Brain Controlled Robot Navigation Based on Low Cost EEG

by

Yang Shi

(Under the Direction of WenZhan Song)

Abstract

This thesis focuses on employing low cost EEG signals to control robots for navigation tasks. A data driven signal processing and machine learning framework is proposed and applied. Power Spectral Density (PSD) and Spectral Analysis are used for feature extraction, and I examined the result of Principle Component Analysis (PCA), and chose non-linear classifiers for machine learning. The algorithm for classification is Quadratic Discriminant Analysis (QDA), and achieved around 88% to 91% accuracy for five-fold cross validation. When testing with a new dataset, the accuracy is around 82%, but will be low in contaminated datasets and at varying electrode locations. I also experimented the real-time system, and most instructions are correctly classified. This thesis provides a novel system for EEG data processing, especially for situations of low cost, low channel amount equipment.

Index words: EEG, BCI, Brain wave, Machine learning, Quadratic Discriminant Analysis, Principle Component Analysis, Spectral Analysis, Power Spectral Density

Brain Controlled Robot Navigation Based on Low Cost EEG

by

Yang Shi

B.E., Central South University, China, 2015

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2017

Brain Controlled Robot Navigation Based on Low Cost EEG

by

Yang Shi

Approved:

Major Professor: WenZhan Song

Committee: Tianming Liu
Hamid R. Arabnia

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
December 2017

Brain Controlled Robot Navigation Based on Low Cost EEG

Yang Shi

December 6, 2017

Acknowledgments

This thesis and everything would not be possible without generous support, instruction, and guidance from Dr. WenZhan Song, which I cannot be more grateful for.

I would also like to thank Dr. Tianming Liu and Dr. Hamid R. Arabnia for their excellent teaching and advisement.

My graduate life has been blessed because of my many friends. Even when things get tough, they never stop caring and supporting me. I would specially thank my postdoctoral colleague Dr. Fangyu Li for his suggestions on issues of signal processing, and PhD student colleague Zhiwei Luo for the help in building up the prototype of the system. I would also thank visiting PhD candidate Minhui Zou for working together at nights and days. I would never forget these hardworking and memorable days.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Previous Work

Human brain generates electrical signals, and one way to research into this signal deeper into brain science is electroencephalogram (EEG). It is non-invasive, and by the electrodes places along the scalp, this technique measures the voltage fluctuations from ionic current changes in neurons from the brain. Some typical applications of EEG are disease diagnosis, including but not limited to epilepsy [2], sleep disorders [3] and other brain disorders. And EEG can also be applied to some research topics like this thesis project, which is generally categorized as motor imagery.

It is known that human brain is vastly complex and varies from different person, and thus, this special quality attracted a lot of interest from different groups of researchers. Different researchers tend to have different tool set for this task available from academia or industrial according to their research tasks and expectations. Some common software tools are listed here: EEGLAB [4], BCILAB [5], BCI2000 [6]. For hardware, OpenBCI [7] is one of commonly used hardware provider showed up in market recent years for low cost EEG signal computation. It provides up to 16 channels of data collection through different

places on scalp, and provides data transferring pipeline based on RF modules. It is light weighted, and people can wear it and walk to anywhere, which is not available for most of other equipment. And it is much cheaper than a whole system usually used for hospitals and laboratories. OpenBCI is a new platform showed up in recent years, and this platform provides a set of tools for processing the data from it, including a GUI interface and data streaming and collection tool in different language platforms, including MATLAB, Python and so on. For the data streaming protocols, it also provides some choices like OSC [8], LSL [9], UDP, or Serial. In our project, I used LSL for data streaming for the sake of ease in the integration to our own system in a normalized format.

There are some recent experiments exist in this area, and here I will introduce some important works. In 2013, LaFleur et al. [10] reported that by using BCI2000 and Neuroscan set, they achieved around 80% accuracy of using EEG signals to control over quadcopters to hit targets. The experiment setting up is that, by analyzing mainly C3, C4s information, and by training the experimenters to first control over 1-D motor imagery, then 2-D, finally 3-D. The headset they used is consisted of 64 channels, and was sampled at 1000 Hz, then filtered by Neuroscan for 200 Hz. One credit for this work is that it achieved relatively high accuracy, but compared to our system, this work has 4 times more electrodes than us, and also pretrain the experimenters for the experiments. Another work notable is that Liu et al., in 2016, explored about whether the movie is good or not by using EEG signals and validated by eye trackers [11]. This project has an impact because that it used multiple data streams for data processing, and is a new way of knowing how the data will work. In our project, I plan to use more data sources in future research, as video data, body sensors, and eye trackers, but in current step is only for EEG signals. Ang et al. published a paper in 2017 addressed about the application of EEG in helping the disabled for rehabilitation [12]. The experiments compared the usage of EEG in brainwaves from a normal person or a person from disability, and discussed how I can apply EEG controlled equipment to help

people with disability. And one recent publication is directly related with our project, which is based on OpenBCI [13]. This paper explored about using low cost EEG devices to control over robot, and achieved about 55% accuracy for motor imagery and 70% for movement assisted motor imagery. It is a good start although the accuracy is not good, but after more precise processing, and better extracted features, I achieved much better accuracy then the experiments of them, and achieved three class classification, where for them, the classification is only for two labels. I will further explain my system in the following chapters.

1.2 System Framework

The framework is shown in Figure 1.1. More detailed framework is shown in Figure 1.2 for offline system, and Figure 1.3 for online system.

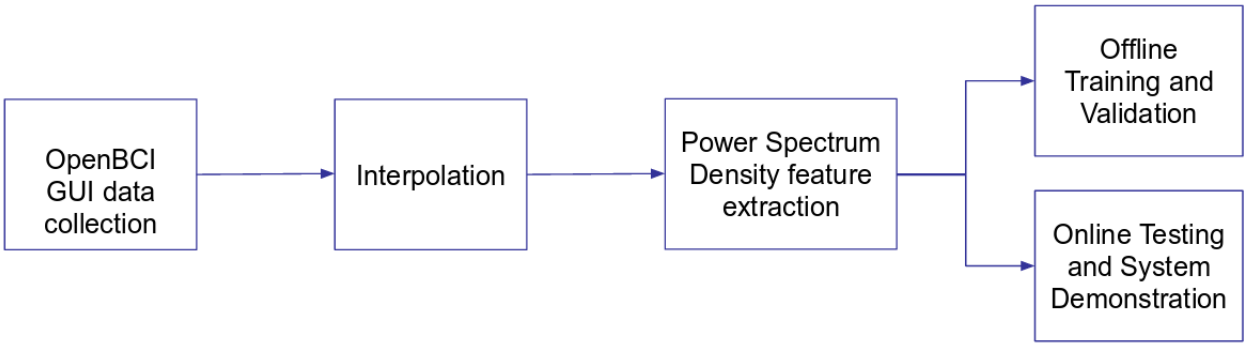

Figure 1.1: Data processing work flow

As shown here, the general workflow of the project will be divided into four parts. The first part will be the raw data collection part. It is fulfilled by the GUI that is provided by OpenBCI. OpenBCI provides access to data for up to 16 channels under the standard of 10-20 system [1], shown in Figure 1.4. The channels I am using are Fp1, Fp2, C3, C4, T5, T6, O1, O2, F7, F8, F3, F4, T3, T4, P3, P4, and references are put on ears, which are A1 and A2 on Figure 1.4. After getting the data from the GUI, it will be not processed, so the next step is to read the data into MATLAB. In MATLAB, the data was read from a txt

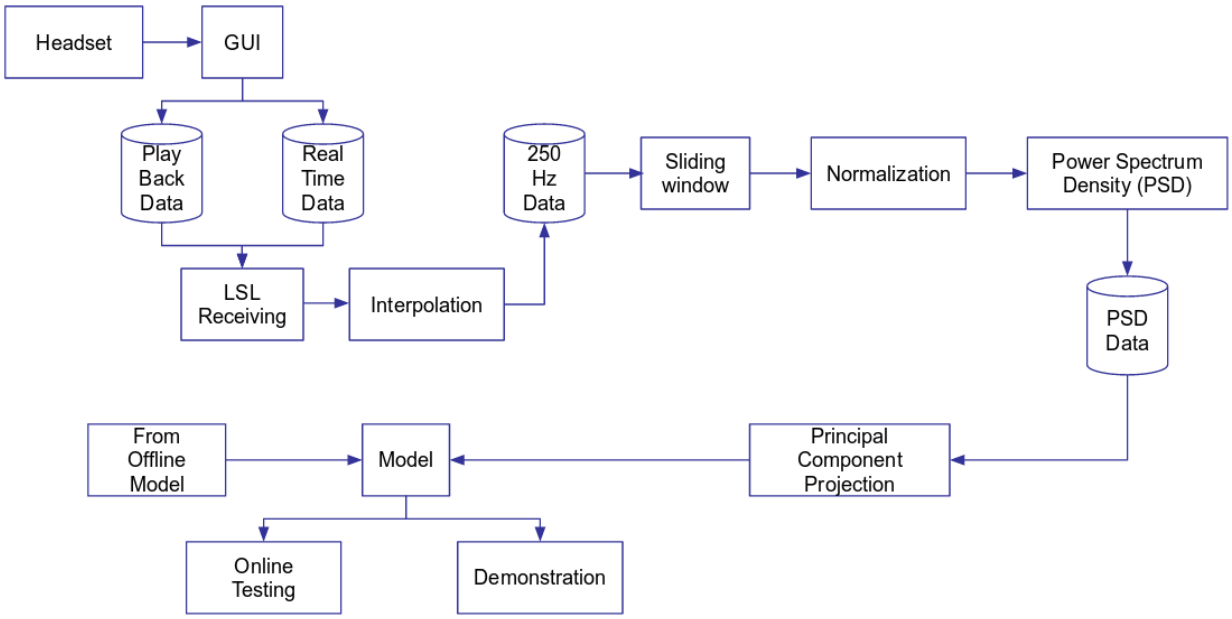

Figure 1.2: Offline system flow chart

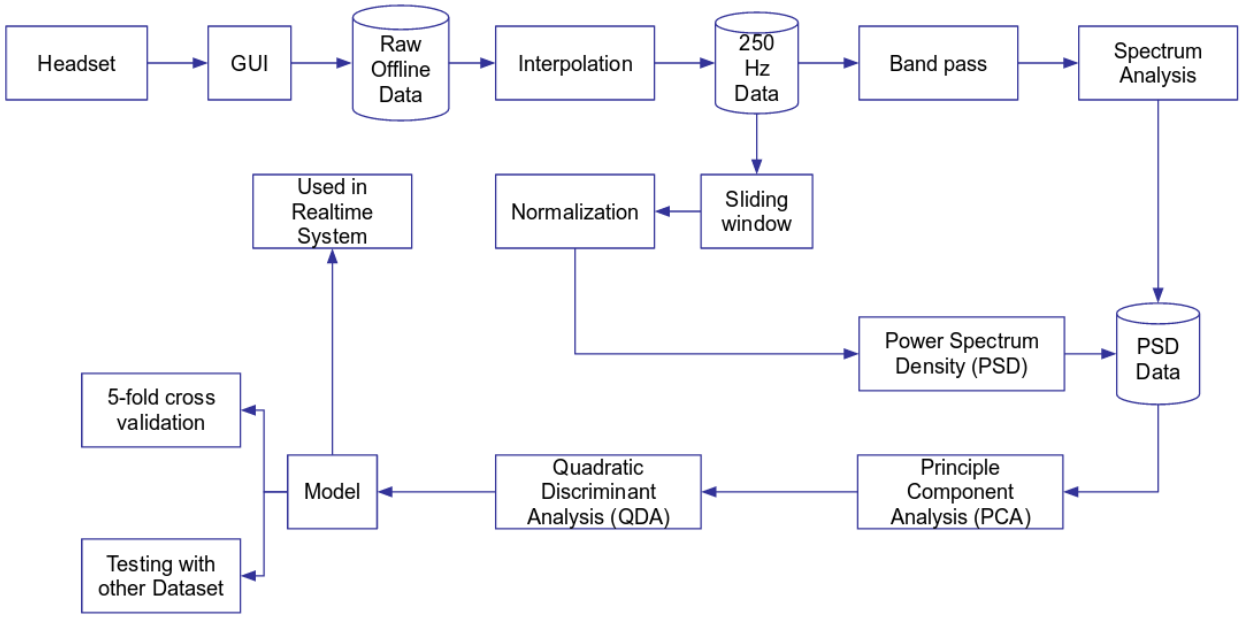

Figure 1.3: Online system flow chart

file that stored from OpenBCI GUI, then it will be interpolated to 1000 Hz. The reason to interpolate it is that because of the instability of power supply or RF signal transmission, the timing of each samples is different. For this reason, I interpolated the samples to the resolution of timing points, which is 1ms, then the frequency will become 1000 Hz. After getting the 1000 Hz data, I need to first pass it through a low pass filter to get the signal back to the normal frequency, which is 250 Hz, then down sample it to 250 Hz. That is the preprocessing part of the data. Then after a spectral analysis, I decide to use some specific bands from the brainwaves to calculate the total power of the band, then use the power as features of training for models. Before feeding the data into the training, I employed PCA [14] to get the first major components, and found that the data is not linearly separable, but I can see some major difference between labels.

