing electrical signal belonging to the frequency range of $8 - 12Hz$.

In our final design we utilised the left and right eye movements as well as the double blink bound these events to key operations in the camera. The left and right eye movements would perform a zoom in and out respectively and a picture to be taken upon a double blink.

2 Events

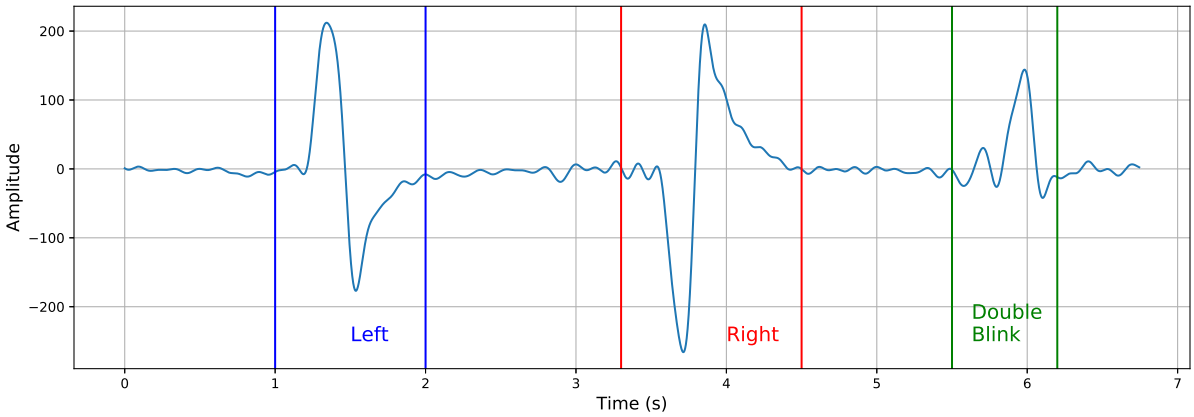

Figure 1: Left event, Right event, Double Blink event

The figure above shows a left event, right event and a double blink event. Each can be identified by some key traits. The left event can be identified by a peak followed by a trough. In figure 1 the left event can be seen occurring between time $\approx$ 1 second to time $\approx$ 2 seconds. The right event can be identified as a trough followed by a peak between time $\approx$ 3.5 seconds to time $\approx$ 4.25 seconds. Finally, a double blink can be identified as two consecutive peaks with smaller amplitudes. This event occurs at time $\approx$ 5.5 seconds and ends at time $\approx$ 6.25 seconds.

3 Design

There were two major components to our final project: the BYB spiker box; and, the software.

Physical setup

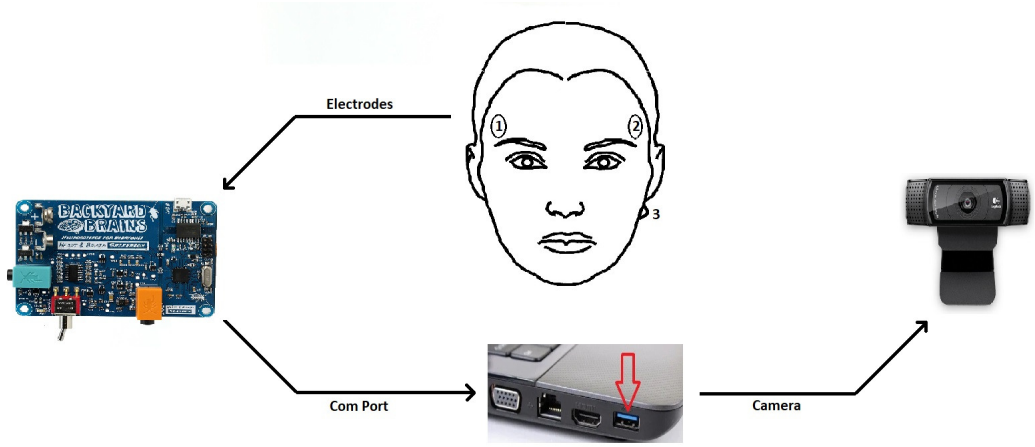

Figure 2: Schematic of the setup[3, 6]

The BYB spiker box is an arduino that measures the potential difference through the electrodes. In our case, we were monitoring the change in potential through the electrodes caused by eye movements. While gathering data for our training set, we noticed stronger and cleaner signals coming from the side of the head. After a little experimentation we found that sticking the electrodes to the temple and having the ground wire behind the ear was the best orientation for collecting data.

