HØGSKOLEN I ÅLESUND

Aalesund University College

Man/Machine interaction through EEG

Fredrik Hoel Helgesen

Rolf-Magnus Hjørungdal

Daniel Nedregård

Submission date: Mai 2015

Supervisor: Associate professor Robin T. Bye

Co-supervisor: M.Eng Anders Sætersmoen

Department of Automation Engineering
Aalesund University College

Preface

This bachelor thesis marks the completion of the Automation Engineering program at Aalesund University College which is comprised of 20 credits and is completed in the spring 2015.

Working on this thesis has been challenging, and thus it has led to a greater understanding in the use of 3D visualization, simulation, signal processing and how to work as a team. It wraps up many essential concepts and key elements in engineering, and thus it has been a valuable experience for the group.

We would like to give our thanks to our supervisors, associate professor Robin T.Bye and M.eng Anders Sætersmoen for contributing valuable advice and, feedback during the project period. In addition to that we would like to give our thanks to the Unity- and Emotiv communities for guidance and sharing of good information in their forums.

Additionally we will like to give our thanks to professor Hans Georg Schaathun for good guidance and advice for solving the neural network.

Fredrik Hoel Helgesen

Rolf-Magnus Hjørungdal

Daniel Nedregård

Abstract

This report has the objective of describing the process of creating a virtual reality game environment that allows the user to practice brain computer interface (BCI) through controlling a BCI enabled wheelchair. It serves as a training platform for those who could possibly need a brain controlled wheelchair. The report also contains technical information on how various technologies such as Emotiv Epoc, A*, EEG (electroencephalography) and Unity 3D work. The work is inspired by a condition known as ALS or Lou Gehrig's disease which gradually leaves the patient in a state of complete paralysis, this means conventional electronic wheelchairs won't be sufficient. With this in mind and the fact that BCI has emerged in recent years, the group is able to investigate future possibilities. To conclude the work the group investigated the process Emotiv would use to distinguish independent cognitive states.

The result is an open world game environment which allows subjects to train and, become familiar with the equipment and the technology. In addition, during the BCI investigation the group discovered that it was possible to discern and classify different cognitive states via EEG using a neural network.

Concepts

Electroencephalography
Electroencephalography is the recording of electrical activity along the scalp. EEG measures voltage fluctuations resulting from ionic current flows within the neurons of the brain.

Emotiv Epoc
The headset used for EEG, where Emotiv is the manufacturer and the model is named Epoc.

Client-server model
Client-server model is a program relationship in which the client sends requests and the server listens to those requests and responds the required task. [F.Kurose and W.Ross, 2013].

Dynamic Link Library (DLL)
A file containing a software library specific to Microsoft Windows. The file contains functionality which can be called upon programmatically by external sources.

Plugin
In this context a plugin is used to include code/programs outside Unity 3D by creating code and scripts as a plugin.

Software Library
A collection of code that is callable by the programmers code which simplifies the programming process by allowing the programmer to use relevant and useful code regardless of context.

API
The interface for the library, meaning the specific functions and methods that are present within it.

SDK
Software Development Kit - A software library with useful tools which help you develop software for the given library.

Common Language Runtime
The virtual machine component of Microsoft's .NET framework.

.NET-framework
Microsoft Windows' large collection of software libraries along with common language runtime which ensures uniform behaviour between programming languages.

MATLAB-Script
A collection of MATLAB commands stored in plain text files. Often referred to as m-files.

Scatterplot
A scatter plot is a type of mathematical diagram using coordinates that specifies each point uniquely in a plane by a pair of numerical coordinates, this is to display values for two variables for a set of data.

Neural Network
A machine learning technique which consists of a network of artificial neurons. The network can be trained to produce output based on the inputs given to the network. In this assignment we will use a neural network to solve a classification problem.

C sharp
A programming language encompassing functional, generic, object-oriented (class-based) and component-oriented programming disciplines.

UDP
UDP (User Datagram Protocol) is used to communicate between two or more processes. UDP is said to be "connectionless" since it is not carrying out any handshake. UDP is sending bytes directly with an address connected to the broadcast
[F.Kurose and W.Ross, 2013].

Oculus Rift
A virtual reality head-mounted display developed by Oculus VR. The Oculus Rift enables the user to experience realistic environments due to a higher level of immersion.

Abbreviations

EEG	Electroencephalography
EMG	Electromyography
ALS	Amyotrophic Lateral Sclerosis
BCI	Brain–Computer Interface
AI	Artificial Intelligence
FFT	Fast Fourier Transform
LSB	Least Significant Bit
DLL	Dynamic Link Library
API	Application Programming Interface
SDK	Software Development Kit
C#	C Sharp
VR	Virtual Reality
A*	A star (type of algorithm)
UDP	User Datagram Protocol
NN	Neural Network

Table of contents

List of Figures

1 Introduction

1.1 Background

In recent years the electroencephalogram (EEG) technology has proven itself to have a more widespread use than previously imagined. With the Emotiv Epoc headset, which is a commercial EEG device aimed at the general public, EEG readings at home are made possible at a relatively cheap cost. It comes with software that enables the user to look at their EEG readings in real-time with functionality which separates the signal into different frequency bands (i.e., delta, theta, alpha, beta). The most important feature in the context of this report is the fact that the software allows the user to store EEG signals as commands in an easy to use interface which essentially means that a machine can be controlled through EEG, achieving what is known as brain computer interface (BCI). Having sophisticated EEG technology at this price is relatively new considering EEG was generally used in hospitals or in psychiatric institutions. It was used as a tool to diagnose by looking at anomalies in the signals that could indicate certain illnesses such as epilepsy or brain tumors. [Vaque, 1999].

With the development of this type of technology the group decided to create a brain controlled wheelchair within the game engine Unity 3D, which could have a positive social impact. ALS is a severe medical condition that completely paralyses the patient which means they are completely unable to use conventional electric wheelchairs such as those controlled by a joystick. If those afflicted with ALS had brain controlled wheelchairs it would significantly improve their quality of life and sense of freedom. In this project the focus is primarily on creating a training platform in virtual reality by using Unity 3D and the Oculus Rift VR headset. As a first time user of this technology the most challenging aspect is being able to focus on one single thought consistently in order to initiate a command. To achieve this goal a training platform would be very helpful. With the experience provided by VR technology the group wanted to explore the possibility of improving focus and learning. Some previous reports on the subject suggest that VR can make focusing and control through thoughts easier by making the experience seem more real. [Lécuyer et al., 2005].

1.2 Scope

The scope of this report is limited to the technology provided which is the Emotiv Epoc EEG headset, Unity 3D game engine and the Oculus Rift headset. It is also limited by the issue which the group defined as creating a training platform in VR for patients that want to use brain controlled wheelchairs. Literature study on EEG and covering its history and use will also be within the scope, this will be covered throughout the report giving a fundamental understanding on how the technology works. The safety concerns and, whether or not this technology can be used, is also visited as part of the discussion. The scope also includes a minor investigation on how pattern recognition and neural networks could realize the technology that Emotiv provides. The company is secretive when it comes to the classification of cognitive states, which in turn peaked the interest of the group, prompting the group to investigate the subject.

1.3 Aim

The issue for this bachelor thesis was given by Aalesund University College. The main goal is to create an experimental platform for real-time communication and control of Unity virtual environments using the Emotiv EEG headset. The different challenges the group will discover through the project are listed below:

1. Establish communication and control between the EEG headset and Unity. The group will test out different methods for communication between Unity and the EEG headset.