In training sessions, I applied QDA as the classifier. And in the evaluation session, to avoid overfitting problem, I applied 5 folds cross-validation to see the classification accuracy. Also, because one major aim of this system is to use the EEG signals I collected to so real-time robot controlling, I also tested on a new dataset. To make sure that the new data was collected at the same condition of the data that trained the model, I required that the experiment subject kept the headset on between the new experiment and the already used dataset. The new dataset test result also showed high accuracy for classification when the training data is not contaminated, but showed low accuracy especially for left and right classification in contaminated situations. Furthermore, I tested the model by online send the data to the classifier, and see the accuracy of the online classification, and the project finally landed on real-time robot controlling. One instructor will give verbal signals to the experiment subject, and the subject will move left or right hand according to the instruction, then the EEG signals will be collected, processed and the attributes will be projected to the selected principle components, then the projected features will be passed through the trained models, make decisions, then control over the robots, which are the LEGO EV3 robot. It has

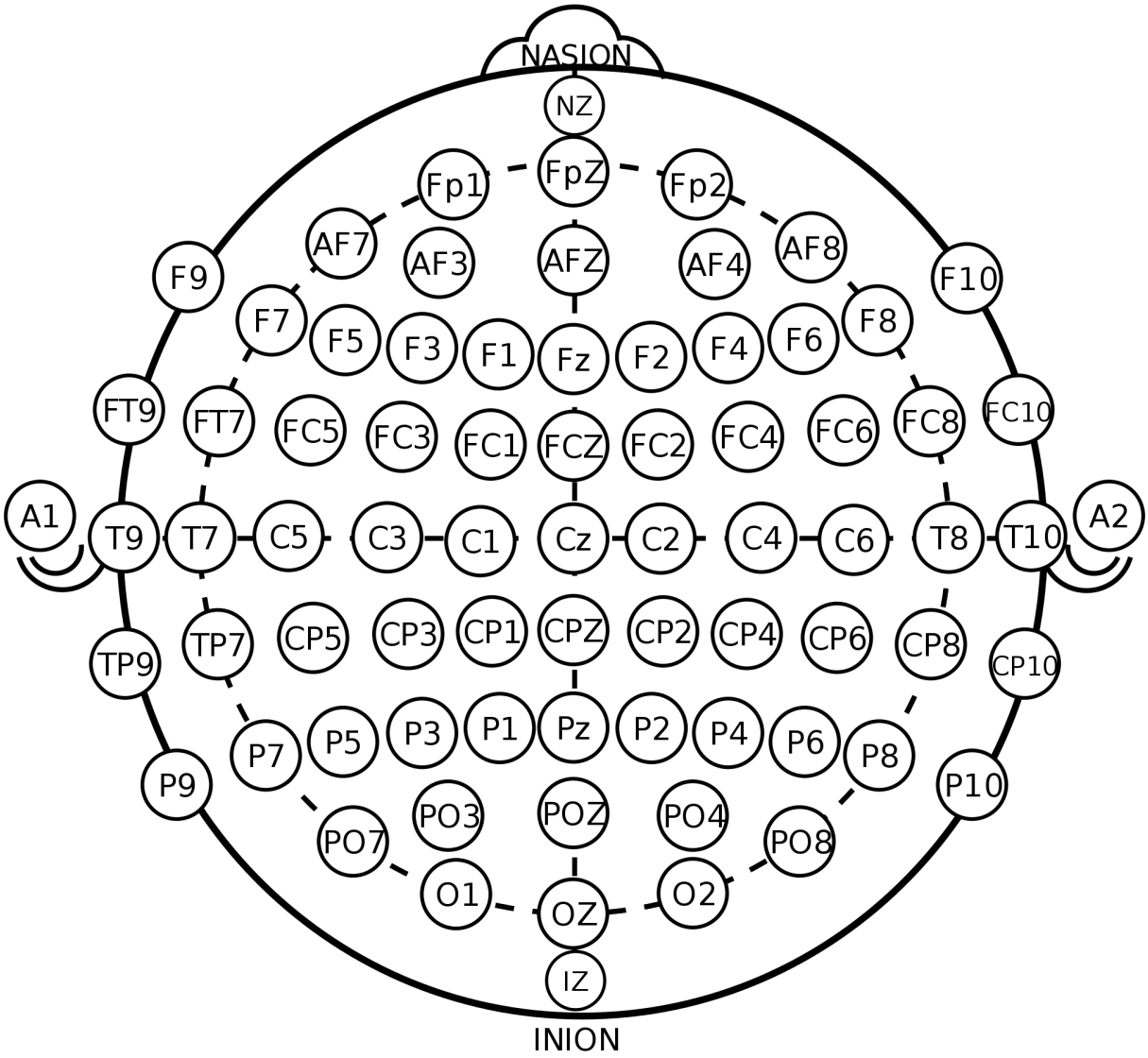

Figure 1.4: 10-20 system electrodes placement [1]

two motors and the brick body is mounted with an Linux system. In the system, a Python package is provided to control the robots motor, and USB wireless network dongle can be plugged into the body, so that it can be connected via wireless network connections.

This thesis focuses on achieving a system for using EEG signals to classify left or right signals with physical movement assistance. In Chapter 2 I will propose a data-driven signal

processing framework to collect and interpolate the data. In Chapter 3, the feature extraction tool, and some more process will be introduced. I will explain in detail that the reason I choose PSD by spectral analysis. And in Chapter 4 I will discuss about the features and the result of PCA components and the machine learning tool I applied, which is QDA. An introduction to Linear Discrimination Analysis (LDA) will also be provided since it is the base technology of QDA. Chapter 5 will be the details about deployment and experiment design. Experiment results will be discussed about in Chapter 6. Finally, future work will be discussed in Chapter 7.

Chapter 2

Data Collection and Preprocessing

2.1 Data Collection

The data collection part was done by OpenBCI GUI. This GUI can be attained from the official website from OpenBCI, and the program language is Java. I chose the version that can run the applications stable, and installed LSL package into the GUI packages. LSL is a standard package that is widely used for sending EEG data from applications to each other. In our application, it gets the data from OpenBCI GUI, and sends the data to MATLAB, and our program will accept the data by batch, with LSL in MATLAB version. For offline data acquisition, the OpenBCI GUI will directly save the data to a local txt file. Incomplete lines will be manually trimmed, and thus the file can be read into MATLAB for processing for next steps.

Following will be a short introduction of the OpenBCI GUI. The GUI is shown in Figure 2.1. It mainly is composed of three windows components and some controlling buttons. The System Control Panel is used for the system settings, it can run in three modes, live, playback, and synthetic. The modes are specified by a menu in the system control panel.

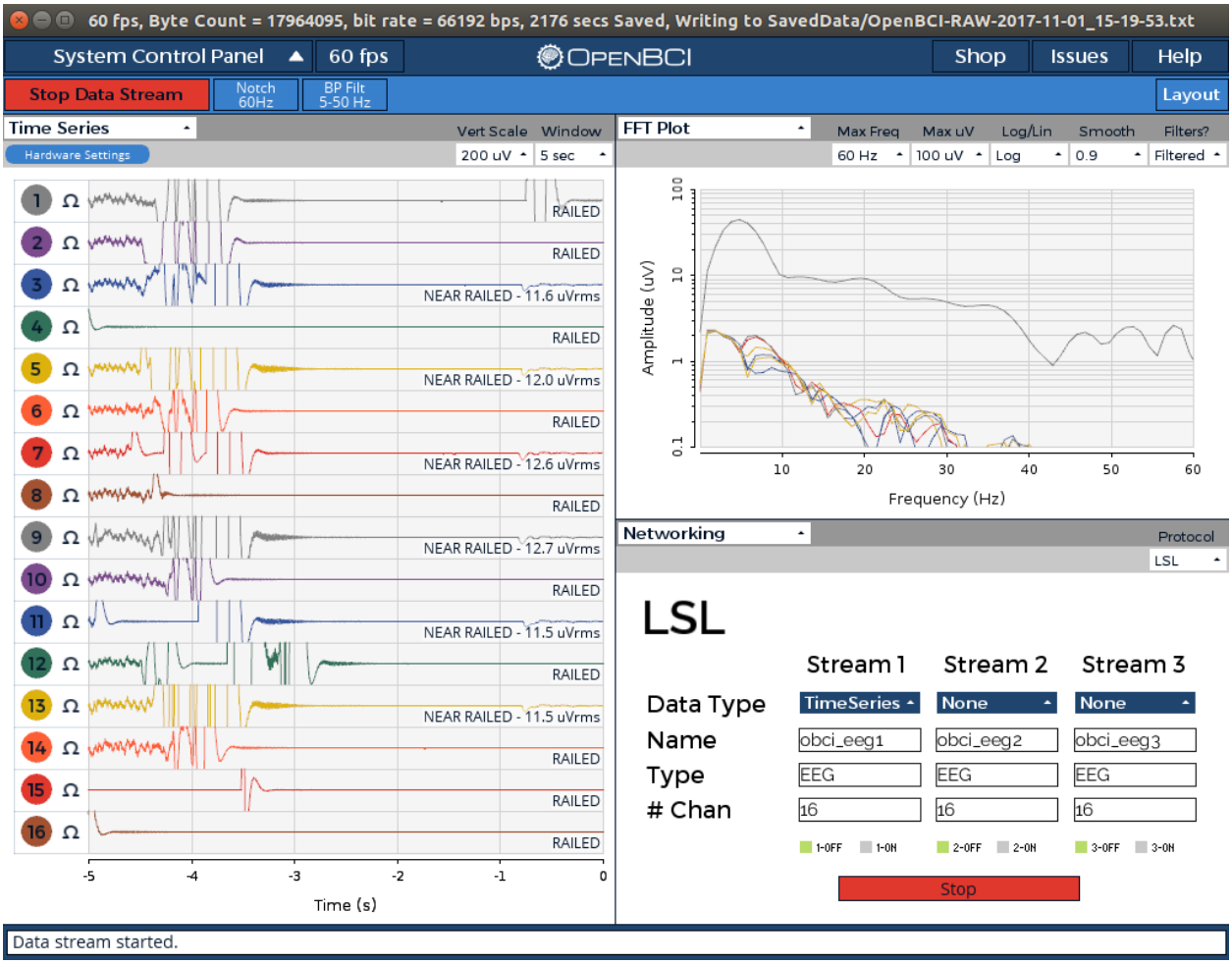

Figure 2.1: OpenBCI GUI, Left panel is the real-time wave visualization, right two panels are FFT plot of each channel data, and the network protocol panel.

In our experiment, only live mode was used. And the port is also specified in the system control panel to direct the program with the USB port mounted with the RF dongle.

For the three windows components, users can self-specify which applications are needed to be shown there. Among the applications, the most important ones are Time Series, FFT Plots, and Network windows. The time series shows each of the sixteen channels real-time data in the timing windows, and users can specify the length of the windows. The real-time data is processed, but not exported to the txt file. The processing includes a notch filter, and

a band pass filter. Generally in experiments, 60 Hz will be selected for the notch because it is the industrial frequency and will pose more environment noises. And usually useful brainwave will be in the range of 0.5 to 50 Hz, thus the band pass used this band range. The FFT Plot shows the 16 channels FFT result. It ensures the experimenters to see the real time FFT result of brainwaves in different channels. The Networking window is used for sending the data to other applications through protocols like UDP, LSL, etc. I use LSL to send the data to MATLAB in this window.