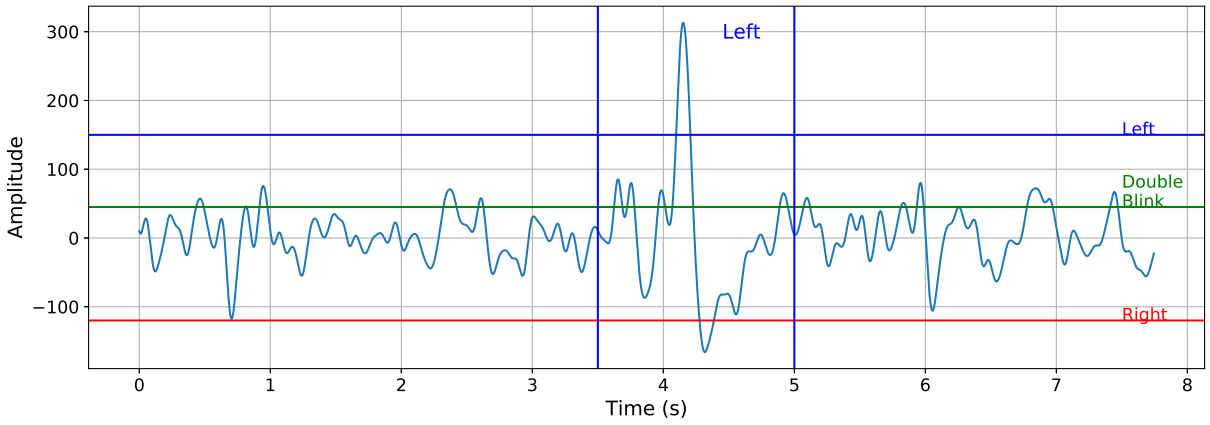

Figure 3: Singular left event with headband electrodes on the forehead

As seen in figure 3, when using the headband, we get massive fluctuations in the noise. As can be seen the double blink threshold had been crossed multiple times without any double blink events occurring, as well as right event potentially being detected at time ≈ 0.75 seconds. Due to these fluctuations, we decided to use the sticky pads which, even the raw data seen in figure 4, lead to much clearer signals with more well-defined peaks and troughs. While we could simply have changed the threshold levels, the ratio of the noise amplitudes to the event amplitudes is far too high, as compared to figure 5, where the noise to signal amplitude is virtually non-existent.

Programming

On the programming side, we used Python3 and Jupyter Notebooks to write our code, as they seemed to be the best middle ground between us and the DATA students. The main focus we had was efficiency to make sure that nothing would slow down our processes. For noise reduction we used a low bandpass filter which removed all the high frequency noise. Next was a threshold classifier that detected if a peak or a trough crossed the thresholds set for left and right events and outputted an event accordingly. As such it was quite efficient.
Finally, we used the OpenCV module to access the webcam. This gave us the ability to access the webcam quickly and easily, with only a few lines of code needed to take the photo. Unfortunately, OpenCV does not have an inbuilt function for zooming in and out yet and hence required an algorithm to be designed.

4 The Process

Physical Component

The physical component of our design had three steps: measurement of potential difference; determination of the optimum electrode placement; and lastly, the input of the data from the spiker box in the software. Electrode placement was a key factor for collection a robust data set. The events we were detecting had the greatest signal amplitudes at the front of the head, so that is where the electrodes were placed. Upon an eye movement; a look to the left, look to the right, or a double blink; the electrodes would detect a change in the

electric field and measure the corresponding change in voltage. These voltage readings were then interpreted by the BYB spiker box as numerical values. Once connected to the software component, we were able to plot both live and prerecorded data as seen in figure 4.

Software Component

The software component of our project had three main parts. The first component was signal processing and noise reduction, followed by event classification, and finally, the event binding to the webcam. Upon receiving data from the BYB spiker box, we applied an FFT to break the data into its frequency components. From here we applied our low bandpass filter which rejected frequencies above a certain value. This allowed us to diminish the majority of the noise from a signal. From here we then performed the *Inverse Fast Fourier Transform* (IFFT), which removed the high frequency noise from the signal. When applying the bandpass filter, we only allowed frequencies between 0 and 8 Hz. This came from observing the frequency domain of a huge data set of about 100 signals.