2. Study literature on EEG experiments involving 1D, 2D, or 3D tasks on screen. The group will do a research on earlier experiments that have been performed using EEG for control.

3. Create one or several test environments ("games") in Unity for performing experiments, The group will choose a main topic to explore further by various experiments in an environment made in Unity 3D. Here the group will either extend the work of someone else or even better, try to experiment with something that has never been done before.

4. Provide documentation and modularity of software for future work. The code must be clear, easy to read, and with complete documentation. The report needs to be at such standards as to make it easy for others to continue this work.

5. Investigate raw EEG signals through processing and analyzing data. The group will try to explore EEG signals that can be used to recognize patterns, hopefully in such a way that they are usable to achieve BCI. In other words, the group wants to tap into Emotiv's secrets.

This report will describe the process of creating a VR environment in Unity intended for use with the Oculus Rift VR headset. The environment will be a training platform for EEG controlled wheelchairs using the Emotiv EEG headset. Literature study on the history of EEG and how EEG is used today as a thought recognition technology is also an important part of this research. Lastly the report will review how EEG can be used to recognize signals as commands used to achieve BCI the same way Emotiv does by trying to achieve some sort of pattern recognition.

2 Theory

2.1 The History of EEG

Only in recent years has EEG been used to control machines, before that it was mainly a tool for diagnostics by providing insight in brainwave activity during ailments such as epilepsy attacks. Other conditions which can be diagnosed include sleep disorders, behaviour change and head injuries. EEG was also used to diagnose tumors, strokes and other focal brain disorders, but MRI and CT scanning has replaced EEG in these areas.

It all started in 1929 when the EEG machine was introduced to the world by Hans Berger, a psychiatrist who suggested that brain currents changed based on the functional status of the brain. EEG readings can be affected during states such as sleep, epilepsy attacks and anesthesia. Since the scientific community did not believe in his conclusions it took five years before the british scientists Edga Douglas Adrian and B. C. H. Matthews in 1934 verified Hans Berger's conclusions through experimentation. In 1936 a respected neurophysiologist by the name of William Grey Walter proved that the EEG technology could be used to pinpoint a brain tumor by using a large number of small metal electrodes that he pasted to the scalp [Vaque, 1999].

Over the years technologies such as amplifiers, EEG electrodes, and output devices have improved. This resulted in a plethora of new knowledge and scientists that discovered how to create electrical maps of the brain. In 1957 a new device called the toposcope was developed by Walter, a machine that used the EEG activity to produce a map of the brain's surface. The map demonstrated and proved that brain waves in a resting state were different than other brain waves that were generated during different states of mind. Today, EEG machines have computer storage memories, multiple channels and intelligent software that offers a wide range of features. [Biomedresearches, 2006].

2.2 Fundamentals of EEG

The brain is composed of neurons which communicate with each other through electrical impulses, this creates fluctuations in voltage at different frequencies across and through the brain flowing all the way up to the scalp where they can be measured using electrodes. EEG mostly detects the currents that flow from synaptic excitations of the dendrites (branched projections of a neuron) of pyramidal neurons (excitation units) in the cerebral cortex (the outer layer of the brain). The electrical current in the brain mostly consists of Na+, K+, Ca+ and Cl- ions that move through neuron membrane channels where the direction is governed by the electric potential of the membrane [Teplan, 2002]. It should be noted that only large collections of neurons are capable of generating signals that can be detected on the scalp. With the headset in this project, the signal is digitally amplified and processed.

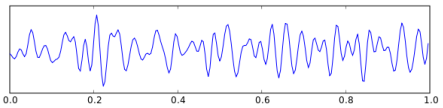
2.2.1 The Brain

The number of neural cells are already developed at birth, the neurons are connected through neural nets called synapses. Adults have about 500 trillion of these, the number of synapses per neuron increases with age and at the same time the number of neurons decrease at such a rate the the total number of synapses decrease too. The brain is dividable into three sections, cerebrum, cerebellum and brain stem. The cerebrum is made up of the left and right hemisphere with the wrinkly cerebral cortex acting as the surface layer. The cortex is a vital part of the nervous system. The cerebrum is responsible for initiating movement, conscious awareness of sensation, complex analysis and emotional and behavioural expression. This partly explains how EEG has such a versatile usage. Lastly the brain stem controls more basic functions as respiration, heart regulation, biorhythms, neurohormone and hormone secretion and so on. [Teplan, 2002]

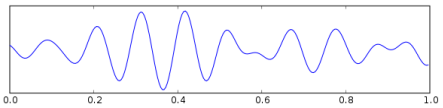
2.2.2 The Different Types of Brainwaves

The brain emits an array of frequencies simultaneously moving from low frequency delta waves to high frequency gamma waves. The frequencies detected by EEG are beta (>13 Hz), alpha (8-13 Hz), theta (4-8 Hz) and delta (0.5-4) Hz [Teplan, 2002]. At any given time depending on the state of mind different frequencies dominate the signal coming through the headset. At a relaxed state of mind the brain operates at lower frequencies than if stressed or exited, the signal is at its slowest when in deep sleep. So the physical technology behind EEG is fairly simple, the challenge and difficulties is found in the question of how the signals should be interpreted. According to different research papers such as Fundamentals of EEG measurement and [1] it is the beta wave that is found most dominant during conditions of normal open-eye wakefulness [Teplan, 2002].

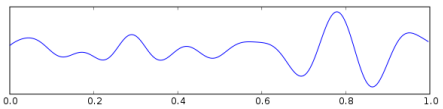
[1] http://www.brainworksneurotherapy.com/what-are-brainwaves



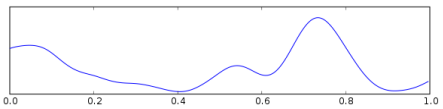
(a) Beta waves(>13 Hz)



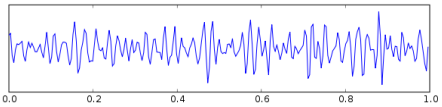
(b) Alpha waves(8-13 Hz)



(c) theta waves(4-8 Hz)



(d) Delta waves(0.5-4 Hz)



(e) Gamma waves(40-100 Hz)

Figure 2.1: The different brain waves represented in time domain [Wikipedia, 2006]

Figure 2.1 found above, represents the different brainwaves in the time domain. It also shows the gamma wave, most commonly known as "the genius brain wave", it tends to occur more often in very intelligent people or someone doing something they enjoy very much, also known as being in a flow state [Slaven Cvijetic, 2013]. The gamma brainwaves are the highest frequency brain waves with a range from 40 Hz to 100 Hz, they can not be measured by EEG, this is because the skull filters out high-frequency signals. [Games, 2006].

2.3 Signal Analysis

To achieve what we call BCI, the technology needs to detect a calibrated brain wave pattern generated by thought. Through EEG readings and with the wonders of signal processing many things are possible. You could say something about where different activities are present through the standardization of sensor placement, also it is possible to say something about which frequencies are the most dominant at any given time through FFT. These are all general and simple concepts, but what makes BCI possible is highly advanced and complicated algorithms that are able to filter out noise and detect patterns in the signal, this pattern recognition gives away whether or not the user is in fact initiating a pre-calibrated command. To calibrate a command, the user would have

to open up Emotivs calibration software. It works by having a list of commands ready for calibration, but first you would have to allow the software to store the patterns generated in your neutral state. When that is done, calibration works by selecting the command to calibrate, followed by focusing on whatever thought you want to use to initiate that command, you would have to maintain focus for 8 seconds so the software can store you patterns properly. When this is done, thinking about the same thing you did during the calibration should initiate the command. This requires some heavy pattern recognition techniques that utilizes advanced algorithms, as described in this chapter.