2.2 Data Interpolation

Channel 14	Channel 15	Channel 16	TimeStamp
-63050.78	93749.98	-73966.14	16:28:48.543
-126658.41	187500	-148705.25	16:28:48.547
-126658.41	187500	-148705.25	16:28:48.552
-126681.75	187500	-148731.78	16:28:48.555
-126681.75	187500	-148731.78	16:28:48.558
-126671	187500	-148722.8	16:28:48.562
-126671	187500	-148722.8	16:28:48.571
-126679.47	187500	-148719.41	16:28:48.571
-126679.47	187500	-148719.41	16:28:48.580
-126660.49	187500	-148716.09	16:28:48.580
-126660.49	187500	-148716.09	16:28:48.583
-126672.2	187500	-148718.83	16:28:48.585
-126672.2	187500	-148718.83	16:28:48.591
-126652.13	187500	-148717.84	16:28:48.596

Figure 2.2: Samples saved from OpenBCI GUI

After getting the data, the OpenBCI GUI will automatically save the data to a txt file, and I will read the file through MATLAB after manually trimming out the incomplete last line of recording. The data I read out will be formatted in the following way: index, 16

channels data values, 3 auxiliary values, timing points. Shown in Figure 2.2 is a sample of the data that I retrieved, and I can see in the red squared areas are the timing points. The timing interval is not the save over several timing samples, and the sample rate is 250 Hz. Due to the unstable data sampling rate, I need to interpolate the data, then subsample to get back to the correct sample rate.

I calculated the interpolation in this way. Due to the communication overdue, there will be some duplicated data samples in the raw dataset. First, I got the unique values of the dataset so that the duplicated samples were removed. Then I calculated the total time interval, then calculated the total milliseconds within the interval, then used the linear interpolation to fill the gaps between raw data until each millisecond is correspond to a sample.

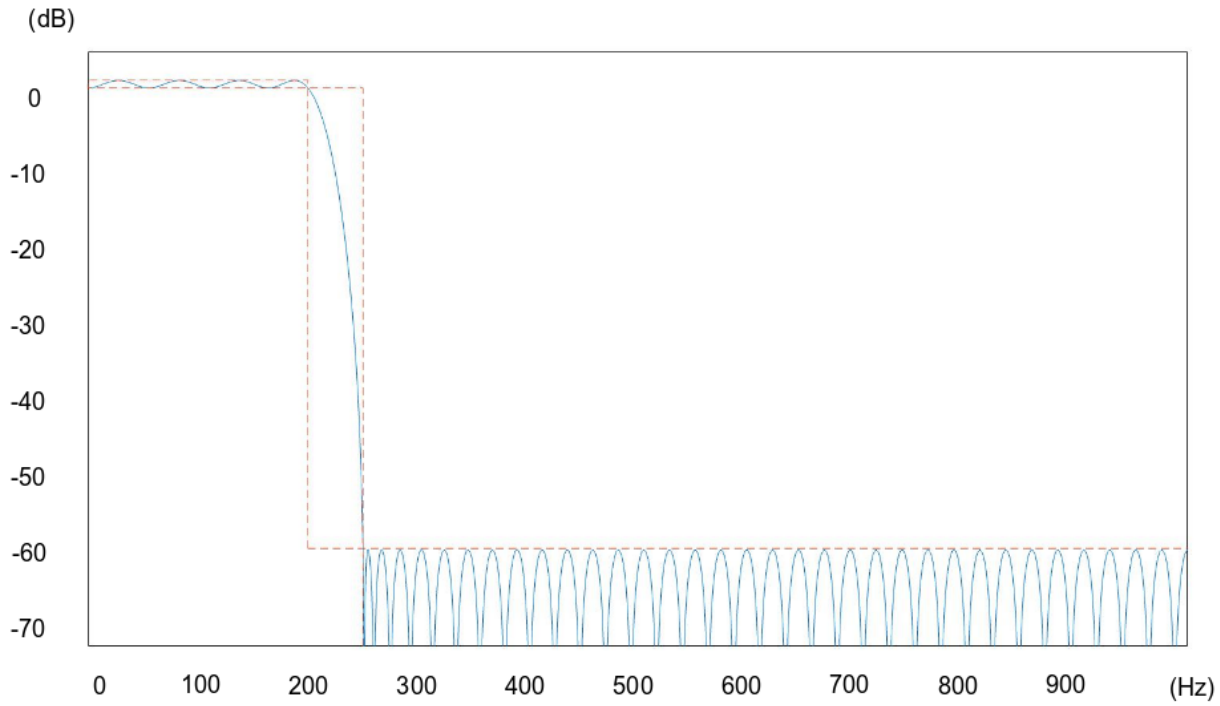

Figure 2.3: Low pass Equiripple filter

The next step, intuitively, will be subsampling. However, if I directly subsample the data, it will have an effect of aliasing. Aliasing [15] happens when high frequency signals aliased

with low frequency signals. After the interpolation, the signal is already 1000 Hz. To avoid the aliasing of signals over 250 Hz, which will be the noise since the sampling rate is 250 Hz, I need to pass the signals through a low pass Equiripple filter [16]. The filter is visualized as Figure 2.3.

I designed this filter as follows. The starting point to filter is 200 Hz, and the stopping frequency is 250 Hz. This is for avoiding Gibbs effect [17]. The low pass area could have a vibration of 1 Hz. After the low pass filter, the data will be sampled into 250 Hz, in accordance to original sample rate.

After the interpolation step preprocessing , my data was further explored and analyzed with spectral analysis and feature extraction, which will be discussed in detail in Chapter 3.

Chapter 3

Analysis of Data and Feature Extraction

3.1 Brain wave bands

After finishing the data preprocessing steps, the next step will be spectral analysis. Details about spectral analysis will be introduced in Section 3.2. According to existing researches [18] [19], there are several main frequency bands for human brain waves. They may differ from different person, and from different time that the signals are collected, but generally the brain waves are composed of some main bands. The bands are mainly composed of these parts:

1. Delta waves (0.5 to 3 Hz): Delta brainwaves are generally slow and loud, and usually is related with deep meditation and sleep without dreams. This wave band is also related with healing process. In our experiments, even I have band pass filters, some of environmental noises will still exist in this frequency level. Also, because this wave band is not related with our experiment, since our experiment is mainly related with movements and attentions. This band is abandoned in our feature extraction process.

2. Theta waves (3 to 8 Hz): Theta waves often happen in sleep process, but also related with meditation. In this wave band, people use it to learn, remember. Intuition also happens in this wave band. Since our project is related with attention and may have effect on learn, remember, I put this frequency band into consideration.

3. Alpha waves (8 to 12 Hz): Alpha wave is important in our experiment, since it is mainly related with quick and quiet flowy thoughts. It mainly is the current thought, and our experiment requires the subjects to have some reactions to figures in real time, this wave is responsible for the reaction.

4. Beta waves (12 to 38 Hz): Beta wave is the most important band frequency in our feature extraction. According to existed research, this wave band dominates our normal working state. For example, when subjects attention is directed by figures shown in our program, this waveband will be strong to process about this cognitive task, and this process is fast. It is also related with decision making. There is already some existed research related with beta wave, and this frequency band can be generally divided into three bands: Low Beta, Beta, and high Beta. The higher frequency, the more complex the task is. Considering the complexity of our tasks, higher frequency Beta wave bands are not needed in our feature extraction.

5. Gamma waves (38 to 42 Hz): This wave band is mainly related with virtues, and emotion. Emotion is not included into consideration in our experiment for now.

To sum up, the frequency bands I am mainly focusing on is around 3 to 30 Hz. In this frequency band, most of the features should be represented for the experiments I designed, which contains attention, simple information processing, and physical movements.

3.2 Spectral Analysis

To better understand the frequencies range that we need to consider about, I applied a band pass filter from 3 Hz to 40 Hz to filter out frequency components that are likely to be noises, and employed Fast Fourier Transform (FFT) for spectral analysis. Only real part of the analysis is shown in Figure 3.1 and Figure 3.2, respectively spectral analysis for brainwaves in task status, and in rest status from the same dataset, same channel. After transforming the signals into frequency domain, in order to see the tendency at a higher level, the signals are smoothed by calculation the average of each consecutive 100 points. This process will not harm the general tendency of the amplitudes since the resolution of the signals is less than 0.005 Hz. The calculation of the frequency resolution will be further explained in section 3.4.

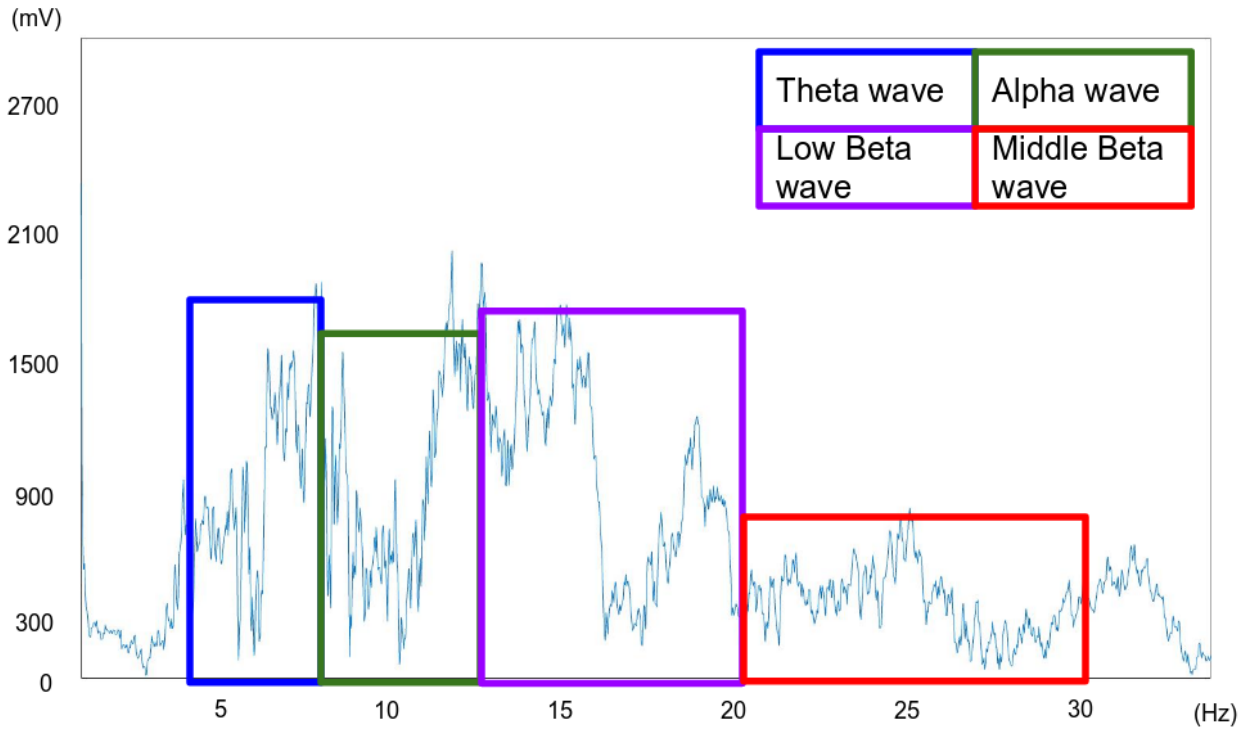

Figure 3.1: Spectral analysis for task status from channel Fp1. Each of the tasks are stacked together according to the label, and applied band pass filter of 3 Hz to 40 Hz to the stacked samples.