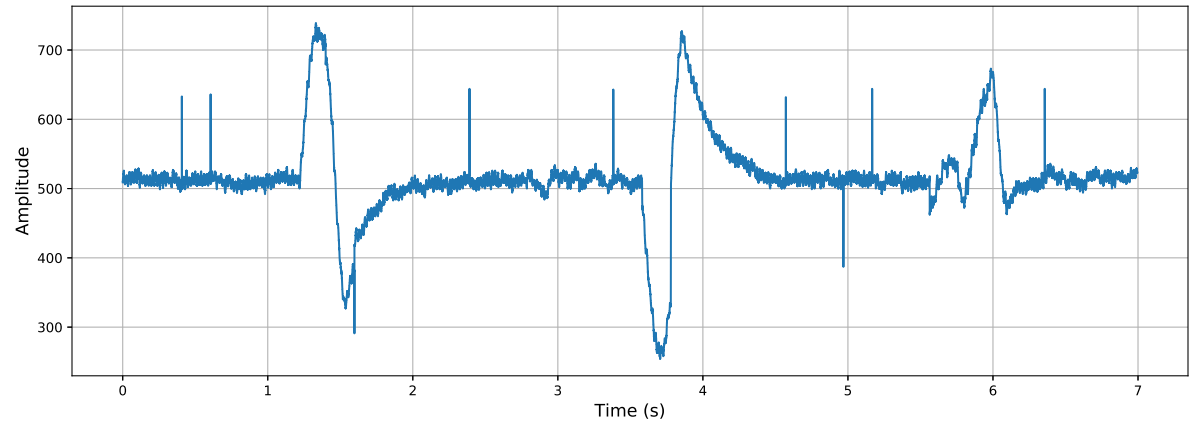

Figure 4: Raw Data

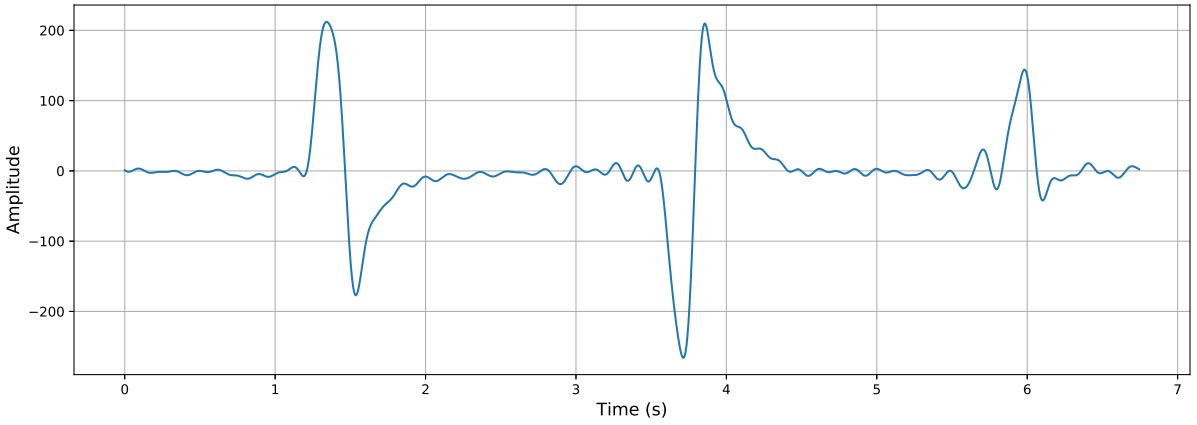

Figure 5: Data post noise reduction

Next the cleared signal was passed through the classifier created by the DATA students. Initially, the classifier was written in R, however, it was subsequently converted to Python3. The classifier operates on a threshold-based algorithm. As seen in Fig. 1, a left event can be classified by a peak followed by a trough. The classifier would observe the wave when a peak is first detected after going over a certain amplitude

threshold. This would trigger a left event to be produced by the classifier. Similarly for the right event, when the trough is first detected, by going below a certain threshold, a right event is detected. Since each signal consists of both a peak and a trough the classifier was designed to stop classifying any events within a certain time frame to avoid a simultaneous left and right detection when only a left event had occurred. Finally, a double blink was classified as when a peak passes through a certain threshold, but does not make it to the left threshold. Initially, it was planned that two peaks pass the double blink threshold within a set time frame whilst not reaching the left threshold, However, due to the unreliability of the BYB spiker box, we were unable to use the above mentioned algorithm. Further elaboration can be found in the *Performance* section.

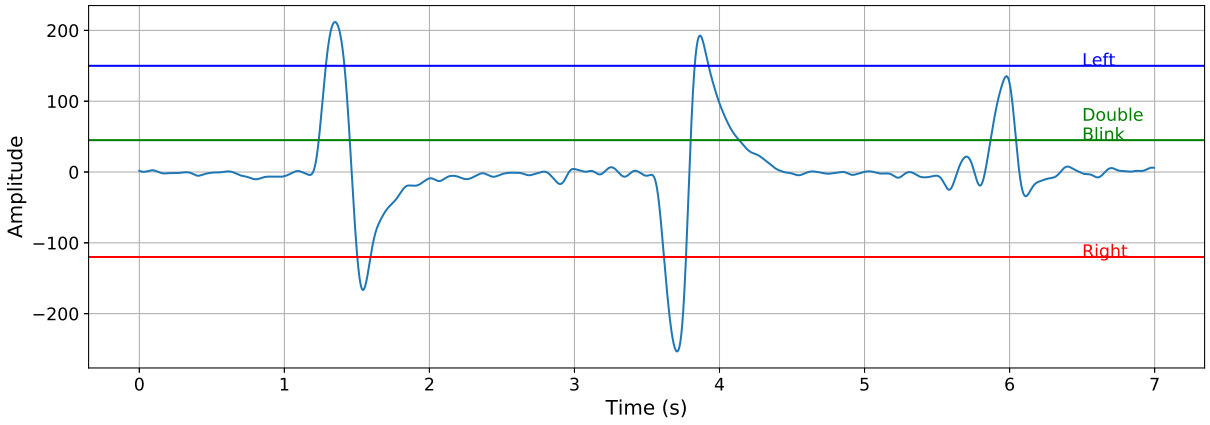

Figure 6: Event Thresholds

Once we had detected an event, they were submitted to a function that operated the camera. This took string input that corresponded to the event. If the input was 'L', a left event, the camera would zoom in; if 'R', right event, was the input, the camera would zoom out; and finally if the input was 'DB', double blink event, the camera would take a photograph.