2.3.1 EEG Measurements

The required components for acquiring EEG signals can be listed in 4 items.

- Electrodes
- Amplifier
- A/D Converter
- Data Recording

This type of EEG technology works by placing electrodes on the scalp in order to detect fluctuations in voltage. These fluctuations in voltage are created by the sum of electric potential of the excitation units between the body of the neuron (soma) and the branch (dendrites). The Emotiv Epoc headset consist of 14 electrodes (7 pairs), plus two reference points functioning as a common ground for the electrodes. The reference electrodes are placed behind the ear at the mastoid. The signal from the electrodes are then fed into a differential amplifier which amplifies the difference between an electrode and the reference node (figure 2.2). Finally, the signal is quantified in an A/D-converter.

Due to low signal-to-noise ratio the differential amplifier must meet several requirements for it to be able to produce an acceptable output signal. First of all, in order to bring the voltage level up to a level where an A/D-converter can accurately convert the signals to digital values, the amplifier needs a high gain. This gain is often 100-100,000 due to input voltages being in the μV-range [Nagel, 1995]. Amplification gain must of course coincide with the A/D-converter being used. In addition, as these measurements are highly susceptible to electrical noise, the differential amplifier must have a high common-mode rejection ratio (CMRR) (¿100dB), as well as a high input impedance ($> 100M\Omega$) [Teplan, 2002]. The common mode rejection ratio is a differential amplifiers ability to suppress signals common to both inputs. As a result, having a high CMRR will greatly reduce the electrical noise. Finally, the A/D-converter samples the continuous analog signal, and converts it into binary values. The A/D-converters specifications should have >12Bit resolution, with the ability to resolve 0.5μV [Teplan, 2002]. Emotiv Epoc used in this project has a resolution of 14 Bits, ultimately making the least significant bit (LSB) resolution 0.51μV [Emotiv, 2014a].

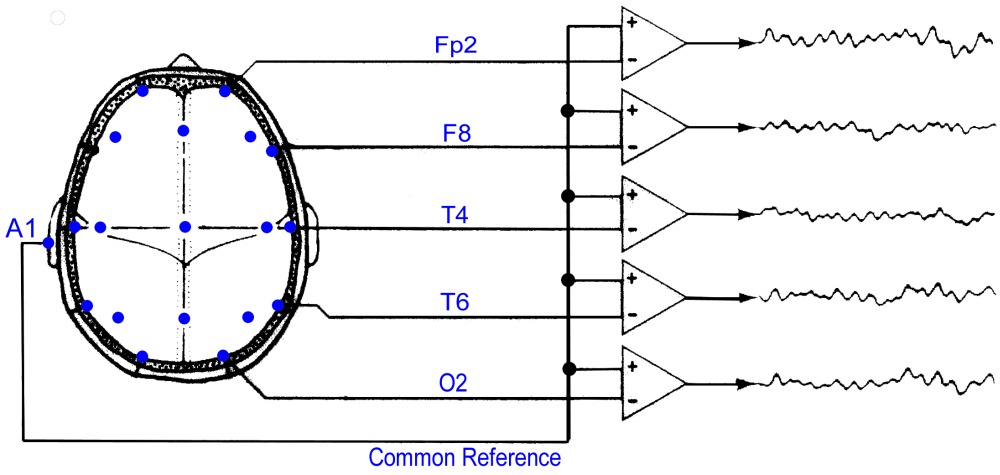


Figure 2.2: Simplified electrical circuit illustrating EEG measurement [Hu, 2013]

The placement of the different electrodes is not random, the 10/20 electrode placement system is an internationally recognized method used to describe the location of scalp electrodes in the context of an EEG test. The figure below illustrates the principle of how this system works. The actual distance between the adjacent electrodes are either 10% or 20% of the total front-back or right-left distance of the skull [Trans Cranial Technologies Ltd., 2012]. The various locations of the electrodes are based on defined areas: the nasion (on the top of the nose, at eye level), and the Inion (the bony lump at the base of the skull at the back of the head). All these points are shown in figure 2.3.

The letters in figure 2.3 each represent placements of electrodes on the scalp. These letters and the corresponding placements are presented in table 2.1.

Letter	Placement
A	Earlobe
C	Central
F	Frontal
Fp	Frontal polar
O	Occipital area
P	Parietal
Pg	Nasopharyngeal
T	Temporal

Table 2.1: The letters related to the different placement of electrodes

With the Emotiv headset the user can directly place the headset on the head, since this is designed and developed according to this standardized method for electrode placement.

Due to the low signal-to-noise ratio raw EEG signals require a great deal of processing to get meaningful data. First of all, the signal must first be filtered at both the high and low end. Applying a high-pass filter with a very low cutoff frequency will remove the DC offset of the signal, ideally without removing parts of the low-frequency delta waves. On the high end of the spectrum, the signal contains a significant amount of noise. The noise

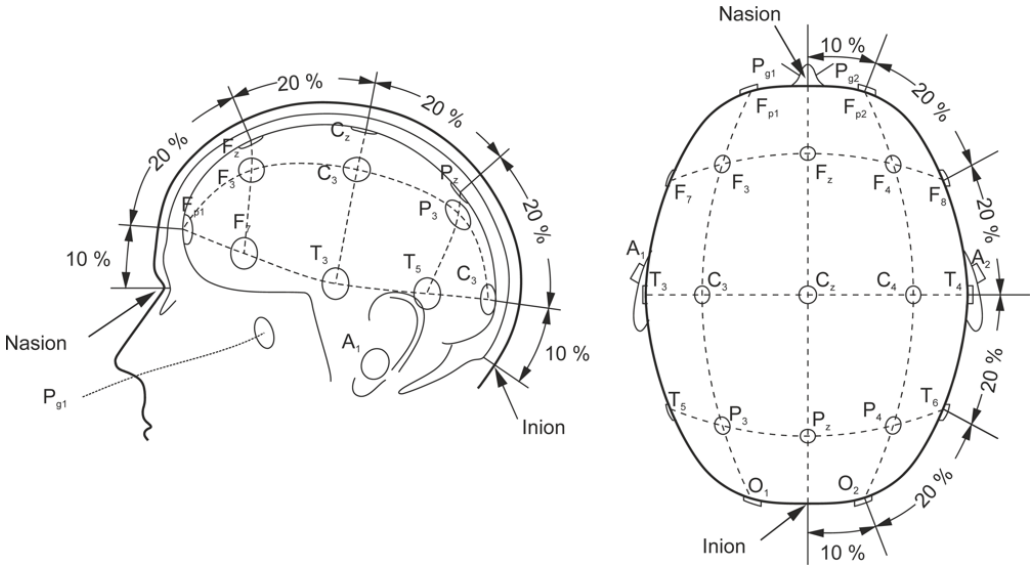


Figure 2.3: The placement of sensors attached to the scalp [Luis Fernando et al., 2012]

can be categorized into two separate categories: noise from external sources, and noise from the subject using the headset. Noise from external sources is mostly electromagnetic interference from electronic equipment like phones, monitors, wireless networks. [Repovš, 2010]. Many of these sources can easily be removed without too much effort User-induced noise is harder to remove, but can be reduced. Muscle activation like blinking and eye movement generates noise at 100 Hz and above, so filtering out frequencies above this would reduce artifacts. In addition to that the headset user should refrain from blinking and moving too much during the critical phases of the EEG measurement.