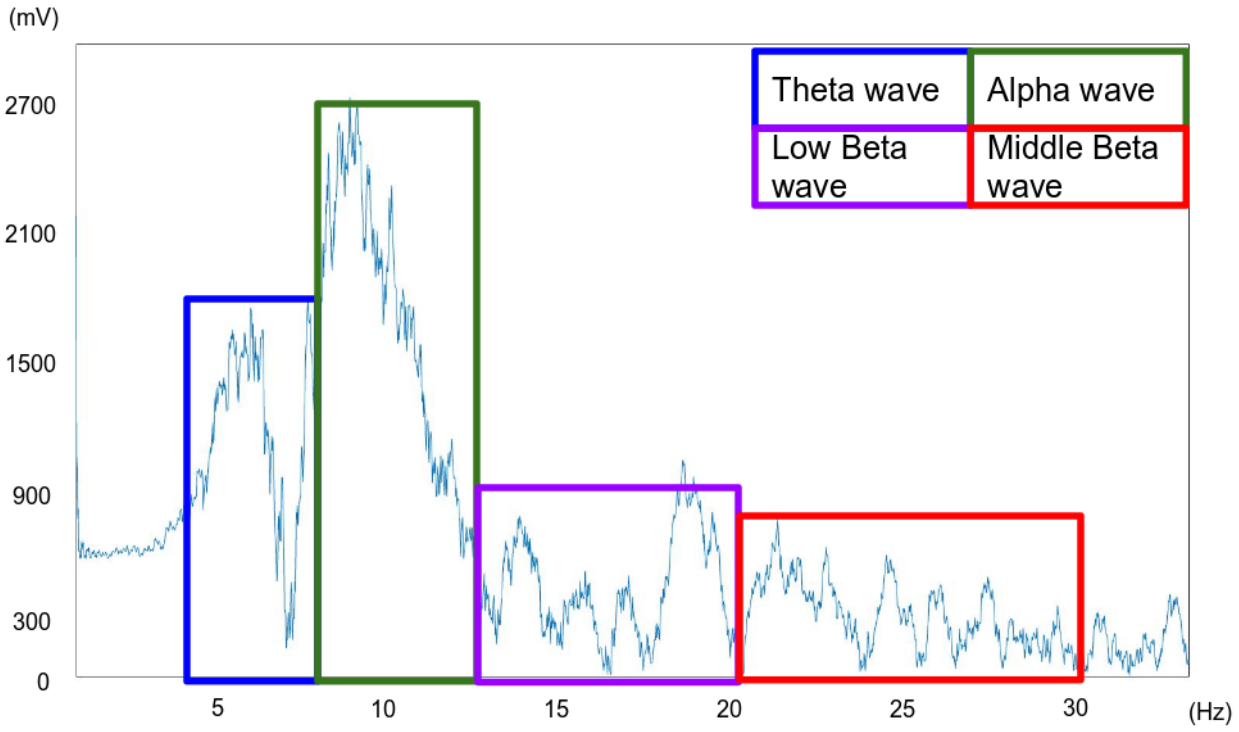

Figure 3.2: Spectral analysis for rest status from channel Fp1. Each of the rest status are stacked together according to the label, and applied band pass filter of 3 Hz to 40 Hz to the stacked samples to keep accordance to the analysis with task status.

From the spectral analysis, we can clearly see some general features from both status, and some brainwave bands that I need to use as features. Even passed by a band pass filter, we can see that the noise is still strong in low frequency area, until it goes to around 4.5 Hz. And there is a clear cut at around 8 Hz in both figures. This area is exactly the frequency band of Theta wave from the existed research. Another wave band can be clearly seen from 8 Hz to around 12 Hz. For rest status the cut is clearer. It can be recognized as the Alpha wave. And for 12 Hz to around 20 Hz, both figures have a frequency cut. I consider that area is the low Beta wave, and for 20 Hz to 30 Hz, we can see some other information is shown but in a relatively low amplitude. I refer this band as the middle range Beta wave. In

total, four bands are chosen as the features for the next step of training. One way to express these features is PSD.

3.3 Power Spectral Density

PSD [20] is a feature frequently used to represent the power distribution through the whole frequency bands. To calculate this feature, we need to first have a timing window, and then use FFT to get frequency domain information for the selected timing window of samples. The result of FFT will be in the form of complex numbers, and the calculation of the PSD for a single frequency point can be explained in Equation 3.1.

$$P(f) = |x(f)|^2 \tag{3.1}$$

In this equation, $P(f)$ is the power of the frequency f, and $x(f)$ is the amplitude of frequency f, acquired by using FFT to transform the time domain information into frequency domain.

After getting the power density of specific frequencies, I need to calculate the total power of a specific band. In our project, frequency range from 4.5 to 8 Hz, 8 to 12 Hz, 12 to 20 Hz, 20 to 30 Hz are calculated. Each of the bands are calculated in Equation 3.2.

$$P(k) = \sum_{i=l_k}^{h_k} (|x(f_i)|^2) \tag{3.2}$$

For each band k, it has a low band l_k and a high band h_k. I sum the power of $x(f)$ density up in the band range l_k to h_k. The result of this step will be the total power over the selected bands. Each of the 16 channels will provide these four attributes respectively, so together the dimension of training is 64.

For the calculation of PSD, I need to use a sliding window to select a series of continuous samples. The selection of time windows length becomes important in this task, and some more process is needed for a more accurate calculation.

3.4 Sliding Window and Normalization

There are some places that I need to pay attention to when I process the data from OpenBCI headset. These mainly includes the selection of length of sliding window and the normalization of input data.

3.4.1 Selection of Sliding Window

The selection of sliding window is mainly related with the frequency resolution and system delay. Because of the nature of FFT, the result of the transformation will be of the resolution specified in Equation 3.3

$$R(x) = \frac{F_S(x)}{N} \tag{3.3}$$

In Equation 3.3, R is the resolution of the selected samples x, and $F_S(x)$ is the frequency of the selected samples. In our experiment, the value is 250 Hz. N is the total count of the selected samples.

We can analyze from Equation 3.3 that the more samples I include in a sliding window, the smaller the resolution of the samples will be. Thus, our next step of calculation of PSD will have more frequencies. For example, if I choose the sliding window size $N = 64$, the resolution will become around 3.9 Hz. But as previously analyzed, I will use the frequency bands of 4.5 to 8 Hz, and 8 to 12 Hz. The resolution of 3.9 Hz cannot represent enough

information of the whole band since in the range I need to use, only one or two values of PSD are calculated.

On the other hand, a length of sliding window cannot be too long either. There are mainly two reasons about it. First, if we choose a too long sliding window, the switch of the status will be unclear and multiple status can be calculated within the same sliding window. The second reason is that if we take too long sliding window, the calculation time will be long and thus in our real-time experiment, the delay of the system will be long.

3.4.2 Normalization of Input Data

We can first look at the input of the system. Figure 3.3 is the data after preprocessing from channel Fp1.

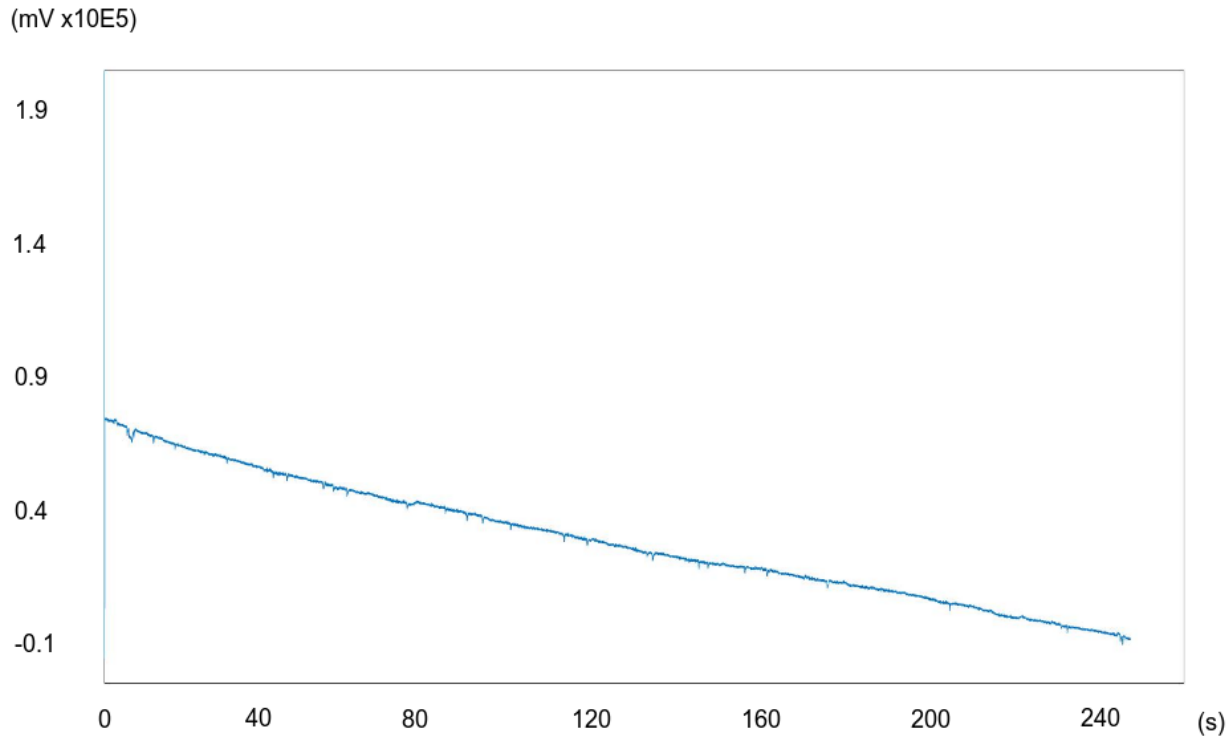

Figure 3.3: Input data for PSD from channel Fp1

From Figure 3.3 we can see that after the preprocessing, the data is drifting, and our data processing approach is focusing on the data within the time window. This means that when we use the absolute value of the amplitudes, the data will be affected by the global drifting. One way to solve this problem will be normalization of the amplitudes. Equation 3.4 shows the detail about the normalization.

$$\hat{P}(f) = \frac{P(f)}{\sum_{i=l}^{h}(P(f_i))} \tag{3.4}$$

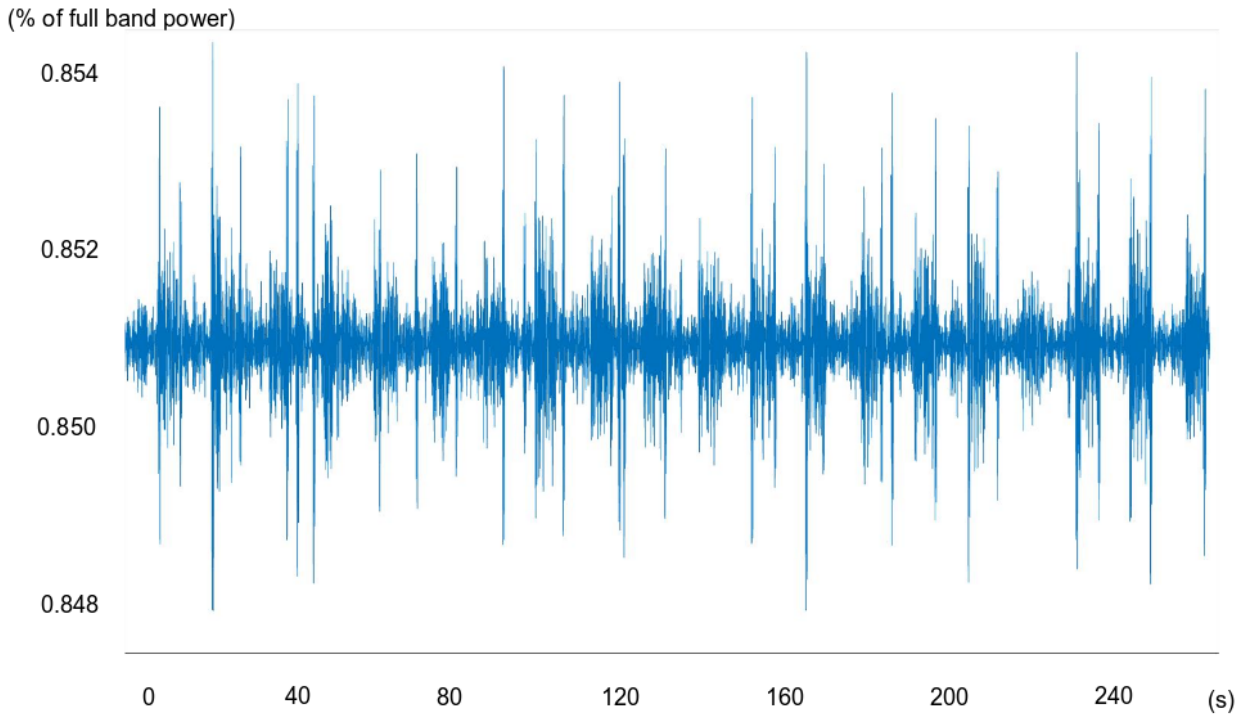

Figure 3.4: Normalized attribute for the Theta wave from channel Fp1

In equation 3.4, $\hat{P}(f)$ is the normalized PSD of frequency f, and $P(f)$ is the absolute value of original PSD calculated from Equation 3.1. It is divided by the sum of the power through the whole band, and the PSD of the frequency is represented by the portion of the power it has through the whole band in this way. Since this normalization process is applied to each timing window, the effect of global drifting is eliminated. Shown in Figure

3.4 is the result of getting the normalized PSD as the attribute. We can see that the data is normalized by this step. And the local noise and drifting is filtered out by a band pass filter mentioned in section 3.2.