5 Performance

Performance Review

Throughout the project, we had a big emphasis on reliability, accuracy, and performance. We chose Python3 to be our programming language of choice as it gave us the best performance (see the *Language Comparison* section) while also being a middle ground for both us and the DATA students. Another reason for this decision was the compatibility between R, which the DATA students wrote the classifiers in, and Jupyter Notebooks. As such, Python3 was the obvious choice. As explained in the explained in the *Physical setup* section, we found that attaching the sticky electrodes to the temples gave the best results for data collection. This most likely due to two reasons. Firstly, the temple is a thin part of the skull and hence closer to the eye. This would allow for a change in electric field to be detected more easily. As well as, that the sticky electrodes would reduce the amount of noise we were getting from movement interference, that we were getting from the electrodes in the headband.

Language comparison

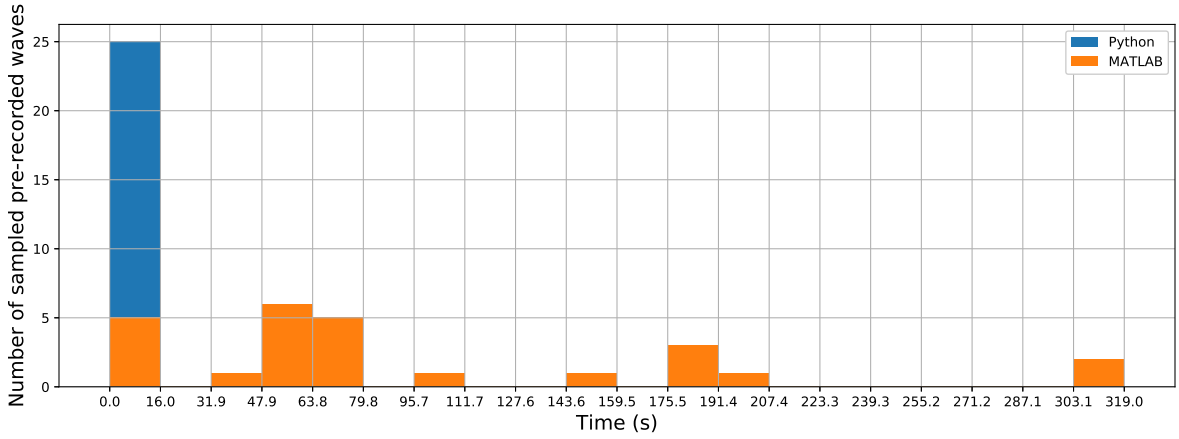

Figure 7: Time taken for Python and MATLAB to reduce noise

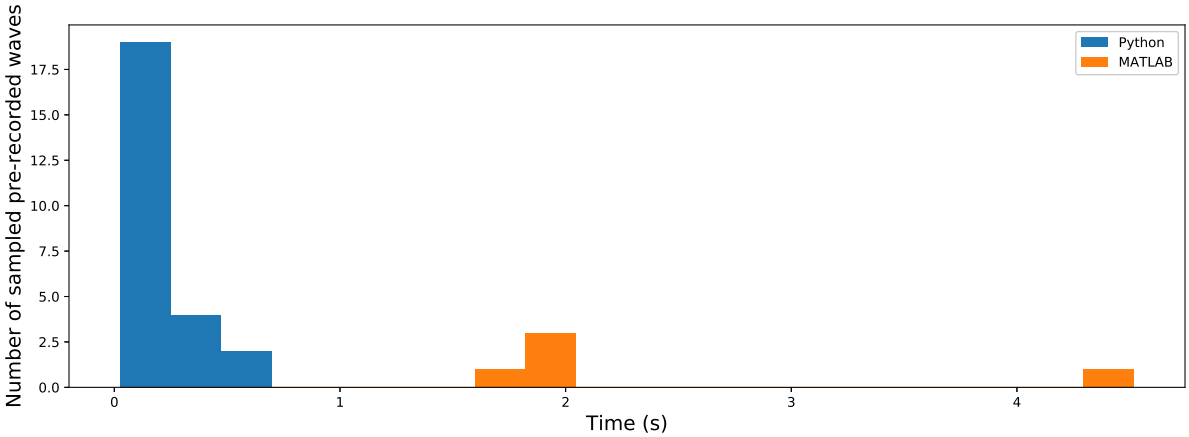

Figure 8: Expansion of the first bar from fig 7

Figures 7 and 8 show a comparison of how well Python3 and MATLAB operate when tasked with large amounts of FFT and IFFT calculations. Python3 was the clear choice as its inbuilt FFT functions suited our algorithm better. This can be seen through the way it handles a multitude of prerecorded cases in less than a second whereas MATLAB is very unreliable requiring times from anywhere from two seconds up to six minutes depending on the length of the wave.