2.3.2 Pattern Recognition

There are two types of algorithms that can be used to help identify EEG signals, regression or classification, the latter is the type that is used the most. The algorithms aim to recognize patterns that exist within the signal, this subject is also related to any pattern recognition technology. How well the pattern recognition system work depend on the features and the classification algorithm, tried out features include amplitude values, band powers, power spectral density, auto regressive, adaptive auto regressive, time frequency, and inverse model based features. The features try to solve a number of problems related to EEG, one of which is noise, EEG signals have a very poor signal to noise ratio, so extracting the signal from the noise is highly important. Also, the signal is multi-dimensional, and a single feature vector is needed. Information about time is also important since EEG signals are time varying and the signals vary rapidly, even from session to session, this means that the signal is non-stationary. The training sets needed for calibration are also small, since they are time consuming and demanding for the subjects.

There are different types of classifiers used in BCI: generative-discriminative, support vector machines, static, stable and regularized. This report will not go in depth on how

these work. Although it should be noted that LDA or Linear Discriminant Analysis is used a lot in BCI as part of a stable solution, it has a low level of complexity and tolerate small variations in the training set.

With this project the group decided to investigate a way of dealing with pattern recognition, the simplest way being to look at two different signals coming from different mental states. Meditating with eyes closed and intensely focusing on a thought to be used as a command is a state that, in theory, should produce a significant difference in the beta band since beta waves are associated with mental alertness and focus [Teplan, 2002]. So by looking at the beta bands and quantifying the amount of beta information the group hopes to be able to differentiate the signals with a neural network.

2.4 Brain Computer Interface

The Emotiv API is declared in three header files(edk.h, EmoStateDLL.h, edkErrorCode.h) and implemented in two Windows DLLs (edk.dll and edk_utils.dll). The Emotiv EmoEngine contains the functionality provided in edk.dll. The Emotiv headset communicates with the EmoEngine through a USB device which translates the wireless data into a USB data stream. From the headset, EmoEngine receives preprocessed EEG and gyroscope data, manages user-specific settings, performs post-processing, and translates the Emotiv detection results into an EmoState. All the Emotiv API functions that retrieve or modify EmoEngine settings are prefixed with "EE_." [Emotiv,].

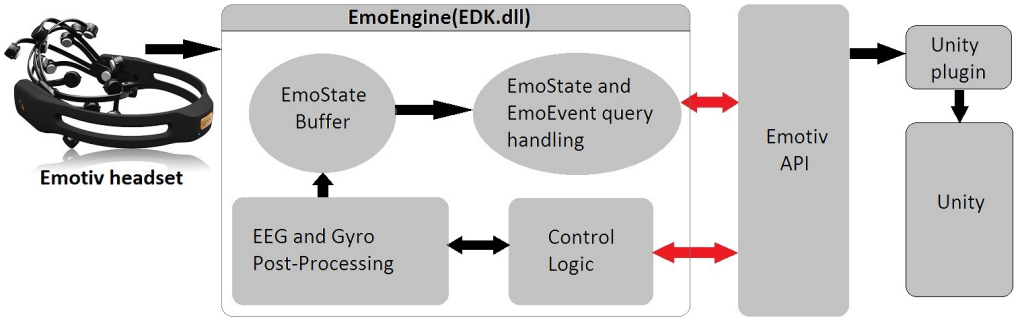


Figure 2.4: Program flow of EEG control

2.5 Artificial Intelligence

Artificial Intelligence (AI) is an attempt to simulate or make machines behave intelligently, meaning that they can make decisions or learn
[Department of Clinical Neurophysiology, 2015]. With AI in the context of this project the A* algorithm serves as the one function that makes decisions based on dynamic conditions. AI in a broader sense aims to make the most of computing by utilizing sophisticated algorithms machines can act as if they are thinking, recognize patterns and perform an array of functions that can be helpful to people or society at large [Negnevitsky, 2005].

2.5.1 A* Search Algorithm

A* (pronounced A star) is an artificially intelligent computer algorithm used in this project as a tool for pathfinding. This means creating a walkable path for game objects in which the path is defined by nodes or waypoints. Each game object that uses the algorithm will choose the shortest path from an initial starting node to a defined target. The path will go from node to node from start to finish, so by setting up the waypoints the user can define which areas of the game are walkable. The algorithm calculates all possible paths before it chooses the lowest cost alternative, meaning the shortest route.

The algorithm uses a heuristic cost function, meaning an approach to problem solving, learning, or discovery that uses a practical methodology which is not guaranteed to be optimal or perfect, but sufficient for the immediate goals. The cost function $f(x)$ is applied to a node to determine the order in which it will visit each of the other nodes in the hierarchy. This cost function is a sum of two functions $g(x)$ and $h(x)$:

- $g(x)$ represents the exact cost of the path from the initial starting node to any node x.
- $h(x)$ represents the estimated cost from node x to the goal. It doesn't know the actual distance until it finds the path, this is due to obstructive objects like walls, characters, water etc.

A* balances the two functions $g(x)$ and $h(x)$ as it moves from the initial starting node to the goal. Each time through the main loop it examines the node x that has the lowest $f(x) = g(x) + h(x)$. The heuristic function $h(x)$ tells A* an estimate of the minimum cost from any node x to the goal, which makes it important to choose a good heuristic function [Department of Clinical Neurophysiology, 2015].

What makes the A* a unique algorithm is that it also takes the distance already traveled into account and it only focuses to reach the goal node from the current node, not to reach every other nodes in the area.

2.5.2 Artificial Neural Networks

A neural network is a model of reasoning based on the human brain that has the ability to learn and adapt. This learning ability is the reason why the group chose to look into it. When the Emotiv Epoc headset is being used the software provided learns to detect your brain signals and the system gets better with time and practise. Neural networks have this ability so the group decided to use Matlabs extensive neural network toolbox to do some experiments regarding this subject.

The principles behind neural networks are based on the structure of the human brain and how all the neurons are connected, hence the name neural networks [Negnevitsky, 2005]. By using a vast number of neurons the brain is capable of computing information at a speed much greater than any computer. One neuron by itself is built in a very simple way, but a large quantity of them create massive computing potential. An artificial neural network consists of many simple neurons which are interconnected and are analogous to

the neurons in the brain. Each and every one of the neurons receive a number of input signals which go from one neuron to another, although each neuron only produces a single output signal. The network learns by using weights on the input signal and these function as long term memory in the network by strengthening the important features and letting noise (irrelevant information) fade out. These weights are continuously being updated as the system is learning. Each neuron determine its output based on a threshold mechanism, one of which is the sign function. The way it works is, if the input is greater than or equal to the threshold, then the output is 1 and if it is less then the output is -1.

There are different types of activation methods for each neuron as shown in figure 2.5: step function, sign function, sigmoid function and linear function.