The next step will be an analysis of the features by PCA. More details of selection of classifiers, the way the classifiers work will be discussed in Chapter 4.

Chapter 4

PCA Based Feature Analysis and Classification

4.1 PCA Feature Analysis

PCA is one major technique for analyzing the features of training samples. One main aim of PCA is to separate the information discriminate more out to reduce the dimension of training. By extracting the dimensions representing more information, dimensions representing less information or noisy information are abandoned and thus the training will be more accurate, and less overfitting issue will happen.

The process of PCA is to get some major components, that can best represent the dataset. It is an orthogonal transformation to a new coordinate system, and a projection to the major components. For example, the original coordinate system might be not efficient in representing the information efficiently, we can find a major component that the data varies more in some other specific coordinates, and PCA is to find the coordinate to better represent the data on the biggest variance.

General steps of PCA include calculation of covariance matrix, eigenvalue decomposition, analysis of eigenvectors with large eigenvalues. These steps will be introduced in detail in this section.

4.1.1 Calculation of Covariance Matrix

The first step of PCA is to find the covariance matrix. Suppose $P_{c,k}(n)$ is the input data, the dimension of P is $n \times (c \times k)$, which represents n samples of $(c \times k)$ dimensions. Specifically, in our experiment, $c = 16$ and $k = 4$, so in total I have the training data dimension 64.

Covariance matrix is the representation of variance for each dimension. The way of calculating covariance is shown in Equation 4.1.

$$C = P^T P \tag{4.1}$$

C is the covariance matrix, it shows the covariance of two dimensions. For example, for each column i and row j, C_ij is the covariance of the attribute i and attribute j. When $i = j$, covariance is the variance of the attribute. The next step is eigen-decomposition or singular value decomposition of C.

4.1.2 Eigenvalue Decomposition

For the covariance matrix, since it is a square matrix and it is diagonalizable. I use eigenvalue decomposition to represent the matrix C.

$$C = V^{-1} \Lambda V \tag{4.2}$$

In Equation 4.2, V is the eigenvector matrix, and each column is one eigenvector. V is also called eigen-space in some situations. Λ is a diagonal matrix that only the diagonal of the matrix are non-zero values. These values are called eigenvalues, and it is arranged in

the order of from bigger values to smaller ones. Each eigenvalue Λ is corresponding to an eigenvector in V. For example, suppose λ_i is the i-th value in Λ and v_i is the i-th value in V, λ_i is the eigenvalue that corresponding to the eigenvector v_i.

The way to calculate about the eigenvalues and eigenvectors is shown in the next equations. The reason that I want to calculate the eigenvalue of a matrix is that we want to use a diagonal matrix to represent a matrix, so that Equation 4.3 will hold.

$$Cv_i = \lambda_i v_i \tag{4.3}$$

In this equation, we want to use a set of eigenvectors and eigenvalues to represent the covariance matrix. In order to calculate the set of eigenvectors, we need to calculate in the Equation 4.4 and Equation 4.5.

$$(C - \lambda_i I)v_i = 0 \tag{4.4}$$

$$det(C - \lambda_i I) = 0 \tag{4.5}$$

After solving Equation 4.5, eigenvalues λ_i is calculated, and according to Equation 4.4, v_i can be thus calculated.

4.1.3 Analysis of Features with Large Eigenvalues

One of the main aim of PCA is to get the eigenvectors of large eigenvalues, and after getting the eigenvectors, the dataset is projected onto the eigenvectors, and I plotted the scatter plot of the projected datasets to see the distribution of the datasets.

For each eigenvector, Equation 4.6 is applied to calculate the projection, and the projections refer to the data distribution on the specific directions of eigenvalues.

$$P' = Pv_i \tag{4.6}$$

After the projection, we can see the distribution of the data, shown in Figure 4.1 and Figure 4.2. The two figures are from the same dataset, and each different color corresponds to different labels. In the figures, blue is the rest status, and yellow is when the task is right, cyan is when the task is left. X axis and Y axis are both variance.

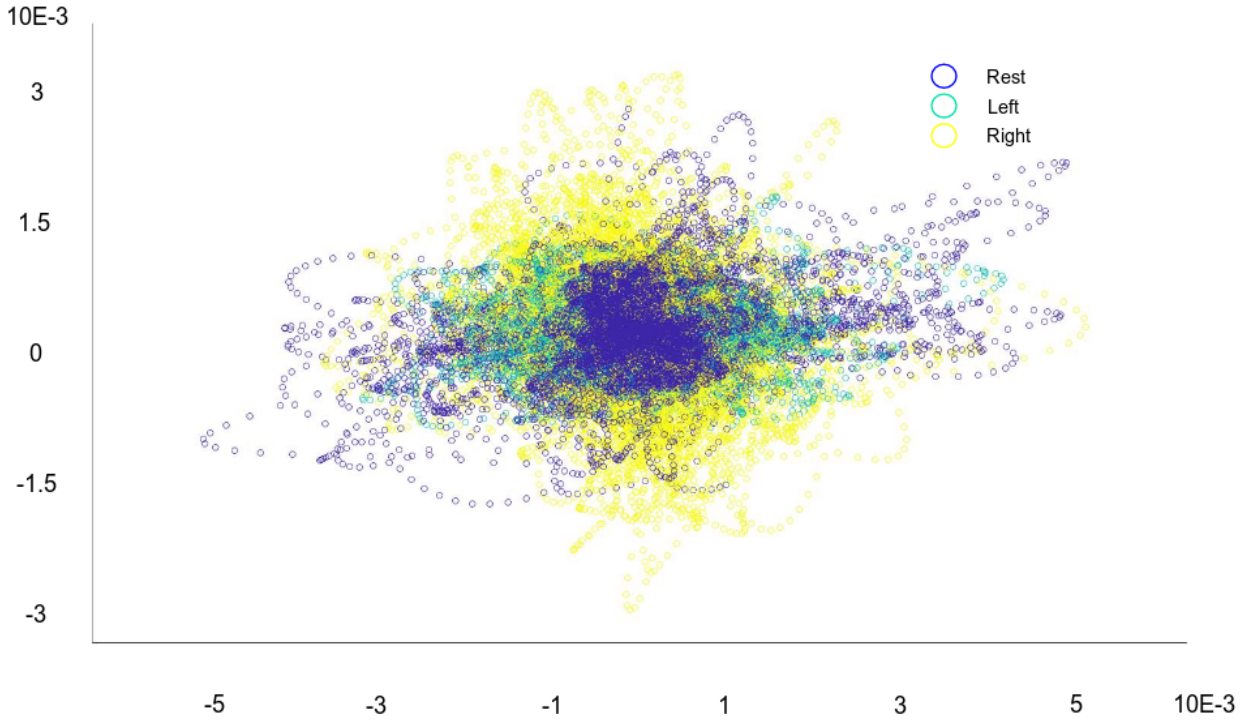

Figure 4.1: Projected data distribution for first and second major components

From Figure 4.1 and Figure 4.2 I can draw mainly two conclusions. First, the labels are not linearly separable, so that I need to find a non-linear model for the training and testing to get a reasonable result. Second, I can also see that some general conclusions can be made that usually in the rest status, the variance will be low since the movement is small and the experiment subject concentrates much on the screen. And for the left and right tasks, especially left task in Figure 18 I can see that the data labeled with left has

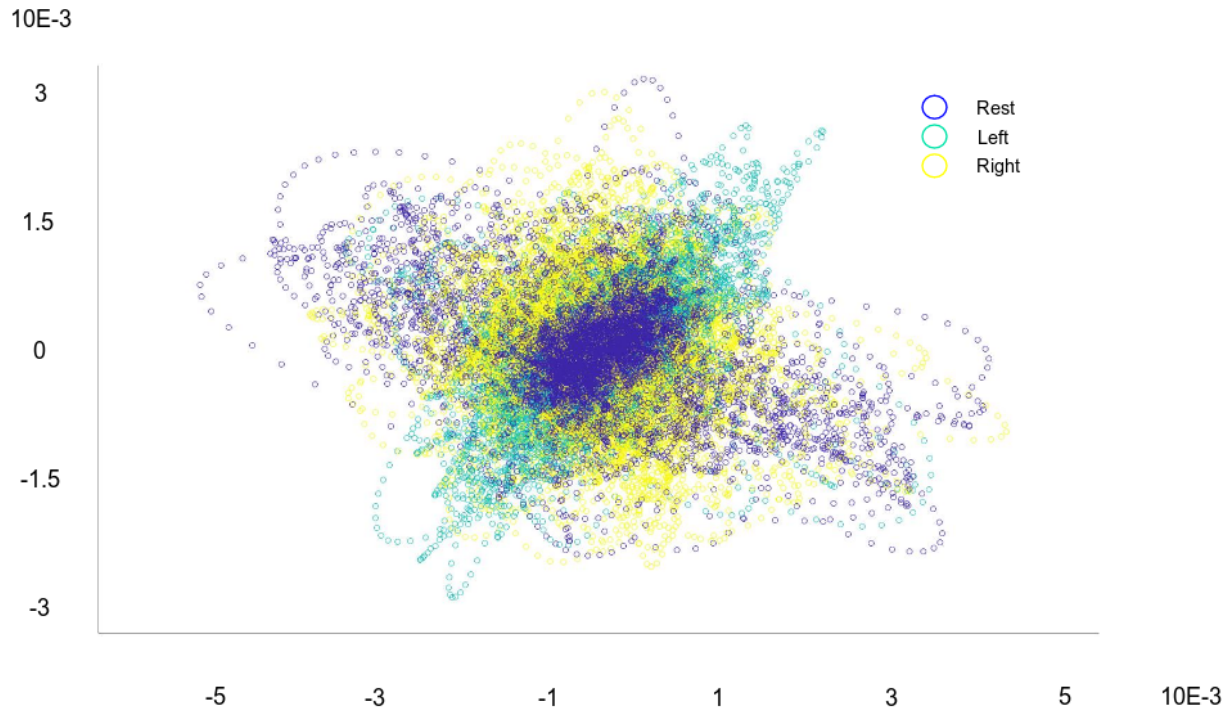

Figure 4.2: Projected data distribution for first and third major components

a tendency of distributing on different dimensions than the right ones. Right ones have a similar distribution to the rest ones, but the variance will be much higher except some noises. Due to the two conclusions, I decided to use Quadratic Discriminate Analysis as the model for the classification of tasks.

4.2 Quadratic Discriminate Analysis

This section will be composed of two main subsections, LDA [21] and QDA [22]. LDA is not applied to this project but still introduced because the two models are closely related and QDA is stepped from LDAs main idea.

4.2.1 Linear Discriminate Analysis

Let's assume that each of the labels is under Gaussian Distribution, and {rest, left, right} are labeled as 0, 1, 2 for the simplicity of representation: $p(l = 0) \sim N(\mu_0, \sigma_0)$, $p(l = 1) \sim N(\mu_1, \sigma_1)$, $p(l = 2) \sim N(\mu_2, \sigma_2)$. The Bayes optimal solution [23] to predict points for two label classifications is shown in Equation 4.7.

$$(x - \mu_0)^T \sigma_o^{-1}(x - \mu_0) + ln|\mu_0| - (x - \mu_1)^T \sigma_1^{-1}(x - \mu_1) + ln|\sigma_1| > T \tag{4.7}$$

This means that if the result of the Bayes optimal solution is bigger than some threshold T, then it will be classified as the second class, otherwise it will be classified as the first one. LDA assumes that $\sigma_0 = \sigma_1 = \sigma$, and for the threshold constant c, Equation 4.8 holds.

$$c = (T - \mu_0^T \sigma_0 \mu_0 + \mu_1^T \sigma_1 \mu_1)/2 \tag{4.8}$$

And Equation 4.8 satisfies Equation 4.9.

$$\sigma^{-1}(\mu_1 - \mu_0) > c \tag{4.9}$$

For multi-class situation, σ is calculated in Equation 4.10

$$\sigma_b = \frac{1}{C} \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T \tag{4.10}$$

After calculation of σ_b, we can get the separation in Equation 4.11

$$S = \frac{w^T \sigma_b w}{w^T \sigma w} \tag{4.11}$$

In 4.11 w is the separation on a certain direction. For example, if the separation direction is the separation of Left and Right, which in our notation is 1 and 2, the definition of w will be in Equation 4.12.

$$w = \sigma^{-1}(\mu_2 - \mu_1) \tag{4.12}$$

This is the implementation of LDA. QDA is based on LDA and will be described in the next subsection.