Noise reduction by bandpass filters

We tested three algorithms to perform noise reduction. The first we tested was a lowpass filter. This was the simplest option, while as giving strong noise reduction for minimal computational time. This consisted of simple reduction to zero of the amplitude when above a set frequency in the frequency domain. This method was the most effective as it held a strong ratio between signal amplitude and noise amplitude. It was the most efficient method and only needed logic statements.

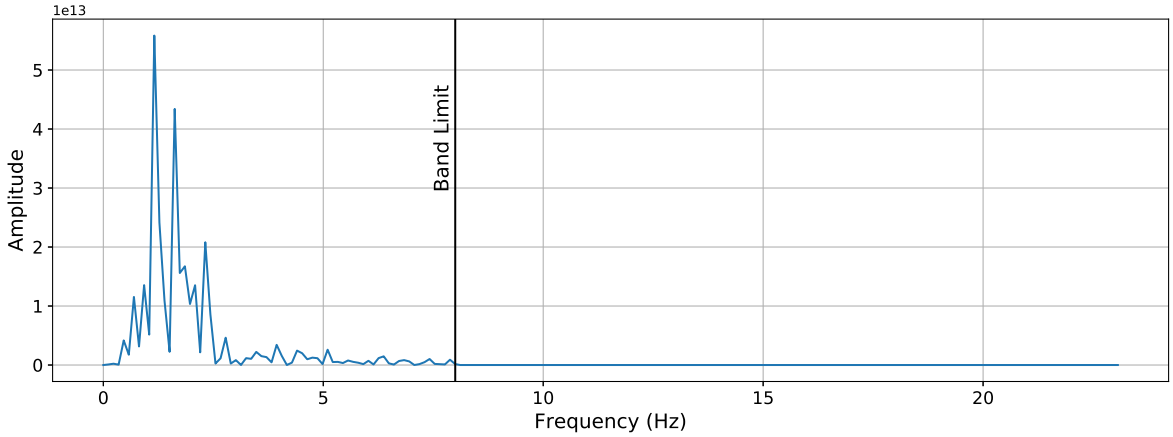

Figure 9: 8 Hz Lowpass filter

Next we tested, we tested a Gaussian distribution fit. This was applied by multiplying the amplitude with the Gaussian function, to allow for an exponential decrease in amplitude weighting[4]. The issue with the Gaussian fit was that the resulting signal had a low noise to signal ratio. While there was not much in the way of an increase in inefficiency from the extra calculation, the loss of contrast between event signals and noise would reduced the accuracy of our classifier.

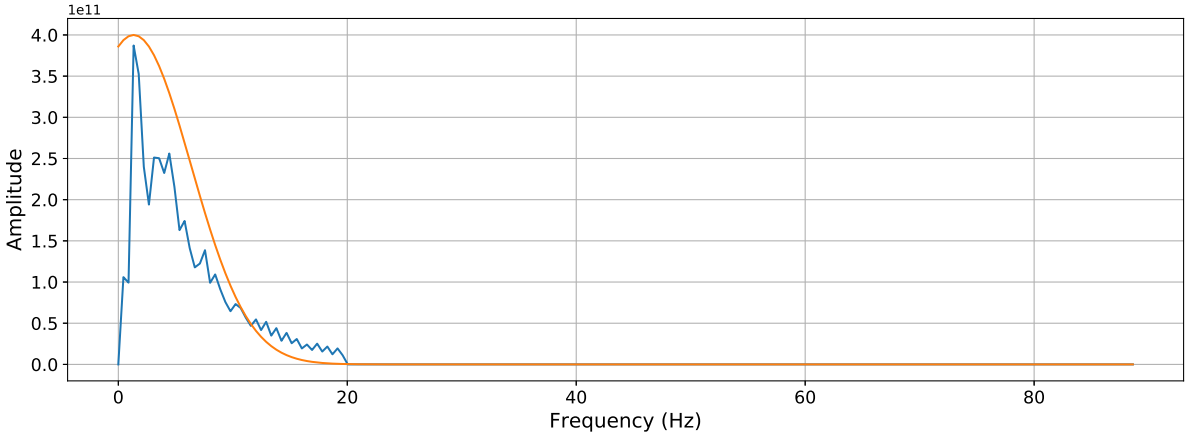

Figure 10: Gaussian fit to a FFT

To achieve an acceptable signal to noise ratio, we cubed the amplitude of the wave after passing it through the lowpass filter. We chose to cube the wave to maintain the amplitude sign, whether it was a positive or negative amplitude. In the end we chose not to proceed with this approach as it led to a very small the ratio of amplitude between a double blink and a left event.