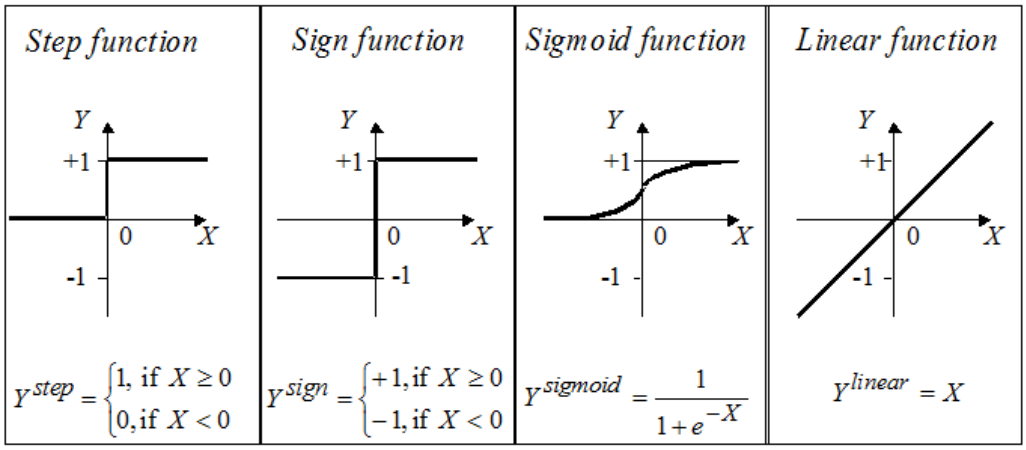


Figure 2.5: Different activation functions [Bjørlykke, oint]

The simplest form of a neural network is called a perceptron, a classifier of inputs, and it's built up by a linear combiner and a hard limiter. This hard limiter receives the sum of two inputs then forces the input to equal +1 if the input is positive, and -1 if the input is negative.

The perceptron learns by adjusting the weights, thus reducing the difference between the desired and the actual outputs. At first the weights are randomly given then they adjust to better match the training examples. All this is just one layer, by adding more layers, new possibilities arise.

Multilayer neural networks are what is called a feed-forward system of layered neurons, some of which are hidden. The primary task of these hidden neurons is to detect features, the weight of these neurons represent the hidden features in the input layer. These features are used in the output layer to determine the output pattern. There is an input layer at the start, some hidden layers following that, all finished by a computational layer at the end. This level of complexity allows for hundreds of different learning algorithms. Deciding the number of hidden layers in the system is a crucial part of the design of the neural network. Using to few can cause a problem known as underfitting, it happens when the data fed to the system is too vast for such a small number of hidden layer neurons. On the other side of the spectrum of problems that can occur, is a phenomenon not surprisingly known as overfitting, that happens when the data is not complex or rich

enough to train all the hidden neurons in the system. Another problem that can occur with using too many hidden neurons is computation time being too long. This occurs when there are too many hidden neurons in the system and that means that the number of hidden neurons has to be just right. [Negnevitsky, 2005]

3 Materials

3.1 Software

Table 3.1 present an overview of all the software equipments used during this project.

Software
Unity
MATLAB
Simulink
Dropbox
Microsoft Project Professional
LaTeX
Visual Studio
Emotiv Developers Kit (Research license)

Table 3.1: List of software used in this project

3.1.1 Unity

Unity is the game engine used for developing the environment. This has been the main program used in this project.

3.1.2 MATLAB

MATLAB (Matrix Laboratory) is a computing program with its own scripting language based on C. The program allows everything from basic to advanced matrix manipulations, plotting of functions and data, visualization and implementation of algorithms [The MathWorks Inc, 2015a].

Matlab was used to treat the raw EEG signal, and from there processed and plotted by different functions found in Matlab. The computing program also offers various toolboxes including the Neural Network toolbox used as a pattern recognition tool in the signal processing part.

3.1.3 Simulink

Simulink developed by MathWorks is a data flow graphical programming language tool for modeling, simulation and analysis of dynamical systems [The MathWorks Inc, 2015b].

Emotive offers a special input for simulink, this Epoc - Simulink EEG Importer provides the real-time EEG data acquired by the Emotiv Epoc headset to a Simulink model [Emotiv, 2014b]. Easily explained the Simulink block receives the EEG data in a vector format and it can be used for further processing in the Simulink environment and also imported into a Matlab script as has been done in this project.

3.1.4 Visual studio

Visual Studio is a development tool for creating software for Microsoft Windows [Microsoft, 2015]. In this project it was used as a development tool creating the first two methods for communication in this project.

3.1.5 Emotiv developers Kit

Emotiv developers Kit (Research license) allowing the group access to the API and all its features.

3.1.6 Microsoft Project Professional

Microsoft Project Professional is a project management tool that allows users to make detail plans for conducting a project. Microsoft Project manages the progression of the project and users can easily determine if a project is on schedule or not. In this project the program was used to generate a Gantt charts for orderly presentation of a project's progress.

3.1.7 Dropbox

By Using Dropbox made it easy for storing and sharing files within the group and the supervisors. Dropbox has also been used as an additional method for security in terms of storing important work, which has been developed throughout the project period.

3.1.8 LaTeX

LaTeXis a document preparation system, it is widely used for the communication and publication of scientific documents in many fields, physics, computer science, including mathematics, statistics, economics, and political science. [LaTeX, 2015]

In this project the group used Sharelatex, an online LaTeXeditor used by students and academics. It allows you to share documents with other people, online compiling of projects to PDF format and real time collaboration [ShareLaTeX,].

3.2 Hardware

This project is not a typical hardware project which means that it contains few physical components, table 3.2 present an overview of all the hardware devices used during this project.

Hardware
Asus G750J gaming notebook pc
Emotiv Epoc headset
Oculus Rift

Table 3.2: List of hardware used in this project

3.2.1 Emotiv Epoc

The Emotiv Epoc headset provides access to raw EEG data throughout 14 channels connected to the users head. More details about how the headset works is explained in chapter 2.3.1 EEG measurements.

3.2.2 Oculus Rift

Oculus Rift is the virtual reality headset that allows the user to be immersed in 3D space. Under is a list of the specs for the headset used in this project, Development Kit 2:

- 1080p OLED screen running at 75 Hz
- 960-by-1080 pixels per eye
- 100 degree field of view
- Accelerometer
- Gyroscope
- Weighs less than 1 lb.
- Magnetometer
- HDMI and USB cable

3.2.3 Asus Laptop

Asus G750J gaming notebook PC is the main computer used in this project. To use the training platform utilizing the Oculus Rift, you will need a relatively powerful computer. The specifications of the supplied laptop is listed below .

- Intel i7-4700HQ
- Nvidia GeForce GTX 770M
- 2x8GB 1333MHz RAM
- 17.3” Display
- 256GB SSD + 750GB HDD

4 Method

4.1 Communication

The connection between the Emotiv headset and Unity 3D is one of the most critical elements in the project to create a working, stable environment. The group therefore investigated three different approaches as a way of finding the best solution and to better understand the communication between the API and Unity 3D. Two of the options namely an UDP-server and our simple plugin were developed in Visual Studio using C#, while the third and last option is an official plugin developed by Emotiv. In this chapter the three different methods will be explained.

4.1.1 Home-made plugin

The home-made plugin was a quick investigation in the early stages of the project done both for educational purposes and as a comparison against the server-client solution created in a former project. The home-made plugin shares some features with the official plugin as it is a software library contained in one or several dll-files.