4.2.2 Quadratic Discriminate Analysis

The main difference from QDA to LDA is that LDA assumes that the covariance of each classes is the same, while QDA doesnt have the assumption. This will usually result in that the classification surface will become a quadratic surface rather than a linear one, which is the surface of LDA. Finally, the classification from QDA is shown in Equation 4.13.

$$d_l = (x - \mu_l)^T \sigma_l^{-1}(x - \mu_l) + ln|\sigma_l| \tag{4.13}$$

d_l is the discriminate score, and the higher d_l is for a specific label l, the more possible that the class is from the label l, and thus the input x will be classified as l.

In the experiment, I used QDA for the classification as the result of PCA analysis. QDA can also be applied as a kernel of other linear classification methods, which will be a good topic to work on with. But a major issue for non-linear problems is the computation time.

In Chapter 5, I will further introduce about my experiment design and some data processing techniques that make our input data less noisy.

Chapter 5

Deployment and Experiment Design

5.1 Hardware Preparation

The hardware set was purchased from OpenBCI. It contains 16 electrodes to detect the voltage on the scalp, and two ear clips for referencing. The electrodes are designed differently due to the accuracy of the signals. If the electrodes cannot attach to scalp closely, the signal will be noisy or even not available. Figure 5.1 is example of the two kinds of electrode units. For the ear clip, since I only have two ear clips, and they need to be connected to Cyton and Daisy boards bias and reference pins, I made an Y cable for splitting the signals into two attached boards. Figure 5.2 is the ear clip after splitting the wires. we can see that it has two connectors to pins.

After preparing for the electrodes, I glued the skeleton of OpenBCI headset together. Because the headset was originally produced by 3D printing, I used sandpaper to scratch the attaching places, to make the places smooth to attach to each other. Each node was settled according to the 10-20 system, provided in Figure 1.4, and each of the electrodes were glued onto the skeleton so that it will not easily fall during experiments.

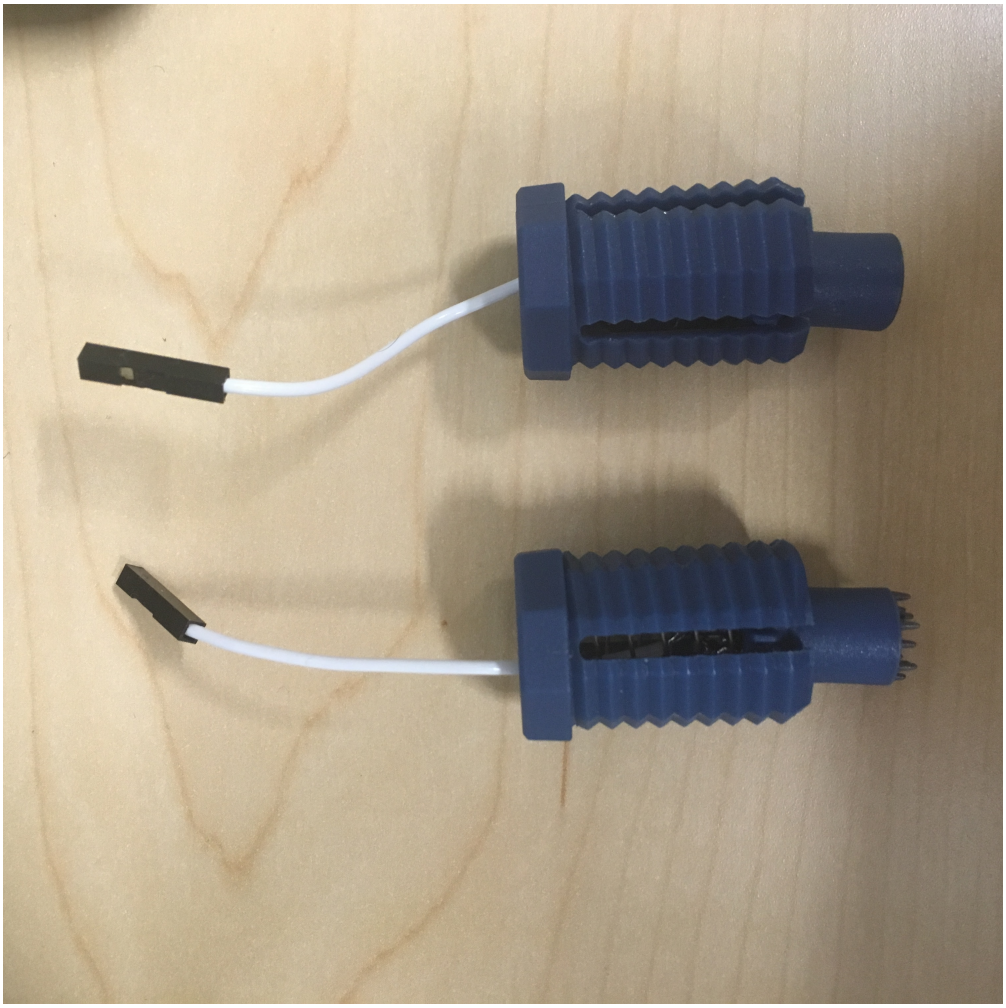

Figure 5.1: Electrodes for detecting brainwave. Upper is a flat unit for placements on front head, another one is a normal unit for places with hair

The chips are also plugged with a specific order. Daisy board is designed to mount on Cyton board as shown in Figure 5.3. The Daisy board has similar functions with Cyton board, both of which have an ADC to convert the analog signal input by the electrodes and reference ear clips to digital values of voltages. Then the Daisy board will send the signal to Cyton board, combine the signal together and finally sends the signals together to the RF

Figure 5.2: Ear clip with Y cable

module, then transmit to the receiver, which is the USB dongle. The transmission protocol is Bluetooth.

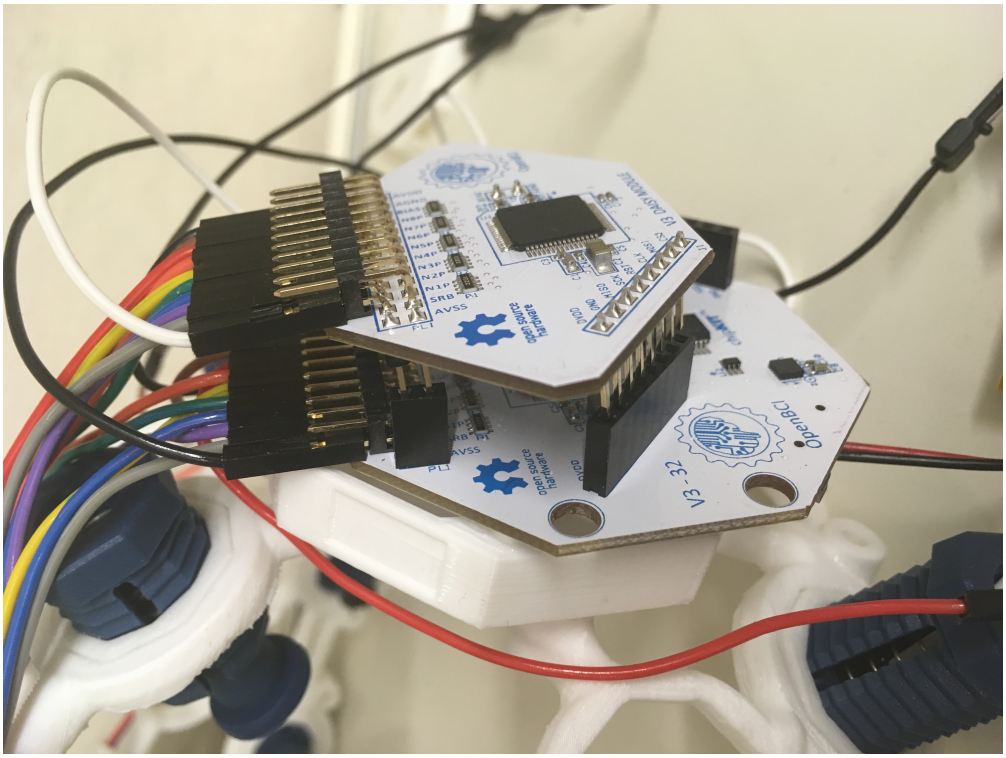

Figure 5.3: Daisy board mounted on Cyton board

5.2 Deployment

Deployment is important for the success of experiment, and I have some design for the specific low-cost EEG device to make the data more accurate, and the labels are labeled as precise as possible.

The experiment subjects are required to sit in front of a screen which will be used for sending the real-time instructions. The instructions are sent by updating the figures in MATLAB plotting window in real time, and the time interval is calculated precisely, and the computation time was considered into the program by timing the computation time. Shown in Figure 5.4 (a) and Figure 5.4 (b) are the sample of the instructions that the subject

receive. Figure 5.4 (a) is the instruction of moving right arms and hands, and Figure 5.4 (b) is stopping and take a rest.

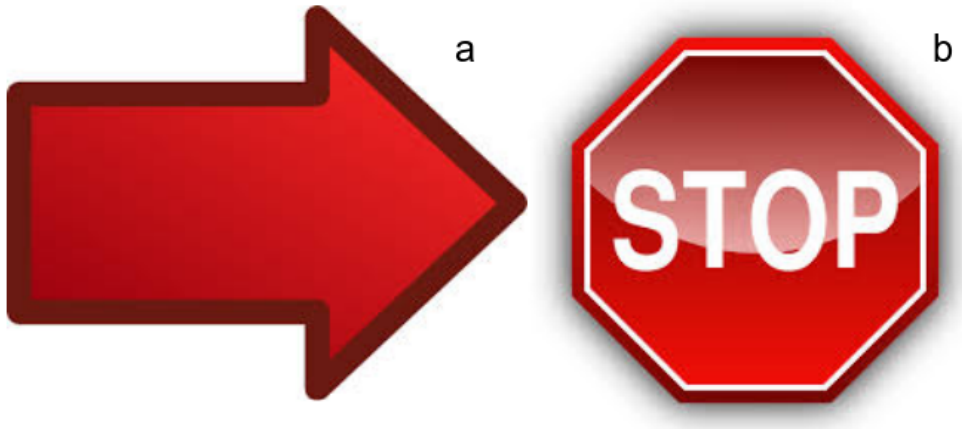

Figure 5.4: (a) The instruction for moving right hands, left hands is on the contrary. (b) The instruction of stopping current move and take a rest.

Although the experiment subjects are asked to take a rest, it is hard to keep in rest status and think about nothing. The experiment subject will be in a status of trying to think of nothing and stare at the screen to keep the data pure until asked to make a movement.

5.3 Experiment design

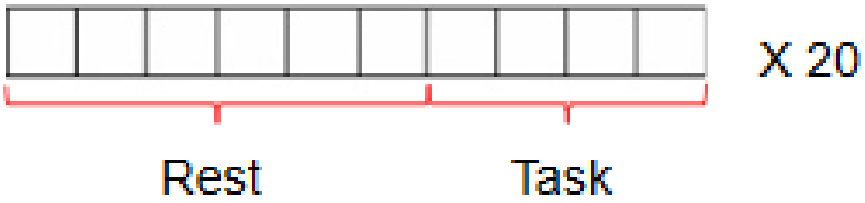

Figure 5.5: Experiment design

Shown in Figure 5.5 is the experiment design. Each small cube is corresponding to 1 second, Every 10 seconds which I will refer as a trial, the subject will first see the instruction of stop shown in Figure 5.4 (b), then with a probability of 50%, the subject will see either Figure 5.4 (a) or the inverse picture of it, and as the subject sees the instruction, they will immediately move the left arm and hand or right arm and hand according to the direction of the arrow.