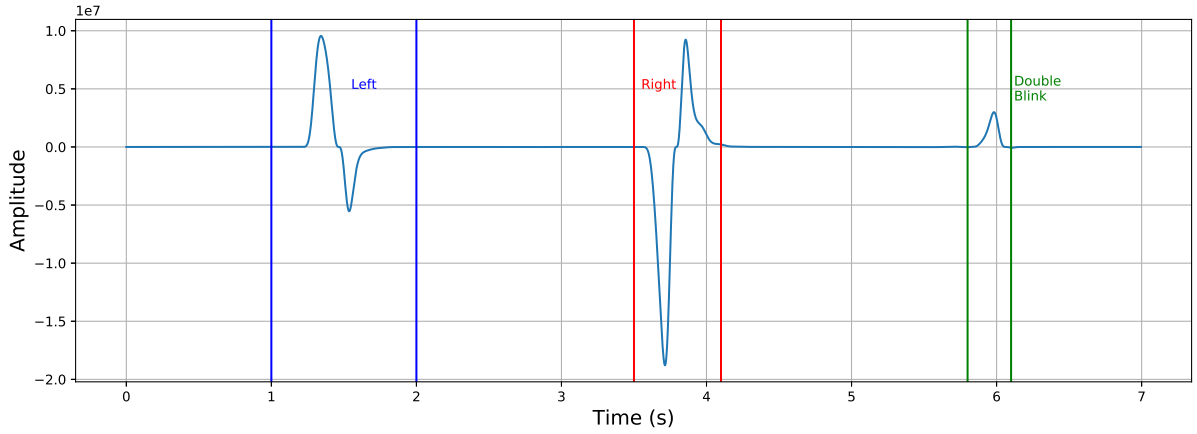

Figure 11: Signal where the amplitudes have been cubed

Zoom algorithm

Since OpenCV, the Python3 module used to access the webcam did not have an inbuilt zoom function, we needed an algorithm to perform the action. The algorithm involved cropping the image and fitting it into the original frame size. This would cause the image to pixelate which resulted in a lower quality image. Moreover, the camera needs to have sufficient resolution so as to not pixelate the image too much. The zoom feature offers 5 discrete zooms in both zooming in and zooming out. Every zoom magnifies the center of the image by 5%. The increase can be made much smaller and as well as be turned into a continuous zoom rather than a discrete zoom. However, there is only so much the eye can do. To allow a more user-friendly control whilst avoiding straining the eye, a discrete zoom seems to be a much better option.

Edge cases

Edge cases occur when an event in the incoming data from the spiker box is divided between two windows. As seen in figure 12 at 4 and 6 seconds (windows 6 and 7), the event occurs at the time a new sample is taken. Since the classifier works on the basis that if a data point is over or under the threshold, the event is detected. This would then result in a double detection. As shown in the case below the detection sequence would be, ['L','R','DB','DB']. We applied several methods to try and solve this problem.

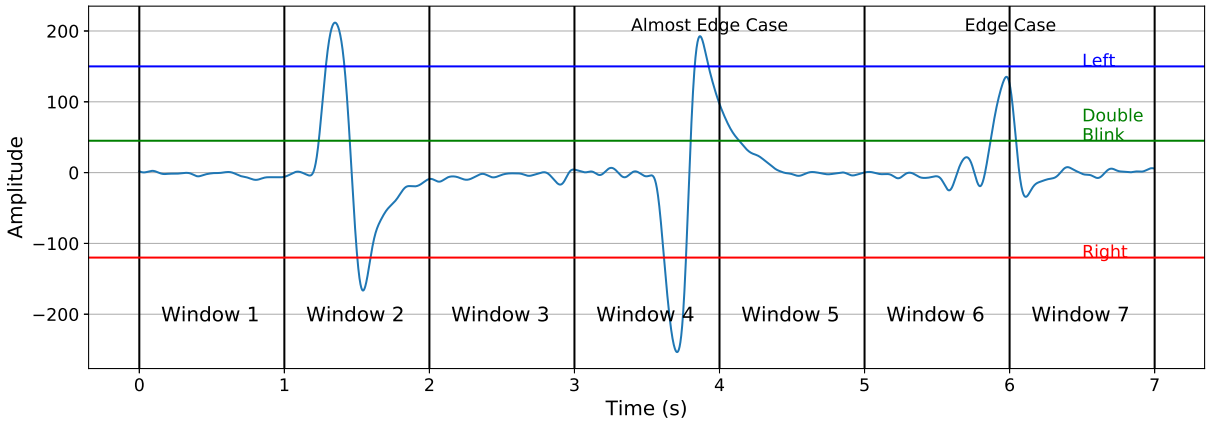

Figure 12: Wave divided into windows of 1 s

The first solution we tested was to increase buffer size, which would consequently increase the window size. This was done to reduce the probability that for an event to occur on the edge of the windows. However, this increased the time lag between an events occurrence and its detection. This would lead to the user trying to apply an action multiple times, where only one was wanted/required. The second way was to increase the time scale we were applying the classifier to however, there were still a lot of false positives coming out of the classifier. Finally, the solution we implemented to deal with the edge cases was that if the data, during live streaming, were to go beyond any threshold, the signal was recorded, and the classifier was applied to recorded wave concatenated with the new data. This method turned out to be the most reliable technique to maintain accuracy with minimal lag between the events and camera implementation.