To create a Unity plugin, start a new "class library" project in Visual Studio. First of all, Visual Studio needs to be able to access both the Emotiv API and the relevant Unity libraries. A reference to the Unity libraries is needed as our project needs to inherit from the MonoBehaviour class. MonoBehaviour is the base class every Unity script derives from. To add these libraries to the project open the reference manager and select the relevant dll-files (DotNetEmotivSDK and UnityEngine). The figure below shows how this is done in the reference manager.

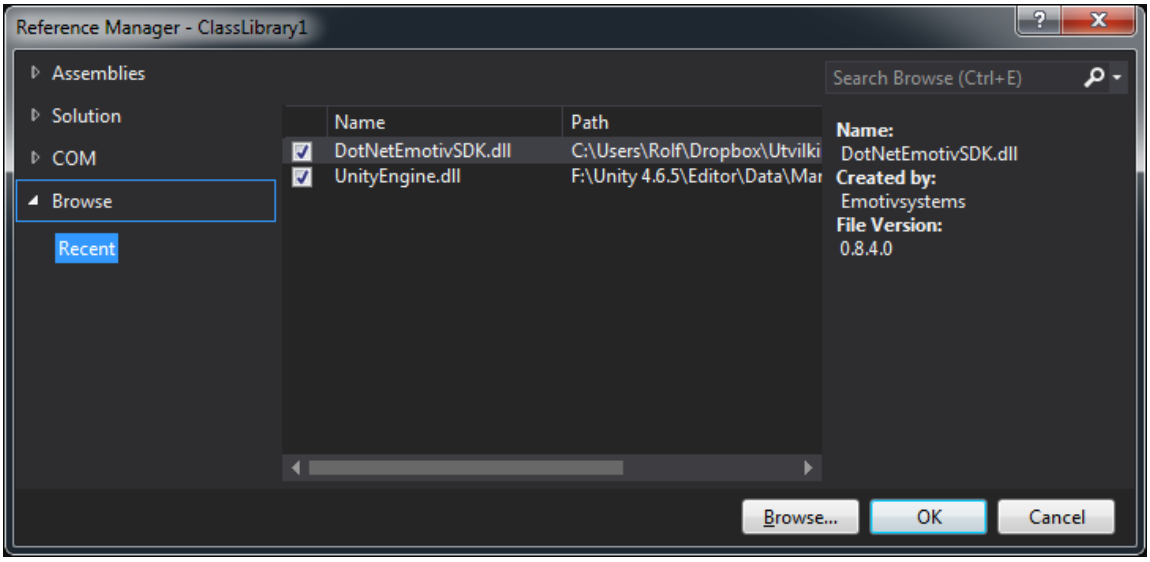


Figure 4.1: Open the reference manager by clicking Project > Reference Manager

When creating plugins it is important the project is configured properly to avoid errors when implementing our plugin. The project needs to be configured for the .NET-framework 3.5, as Unity can't use the newer versions of the .NET-framework. Furthermore, as the Emotiv API is compiled as 32-bit the group's plugin will need to be compiled for 32-bit as well. It should be noted that Unity 5 handles plugins differently from Unity 4 regarding 32/64-bit architectures.

After configuring the project as described it is possible to access the Emotiv API. In the example code shown below the Emotiv API is accessed by getting a constant from the class which can then be used within an Unity script.

Listing 4.1: Class Library Project

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Emotiv; //Namespace containing the EdkDll class
using UnityEngine; //Needed to inherit from MonoBehaviour

   public class Class1 : MonoBehaviour
    {
      // Gets a constant from the Emotiv API which Unity can now use
        public int UnknownError = EdkDll.EDK_UNKNOWN_ERROR; //Variable is
            equal to "1"
    }
```

After building the project it outputs a DLL-file. The dll-file is then copied to assets/plugins/ in the Unity project. We can then instantiate our plugin inside of a script and use its functions as shown in the code below.

Listing 4.2: Unity Script

```
using UnityEngine;
using System.Collections;

public class UsingPlugin : MonoBehaviour {

	void Start () {
		Class1 myPlugin = new Class1(); //Instantiate
		print (myPlugin.UnknownError); //Should print "1"
	}
}
```

Upon executing the script within Unity we can see that the console prints the value of the constant from the Emotiv API. The group then concluded that the basis for a home-made plugin is established and that possibilities for a fully functional plugin exists.

After expanding our horizon on how plugins and the Emotiv API works the group chose to look into a previous groups work with emphasis on their UDP-server solution.

4.1.2 Server-client solution

The server-client solution is a workaround using an UDP-server. The method of creation is similar to the previously described Unity plugin with the main exception that you would either create a "Console Application", or a more GUI-oriented "Windows Forms Application". The reason this workaround was initially developed was that when the previous group started their project in early 2011, Emotiv had no official support for Unity. And thus, the available options were either to write their own plugin similar to the later official plugin, or to create a simple server-client solution which communicated the essential variables to Unity. This method was used for a short amount of time in the starting phases of the project, before the group decided to try the official plugin.

This solution consists of a small console application in which the user must connect either to the real Emotiv headset via the Emotive Control Panel, or the "Emotive Composer" headset emulator. The program is event-based so whenever the Emotive API is updated due to the user for instance thinking "push", the variable automatically updates in the program. Having an active cognitive function such as "push" prompts the program to send the new variable to the connected client. Our implementation of the UDP server is slightly modified. The source code was modified slightly so that several cognitive functions can be distinguished. On the receiving end, a Unity script parses the cognitive function such as "push", "pull", or "left", and its magnitude into two separate variables. These variables can then be accessed through their respectable Get-methods for use in other scripts. After discovering this solution often caused Unity to crash the group investigated the difficulty of implementing the official plugin.

4.1.3 Official plugin

In March 2011 Emotiv released their Unity plugin. This plugin uses the same approach as our home-made plugin in the sense that the it consists of a library accessing the Emotiv API. The difference between the official and a home-made plugin is that the official solution offers a full integration to the Emotiv API including cognitive activity, gyroscope data, signal strength, battery level, and many more. Compared to the other alternatives, the installation is quite user friendly in the form of the usual .exe installation file. The installer extracts a unity asset containing the needed plugin files together with needed documentation, after adding said asset to your project, the procedure for implementing the plugin to the project is described in the included readme-file. After the plugin is implemented, its use is quite straight forward. Below some example code is shown. The code snippet below shows how one can get the current active cognitive function such as "push", "pull" etc.

Listing 4.3: Example usage of EmoCognitiv.cs