The plan of experiment is that for each trial, the subject will follow the instructions which is in accordance with Figure 5.5, and they will be asked to do up to 20 trails in an experiment.

To make the time label in accordance with the data collection, a bash program is written to make sure that the start of instruction and data collection is at the same time.

5.4 Training Data Trimming

We can notice that when OpenBCI GUI starts, the data it collects will be mostly noise. I set 10 seconds as a buffer to make sure that the noise data does not affect the experiments. The instructions will start after the 10 seconds in each experiment. The comparison of Figure 5.6 and Figure 5.7 shows that the start of the data and in most channels the data is loud and noisy, and the stable data is different from it. After several seconds, the data becomes stable and available for the experiment.

Another point we need to pay attention to is that in the real training process, I trimmed the first seconds of both rest status and task status. The reason of doing so is that in the actual experiment, it is hard for the experiment subject to react to the instructions at the time they notice it. For channel P3, since the skeleton is too far from the subjects scalp, no signal is taken from the channel.

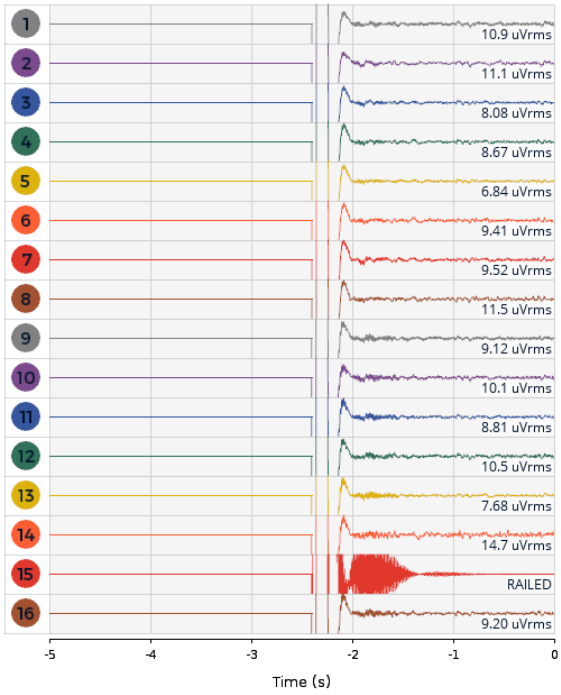

Figure 5.6: Data at the start of GUI for different channels

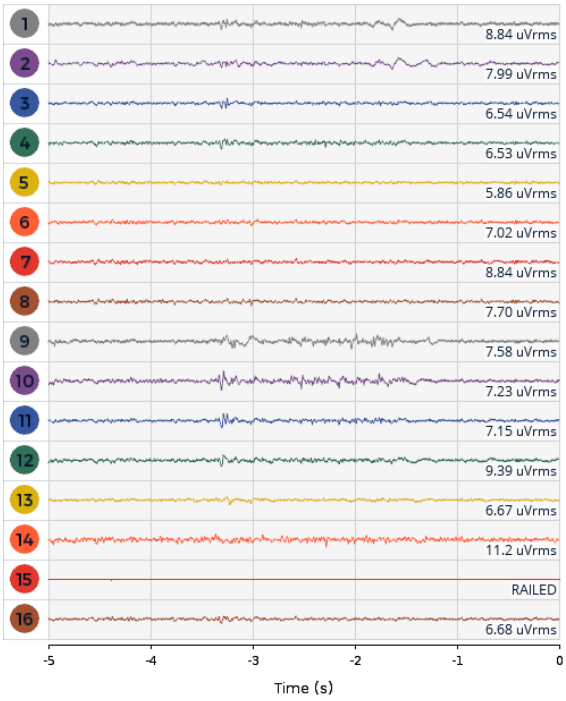

Figure 5.7: Data when the GUI is stable and collecting data correctly

In Chapter 6 I will specify about the result of the experiments, mainly including four parts, self cross validation, testing with another dataset, online dataset testing and real-time robot control demonstration.

Chapter 6

Experiment Result and Discussion

6.1 Result with 5-fold Cross Validation

For 5-fold cross validation, I trained data based on three datasets, and Table 6.1, Table 6.2, Table 6.3 are the accuracy and classification confusion matrices of the three datasets. In all trainings, 99.9% of the information are held through PCA process. Figures in parentheses are the number of the samples. Dataset 1 is with 10 trials, Dataset 2 is with 15 trials, and Dataset 3 is with 20 trials.

Table 6.1: Validation result from Dataset 1

Accuracy = 88%	**Rest**	**Left**	**Right**
Rest	90% (18830)	6% (1216)	4% (896)
Left	17% (1001)	80% (4636)	3% (149)
Right	10% (643)	7% (439)	84% (5633)

Table 6.2: Validation result from Dataset 2

Accuracy = 88%	**Rest**	**Left**	**Right**
Rest	90% (23398)	3% (747)	7% (1936)
Left	18% (1055)	80% (4614)	2% (113)
Right	11% (1104)	3% (269)	86% (8627)

Table 6.3: Validation result from Dataset 3

Accuracy = 91%	**Rest**	**Left**	**Right**
Rest	94% (38805)	4% (1470)	3% (1196)
Left	17% (1001)	80% (4636)	3% (149)
Right	10% (643)	7% (439)	84% (5633)

From this result we can see that generally this model runs well for the datasets, all the datasets have shown good result for the three label classifications, some of the misclassifications might be related with the labeling and other attention issues, thus this result is good for self-validations with some bearable misclassification rate.

But this also poses a question that what will happen if I use the model to classify a set of data that comes from another experiment. There will be in total three situations considered in this testing with different dataset problem.

6.2 Results with Testing with Different Datasets

Mainly there will be three different situations in this testing with different dataset problems, listed here:

1. The headset is moved between the two datasets, that is, the headset has been moved, or removed from head, then put on, so that the electrode location will be different

between the two datasets, although the location is similar to each other. The dataset is clean.

2. The headset is not moved between the two datasets, but the training or testing dataset is contaminated. This is because that the OpenBCI headset is also sensitive to environmental noises. Even I have passed the training and testing data for band pass filters, there may exist noise lies in the range that I did not filter out.

3. The headset is not moved between the two datasets, and both datasets are clean.

Suppose the dataset 4 and dataset 5 are taken in a clean way, and dataset 6 is taken when some other people are talking and walking around the subject while the experiment is on. I first record dataset 4 and dataset 5 without moving the headset, then record dataset 6, but use data contaminations as described as situation 2 above. Then record dataset 7 with no data contaminations but movement of headset. To accelerate the experiment, I used 15 iterations rather than 20 iterations as before.

Table 6.4 is the testing result of the situation 3, which trains with dataset 4 and test with dataset 5. Table 6.5 is the result of situation 2, where the testing data is contaminated. Table 6.6 is the result of situation 1, where the test data and training data are not taken in the same headset location.

Table 6.4: Result of training with Dataset 4 and testing with Dataset 5

Accuracy = 83%	**Rest**	**Left**	**Right**
Rest	91% (19030)	5% (1073)	4% (899)
Left	22% (1421)	63% (4079)	17% (1018)
Right	15% (897)	10% (566)	75% (4414)

Table 6.5: Result of training with Dataset 4 and testing with Dataset 6

Accuracy = 77%	**Rest**	**Left**	**Right**
Rest	80% (33293)	8% (3522)	11% (4656)
Left	12% (675)	40% (2335)	48% (2766)
Right	5% (324)	3% (206)	92% (6185)

Table 6.6: Result of training with dataset 4 and testing with dataset 7

Accuracy = 68%	**Rest**	**Left**	**Right**
Rest	73% (19087)	8% (2150)	19% (4844)
Left	16% (915)	15% (889)	69% (3978)
Right	7% (760)	6% (611)	86% (8629)

We can see that the first one of the test shows good or acceptable result, while other two with location changes and contaminations are bad. It can lead to mainly three conclusions:

1. From Table 6.4 we can see that even the datasets are different, if the electrode locations dont change, and the data is not contaminated, the accuracy is high, but we can see some accuracy drop in the classification of left and right labels. For the drop in classification of Left, the drop is obvious, but since more classifications are classified as rest, the controlling task will not be biased. There is also no obvious bias towards specific labels for this testing set.

2. From Table 6.5 we can see that if the data is contaminated, the classification will be biased to some other labels, and the result will not be acceptable. Also for Table 6.6, we can get the conclusion that the data is sensitive to small changes in electrode location too. Each time we need to deploy the applications, we need to do online calibration first, then the application can be accurate

3. Table 6.5 and Table 6.6 in another prospective, showed that our experiments result is legal. If the classification of labels is mainly from noises like waving hands, then both result of Table 6.5 and Table 6.6 will be more accurate since that is not changed in the experiments.

6.3 Online Testing

I also tested with online system. For the building up the online testing platform, I mainly used LSL as the data transmission protocol to send the data by chunk from OpenBCI GUI to MATLAB, and used the same processing techniques for the new data, then projected to the eigenspace for training the model. For the online testing, I also used the same plan of the experiment, and directly plotted the actual label and the real-time classification labels. Figure 6.1 and Figure 6.2 are the comparisons of the classification result and the original label from the dataset.

From the comparison, we can mainly find two conclusions:

1. The real-time system has some delay, it started early but got some delays for starting the first task which is acceptable, it is mainly because of the data transmission, and processing time. Waiting for new chunk of data also takes time. For the first 10 seconds of the starting time, the system still worked well for classification.

2. The general accuracy is acceptable, the tendency of the classification fits the original label. There are some spikes mainly in the sessions of tasks, but the spikes are acceptable. During the whole session, only around 6 seconds are wrongly classified in the whole time span of 160 seconds, we can get the conclusion of the accuracy is around 96%, not considering the system delay and mis-classifications on status change.