BYB Spiker Box Faults

Toward the end of the project the BYB spiker box started to show a few faults. the first thing that happened was the presence of some spikes that would occur randomly in a signal. This can be seen in figure 4. We tried changing the device that we used to collect data, but over time it would begin to happen again. It was not that big of an issue however as when noise reduction was applied, these signals were removed, as seen in figure 5. Next, upon initiating the live-stream, a random peak would appear at the start. These peaks remained even post noise clearing. These needed to be omitted as they would sometimes trigger the thresholds, causing an event to be erroneously detected. Third, by the end of the project the BYB spiker box would stop giving signals after 15-20 seconds of activity. This is to say that after sampling for 15-20 seconds the box would no longer give large amplitudes, resulting in no thresholds being breached and hence no events being detected. This was not too much of an issue but is definitely could have become one if more testing was needed. Finally, the BYB spiker box began to randomly detect triple or quadruple peaks upon a single event, as seen in figure 13. This started happening around week 9 or 10, but then suddenly stopped the following week.

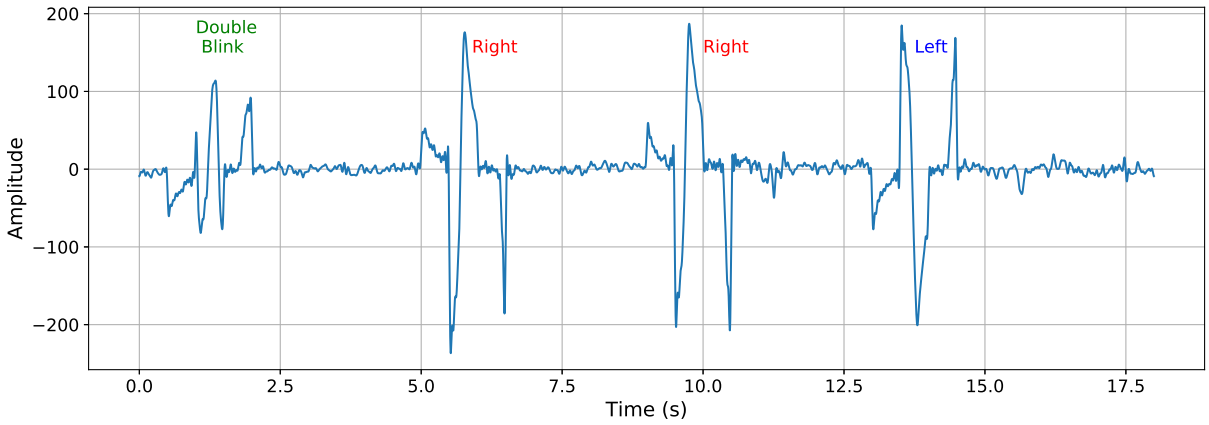

Figure 13: A sample recording with unexplained quadruple peaks for each event

Apart from the odd signal at the beginning of the data and the absence of robust signal after a certain timeframe, all problems seemed to be resolved themselves or were fixable with noise clearing.

6 Discussion

See *individual discussion*

7 Further Remarks

Throughout the project there were a few things that we did not attempt in terms of software. One thing would have been testing more types of classifier in the live code. It seemed that while the threshold classifier worked the best in an environment with only prerecorded data, it may have not been the best option for a live-streaming environment. On the administrative side of things, we would definitely be more organised and have a strong sense of timeline and having better communications, trying to get progress updates more often.

If we had more time and access to laboratories on this project, we would implement a timer that begins its count down upon the detection of alpha waves. We did not have enough BYB spiker boxes, as alpha waves are best detected when electrodes are placed on the back of the head, while then event we were detecting needed the electrodes are the front of the head. Secondly, we would have been stretched for time in terms of classifier creation for the alpha waves.

In terms of our current situation, there were two main limiting factors which inhibited on our productivity and the quality of our final project. First thing was the limited to no personal interaction due to the Coronavirus (COVID-19). Due to this, it was difficult to maintain a strong work relation, and ....
The second being the unreliability of the spiker box. The spiker box would be quite temperamental and give triple and quadruple peaks one one event. In an overall context, the minimal amount of eye movements and events we can classify. Events such as looking up and down have the same signal as left and right eye movements[2]. Without more spiker boxes, we would not be able to classify more events and limiting the capabilities that we can assign to the camera.