```
//sets power and name of active action
	void CognitivActionUpdate(){
		int count=0;
		foreach (float f in EmoCognitiv.CognitivActionPower)
		{
			//only the active action can be over 0
			if(f>0f)
			{
				CurrentCognitivPower = f;
				CurrentCognitivAction =
				EmoCognitiv.cognitivActionList[count].ToString();
			}
			else
			{
				count+=1;
			}
		}
	}
```

4.2 Unity3D Environment

Unity provides an easy to learn approach to game development and with no prior game development experience this game engine became a natural choice. The actual 3D models and textures have all been imported from other sources most of which are from Unity Asset Store where a vast array of different textures and models are provided. Hence drag and drop, simple scripting, stretching and placing have been the hallmarks of the design process. There's a lot of tutorials online on learning how to create a Unity game. This chapter will describe a bit on how the environment is made, and what ideas lie beneath the surface. It should be said at the the whole environment design process is basically

characterized by finding useful building blocks, be it models, animations, textures and meshes, to later assemble them almost like set of legos. The walking characters on level 5 serve as a prime example on the design process. 3D models of samurais has been found online with attached walking animations, the game object makes use of the A* algorithm and waypoints through Unity 4's character controller system to move around. T he method behind the environment design can be summed up by basically saying that it's gathering and assembly of game objects and components using tools provided in Unity's development platform.

In addition to this, the game engine provides a decent physics engine, game objects can have a component attached called a rigidbody. This gives the object physical properties like mass, gravity, forces can be applied to it, etc. Another useful bit worth mentioning is the collider system, colliders serve as boundaries for the objects that has the colliders attached to them. It can have the property of serving as the physical connection point with other colliders in the game, for instance the ground, a wall or a tree and this way objects don't fall into each other. They can also be used as triggers, then they become invisible and objects can move through them, a script can then be added to it initiating a command whenever a specific object touches it. It is used to make the environment interactive and it gives a certain intelligence to the game world, the group used them as timers, collision detection and setting off explosions. The group also provided a version of the game that is compatible with Oculus Rift, this might give the user a more realistic experience.

4.2.1 Level design

The game world is an open environment design with game objects and terrain design signifying the boundaries of where the player can move. When in the game the group wanted a design that would provide an intuitive understanding of what to do and where to go. The game world is built on a game object called a terrain which is a big 3-dimensional surface that can be mended and shaped into a natural looking terrain. Unity has a built in terrain editor that allow the designer to easily add textures, grass, trees which in turn makes the design process very simple. By shaping the terrain and adding various game objects like boxes, buildings and piles of rock the boundaries in the game become very natural, the group did not want to use too much invisible walls to set boundaries for the player.

4.2.2 Components

When it comes to the design of the components in the game it's fairly straightforward. Level 1 wants to focus on training the move forward command, the level is then built like a drag race track with a start and a finish line separated by a piece of tarmac. There is a line of cubes with colliders attached. Surrounding the level is an invisible box with a trigger collider, triggers can have a script attached to them and these scripts can gather information on whether another object is present within the boundaries of the trigger.

Different methods like OnTriggerStay(), OnTriggerEneter() are used to initiate different behaviour.

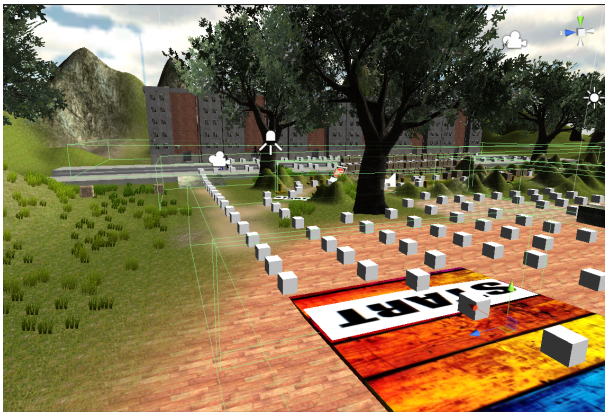


Figure 4.2: Invisible boxes acting as triggers

The box is then tagged in a way that allow a script to access information on what level the box is associated with. This way the GUI displays information on where the player is located and what to do.

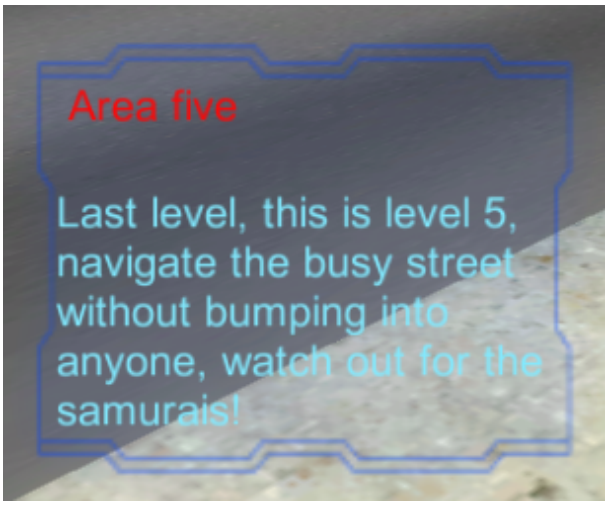


Figure 4.3: Main GUI giving essential information to the trainee

There are five of these boxes located in various places around the game environment corresponding to level 1 through 5. There are also triggers located at the start and end of the line of tarmac on level 1, corresponding to the start and finish line. These triggers are used in a stopwatch script that gives the user feedback on how much time it took them to finish the level. The best time is stored and whenever a new best time is set the user is notified through the GUI. On top of that the main part of the GUI also changes when a timer is being initiated telling the user that the timer has started and that they should get to work. This same principle of design is used in all levels with the exception being level 5 which act more as a playground where the trainee would try to avoid bumping into the randomly moving game objects present.

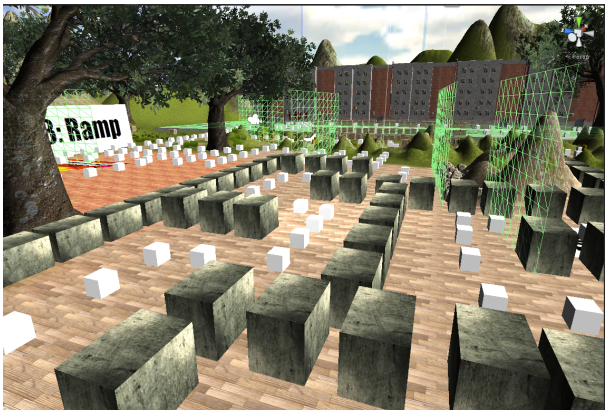


Figure 4.4: Invisible walls

These walls exist whenever a timer is active and it makes sure the user cannot do the levels backwards. This would stop timers before they start and all in all create an unwanted amount of bugs and unwanted game behavior. In addition whenever the user experiences the inability to move it is because they are not supposed to.

4.2.3 The player

With the design of the wheelchair the group wanted a somewhat realistic approach to how an actual wheelchair would behave. Unity has a collision object called wheelcolliders, four of these are placed on the wheelchair. The actual 3D model of the wheelchair was found online.

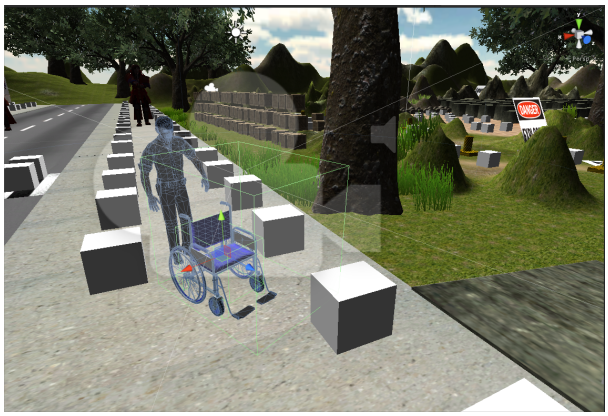


Figure 4.5: The player with the triggers and colliders

They are basically circular objects without width that act as the wheelchairs connecting point with the ground where scripts can be attached and through that apply a force on the axis of the wheels.

4.2.4 Special features