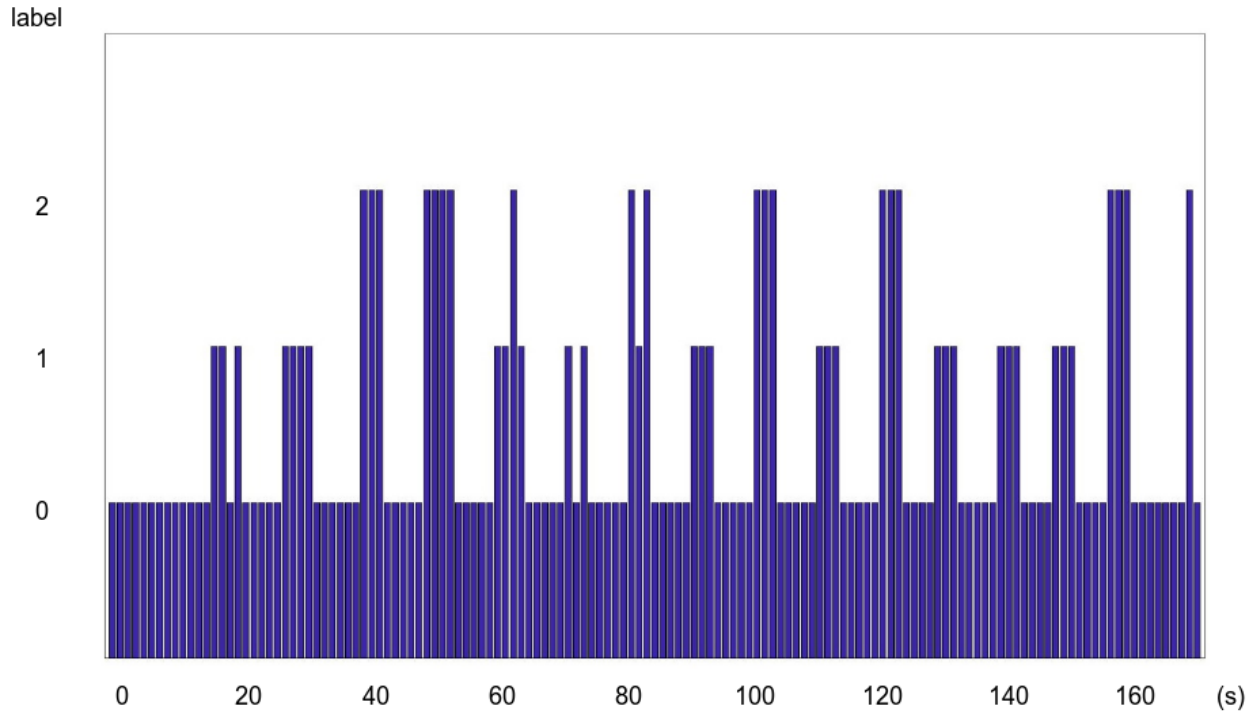

Figure 6.1: Realtime testing classification result

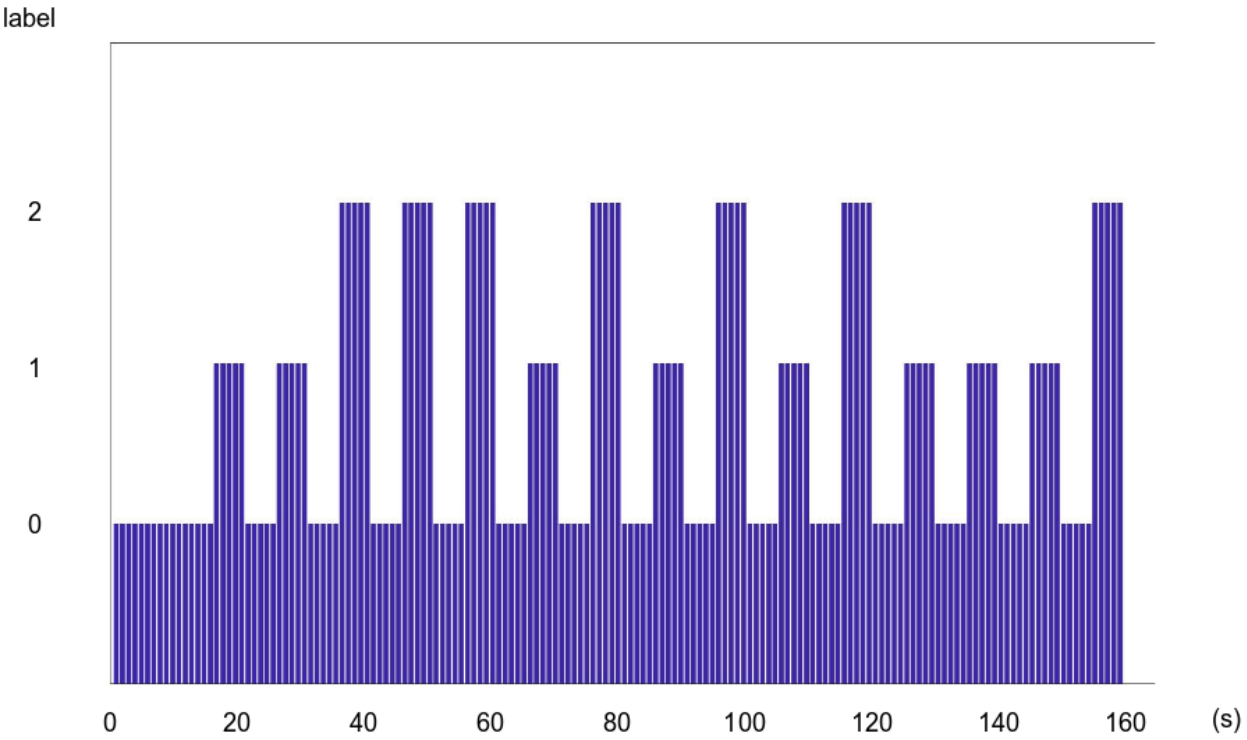

Figure 6.2: Realtime testing original label

6.4 Testing with Robot and Oral Instructions

The real-time application will be the controlling of a robot to turn left or right. I set up a router to link the workstation that process the data and do the classification to the robot that performs the actions. Once a classification is done, a control signal will be sent to the robot by TCP connection, and thus the robot will perform accordingly. I programmed the robot to listen to the signal all the time, and one a signal is available, it will capture the signal and perform the action of the instruction.

The demonstration video of this experiment is available here: Left movement demonstration and Right movement demonstration. At the start of the videos, the instructor will instruct the subject of moving left or right hands, and the subject will move accordingly, the headset will capture the brainwave in real-time, and sent back to the work station to do the classification, and the work station will send the control signal to the robot. Figure 6.3 is the LEGO EV3 robot that I used for the final experiment and demonstration.

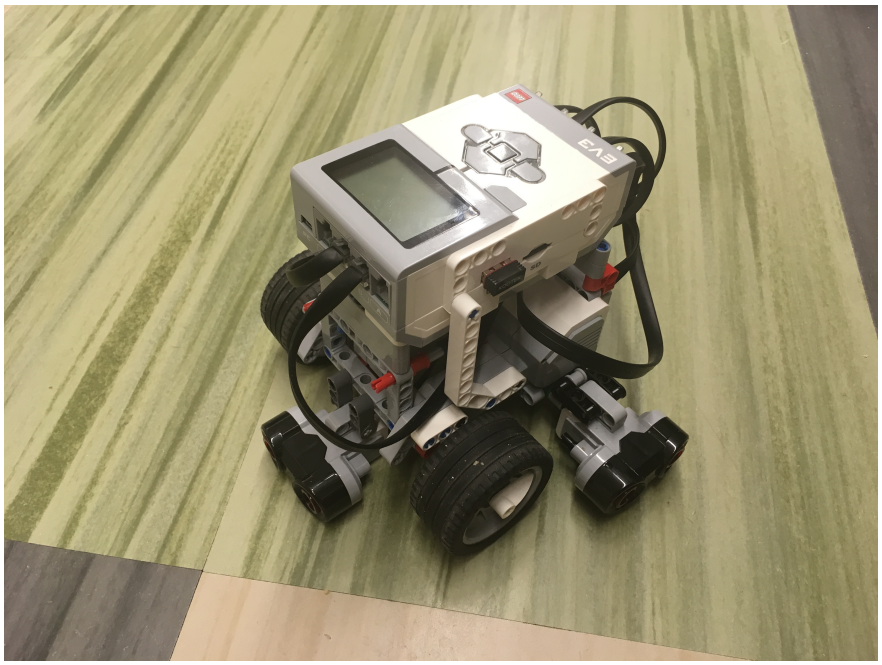

Figure 6.3: LEGO EV3 robot for experiment

Chapter 7

Conclusion and Future Work

The main contributions of this work are as follows:

1. A system that works with real-time application for low cost EEG devices is provided, which in our project, is OpenBCI. Both hardware and software implementation has been finished and available. The whole system is explained in detail.

2. A data driven pipeline has been proposed for the process of low cost EEG signals, which is different from a normal EEG device. Low cost EEG devices is good in terms of convenience, availability, and portability, but falls short in the information quality and amount. Some data processing techniques are applied in this project to get as much useful information as possible, and the final classification result is acceptable.

3. The system Achieved a satisfiable classification accuracy of 88% to 92% for the classifications, even for real-time classification tasks, the accuracy keeps high and this shows a high availability of the use of the system.

we should also notice that, the current work has several limits. The experiments are done mainly by a single experiment subjects, so in the future, it will be interesting to see more subjects validating this system. Another major limitation is that the system for this step

requires a long time for calibration, which is highly sensitive to environmental noises. More future work can be focused on diminishing the effect of noises, and the generalization of models with more ability for classifying without the effect of electrode locations. In order to do so, a lot more experiments need to be finished and more experiment subjects are needed. It will also be a bright future for combining EEG signals with other data such as visual data, and other sensing data to have more level of information, to better and more accurately monitoring the movement, or thought of human beings. BCI is a promising field that can lead to more applications that can have impacts on this whole world and the future of human beings.

Bibliography

[1] R. W. Homan, J. Herman, and P. Purdy, "Cerebral location of international 1020 system electrode placement," *Electroencephalography and Clinical Neurophysiology*, vol. 66, pp. 376–382, Apr. 1987.

[2] M. Salinsky, R. Kanter, and R. M. Dasheiff, "Effectiveness of multiple EEGs in supporting the diagnosis of epilepsy: An operational curve," *Epilepsia*, vol. 28, pp. 331–334, Aug. 1987.

[3] "The application of EEG sleep for the differential diagnosis of affective disorders," *American Journal of Psychiatry*, vol. 135, pp. 69–74, Jan. 1978.

[4] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," *Journal of Neuroscience Methods*, vol. 134, pp. 9–21, Mar. 2004.

[5] C. A. Kothe and S. Makeig, "BCILAB: a platform for braincomputer interface development," *Journal of Neural Engineering*, vol. 10, pp. 056014+, Oct. 2013.

[6] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "BCI2000: A General-Purpose Brain-Computer interface (BCI) system," *IEEE Transactions on Biomedical Engineering*, vol. 51, pp. 1034–1043, June 2004.

[7] P. J. Durka, R. Kuś, J. Żygierewicz, M. Michalska, P. Milanowski, M. Łabcki, T. Spustek, D. Laszuk, A. Duszyk, and M. Kruszyński, "User-centered design of brain-computer interfaces: OpenBCI.pl and BCI appliance," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 60, Jan. 2012.

[8] M. Wright, "Open sound control: an enabling technology for musical networking," *Organised Sound*, vol. 10, pp. 193–200, Nov. 2005.

[9] C. Kothe, *Lab streaming layer (lsl)*.

[10] K. LaFleur, K. Cassady, A. Doud, K. Shades, E. Rogin, and B. He, "Quadcopter control in three-dimensional space using a noninvasive motor imagery-based braincomputer interface," *Journal of Neural Engineering*, vol. 10, pp. 046003+, Aug. 2013.

[11] S. Liu, J. Lv, Y. Hou, T. Shoemaker, Q. Dong, K. Li, and T. Liu, "What makes a good movie trailer?," in *Proceedings of the 2016 ACM on Multimedia Conference - MM '16*, pp. 82–86, ACM Press, 2016.

[12] K. K. Ang and C. Guan, "EEG-based strategies to detect motor imagery for control and rehabilitation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, pp. 392–401, Apr. 2017.

[13] I. Sakamaki, C. E. P. del Campo, S. A. Wiebe, M. Tavakoli, and K. Adams, "Assistive technology design and preliminary testing of a robot platform based on movement intention using Low-Cost brain computer interface," Oct. 2017.

[14] I. T. Jolliffe, "Principal component analysis and factor analysis," in *Principal Component Analysis*, Springer Series in Statistics, pp. 115–128, Springer New York, 1986.

[15] R. Srinivasan, D. Tucker, and M. Murias, "Estimating the spatial nyquist of the human EEG," vol. 30, no. 1, pp. 8–19, 1998.

[16] M. S. Chavan, R. A. Agarwala, and M. D. Uplane, "Design and implementation of digital FIR equiripple notch filter on ECG signal for removal of power line interference," *WSEAS Trans. Sig. Proc.*, 2008.

[17] L. E. Scriven and C. V. Sternling, "The marangoni effects," *Nature*, vol. 187, pp. 186–188, July 1960.

[18] E. Başar, C. Başar-Eroglu, S. Karakaş, and M. Schürmann, "Gamma, alpha, delta, and theta oscillations govern cognitive processes," *International Journal of Psychophysiology*, vol. 39, pp. 241–248, Jan. 2001.

[19] Y. Yasui, "A brainwave signal measurement and data processing technique for daily life applications," *Journal of PHYSIOLOGICAL ANTHROPOLOGY*, vol. 28, no. 3, pp. 145–150, 2009.

[20] W. Freeman and J. Zhai, "Simulated power spectral density (PSD) of background electrocorticogram (ECoG)," *Cognitive Neurodynamics*, vol. 3, no. 1, pp. 97–103, 2009.

[21] J. Ye, R. Janardan, and Q. Li, "Two-dimensional linear discriminant analysis," in *Proceedings of the 17th International Conference on Neural Information Processing Systems*, 2004.

[22] S. Bhattacharyya, A. Khasnobish, S. Chatterjee, A. Konar, and D. N. Tibarewala, "Performance analysis of LDA, QDA and KNN algorithms in left-right limb movement classification from EEG data," in *2010 International Conference on Systems in Medicine and Biology*, pp. 126–131, IEEE, Dec. 2010.

[23] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Regularized discriminant analysis for the small sample size problem in face recognition," *Pattern Recognition Letters*, vol. 24, pp. 3079–3087, Dec. 2003.