8 Interdisciplinarity

The DATA students were solely responsible for the creation and operation of the classifier. They tested two types of classifier algorithms over a two week period. The two that they tested were the one that was used in the final product - the threshold classifier which was explained in the performance section, and an algorithm that works with comparing test statistics. This algorithm calculates test statistic of the entire wave.

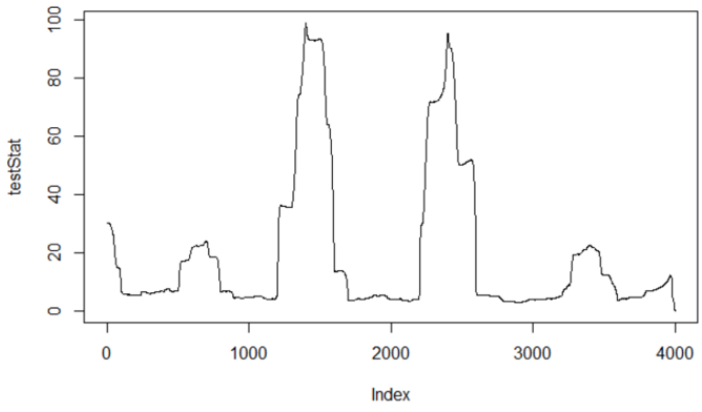

Figure 14: Test Statistics for the wave DB_L_L_DB

If the test statistic crosses the threshold of 20 but lower than 60, the event is classified as a double blink. If the test statistic crosses 60, the event is classified as either a left or a right. The differentiation between the left and right events is done by looking at the wavfile. If the amplitude of the wave went down in the beginning, the event was classified as right. Conversely, if the amplitude of the event increased, then the event was classified as a left.
To pick which classifier to use, the DATA students tested them against prerecorded data to find the most efficient and the most accurate algorithm. The threshold algorithm, described in section 4.2, was the simplest,

but the most effective classifier, giving the highest accuracy to efficiency ratio when applied to prerecorded data.

9 Conclusion

Our aim in this project was to design as camera which can be operated via eye movements providing additional degrees of freedom to all the camera users, especially the physically disabled. Two electrodes are attached to the temples of the user and one behind the ear. The change in potential due to eye movements is measured and processed by the spiker box and the data is fed to a computer. The software in the computer cleans the data, identify events if any and triggers the action intended to perform via the camera.
This project is just the tip of the iceberg compared to the full potential of our idea. Some future prospects for our project are: the incorporation of the electrodes into a pair of glasses. This would be combined with making it a wireless design and converting the BYB spiker box into a phone app or a dongle, all in an effort to make the experience less intrusive and far more streamlined. Secondly, a machine learning or even an AI classifier can be implemented which may provide a much higher efficiency. This would give us a fairly better accuracy for events while live-streaming. However, this was too far out of the scope of the course and time frame we had. Finally, would be to access a professional camera, and make use of the additional features, such as focus and shutter speed. There is also the potential for other ways to perform event detection such as facial tracking and muscular events.

10 Files and the code

The files and the code are accessible *here*

11 Acknowledgements

We would like to thank Kathy Huang, Johnson Yun, and Frida Jiao for their contribution toward the project, working solely on the classifier. We would also like to thank Dr. Alessandro Tuniz, Alison Wong, and Zoe Stawyskyj for their guidance throughout the project.
We would also like to acknowledge and pay respect to the traditional owners of the land on which we meet (the Gadigal people of the Eora Nation). It is upon their ancestral lands that the University of Sydney is built.
As we share our own knowledge and research practices within this University may we also pay respect to the knowledge embedded forever within the Aboriginal Custodianship of Country.

References

[1] Barrett, Kim E., and William F. Ganong. *Ganong's Review of Medical Physiology.* Twenty-Fifth Edition. New York: McGraw-Hill Education, 2016. Print.

[2] Berg, P., Scherg, M. (1991). Dipole models of eye movements and blinks. *Electroencephalography and Clinical Neurophysiology* doi: 10.1016/0013-4694(91)90154-v

[3] The Heart and Brain SpikerBox. Retrieved 2 June 2020, from `https://backyardbrains.com/products/heartAndBrainSpikerBox`

[4] University of Edinburgh, School of Physics (2007). Topic 6: Digital Filtering. `https://www2.ph.ed.ac.uk/~wjh/teaching/dia/documents/filtering.pdf`

[5] 28 The Electric Signals Originating in the Eye. (2020). Retrieved 2 June 2020, from `http://www.bem.fi/book/28/28.htm#:~:text=28.1%20INTRODUCTION,negative%20pole%20a%20the%20retina.`

[6] Hallacoglu, B., Sassaroli, A., Fantini, S. (2013). Schematic diagram of the experimental setup for the human subject measurements. Retrieved 2 June 2020, from `https://figshare.com/articles/_Schematic_diagram_of_the_experimental_setup_for_the_human_subject_measurements_/705290/1`