There are some safety issues that have been addressed such as when the wheelchair resides on a slope a safety brake is engaged if no command is present. This ensures that the wheelchair will not start to roll when on a slope and this feature is highly valuable on level 3 where the purpose is moving up and down on a slope. In addition an invisible box is placed around the player that works as a collision avoidance device, simulating laser or other forms for distance detectors. Whenever a game object with the tag "collidable" enters the box, the safety brakes are engaged. The autopilot system will be described in another chapter, but it should be mentioned here that it makes use of waypoints.

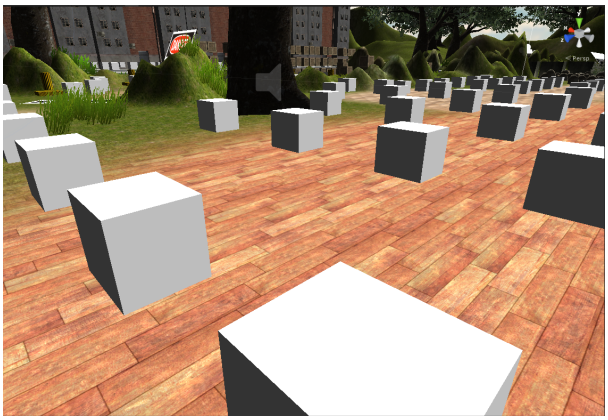


Figure 4.6: Waypoints scattered around

These are game objects placed around the environment that act as target points which an algorithms calculates a route through.

Figure 4.7: Birds eye view of the waypoint layout

The vectors describing the route are then used as a force vector pushing the wheelchair around until it gets to desired location.

Level 4 is the only level where doing a mistake can potentially be hazardous, this is to see whether stressing the trainee has an effect on their performance. Level four is built like

a dirt road with cones scattered in such a way that the player knows where to go. If the player comes in contact with any of the cones, a big explosion is set off which prompts the game to reload thereby loosing the user's best score.

4.3 Implementing an Autopilot

In order to create an autopilot a combination of a pathfinding algorithm along with a user friendly GUI is required. With Unity 3D introducing their new GUI system in version 4.6, a major part of the autopilot was simplified. This meant that only the implementation of a path finding algorithm remained.

The implementation of the pathfinding algorithm is done by installing a plugin made by Aron Granberg [Granberg,]. The installation of the plugin is done either by adding a Unity asset downloaded via Granbergs website, or via the Unity asset store. After the installation the plugin can then be accessed programmatically through scripts in Unity, or manually through UI elements in Unity. An example demonstrating a pedestrian is provided in the "AI" chapter in Listing 4.13.

4.3.1 User interface for autopilot

An Autopilot GUI is always present and is activated by thinking "pull" then "left" or "right" and selects the desired location whereas "push" engages the autopilot. Thinking "pull" thereafter deactivates the autopilot GUI, and "pull" works as a toggle command activating and deactivating the autopilot.

By using the built-in GUI system provided by Unity the group built an autopilot system. As described above, the fourth command acts as a toggle between manual and auto drive. To continuously move the wheelchair the user would normally have to focus intensely for a prolonged period of time which can be very strenuous. Autopilot can in this way help by having some preset common destinations that can be selected. The waypoint system works by having multiple game objects placed in the environment, and these objects are used in a script that identifies them as reference points used to calculate a path using the A* algorithm. You don't see these points while playing the game because another script hides them.

4.4 Signal Processing

In the signal processing part of this project the group decided to do an experiment with the use of raw EEG data by taking 200 raw EEG datasets comprised of two different cognitive states. The first 100 datasets were sampled while the user was meditating with closed eyes, trying to become completely relaxed. The other 100 datasets were sampled thinking the cognitive action "push" which in many ways can be considered a polar opposite to "meditation". In this context thinking "push" means to concentrate and

try to visualize a specific picture that relates to the task. In this case the user needs to visualize the the specific picture to correlate as a move forward command. During the experiment the user controlled the wheelchair in Unity to ensure that the correct cognitive state was active, meaning that if the wheelchair was not moving the "push" sample would be discarded. Measuring the quality of meditation is a little trickier because the nature of high quality meditation remains subjective. As long as the eyes are closed less beta activity in the brain is expected. Each dataset was sampled with a duration of 10 seconds, all done by the same person during the same day which makes the datasets as accurate and reliable as possible.

By using Matlab the group processed the data further and used it to train an artificial neural network (ANN). This network was supposed to decide which of the data was either "meditation" or "push" in order to classify.

4.4.1 Raw EEG signal

Simulink EEG Importer is a toolbox designed to transfer raw EEG data in real-time. This makes it possible to analyze and process ones own EEG data. The signal-server demonstrated in Figure 4.8 established communication between the Simulink EEG Importer and the Emotiv Epoc headset [Xcessity Software Solutions,]. Clicking on the "start" button on the signal-server after turning the Emotiv Epoc on allows the user to measure raw EEG data in real-time through the Emotiv toolbox in Simulink.

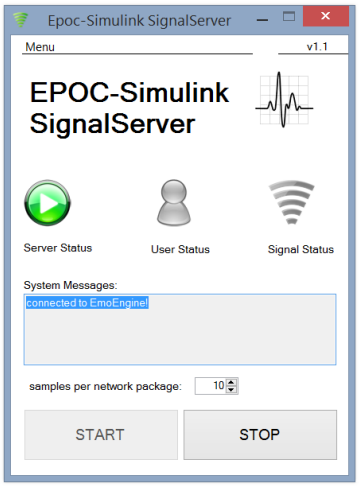


Figure 4.8: Simulink signalserver

The EmotivEpocEEG block shown in Figure 4.9 receives the EEG data as a vector. The Simulink block is created out of a Matlab mex S-Function and has a double[22] data vector as output. Each call to the Simulink block returns with a data sample including the following data: COUNTER, $AF3$, $F7$, $F3$, $FC5$, $T7$, $P7$, $O1$, $O2$, $P8$, $T8$, $FC6$, $F4$, $F8$, $AF4$, GYROX, GYROY, TIMESTAMP, FUNC_ID, FUNC_VALUE, MARKER, SYNC_SIGNAL [Xcessity Software Solutions,]. The scope block shown in Figure 4.9 is reading out the raw EEG signals in real-time.

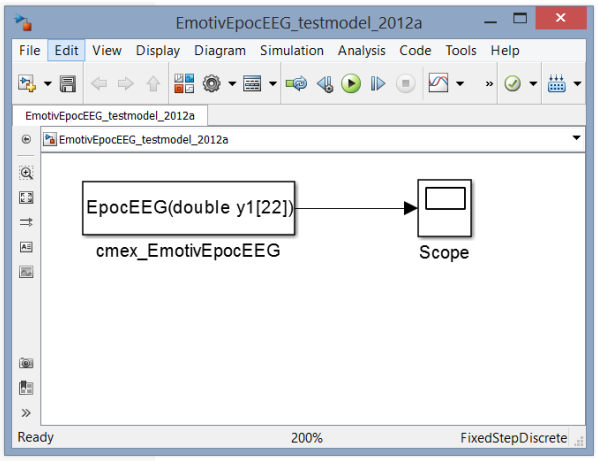


Figure 4.9: Simulink EEG importer

Figure 4.10 demonstrates how the raw EEG data is presented visually with time in seconds represented on the horizontal axis and μV on the vertical axis. The user can choose how many seconds each data set should contain, and in this case it was used 10 seconds on each dataset which resulted in 1280 samples. From the scope block data can be directly transmitted as a 1281x23 matrix to Workspace in Matlab. Saving the datasets in the workspace creates a struct datatype containing 200 fields which can later be processed.

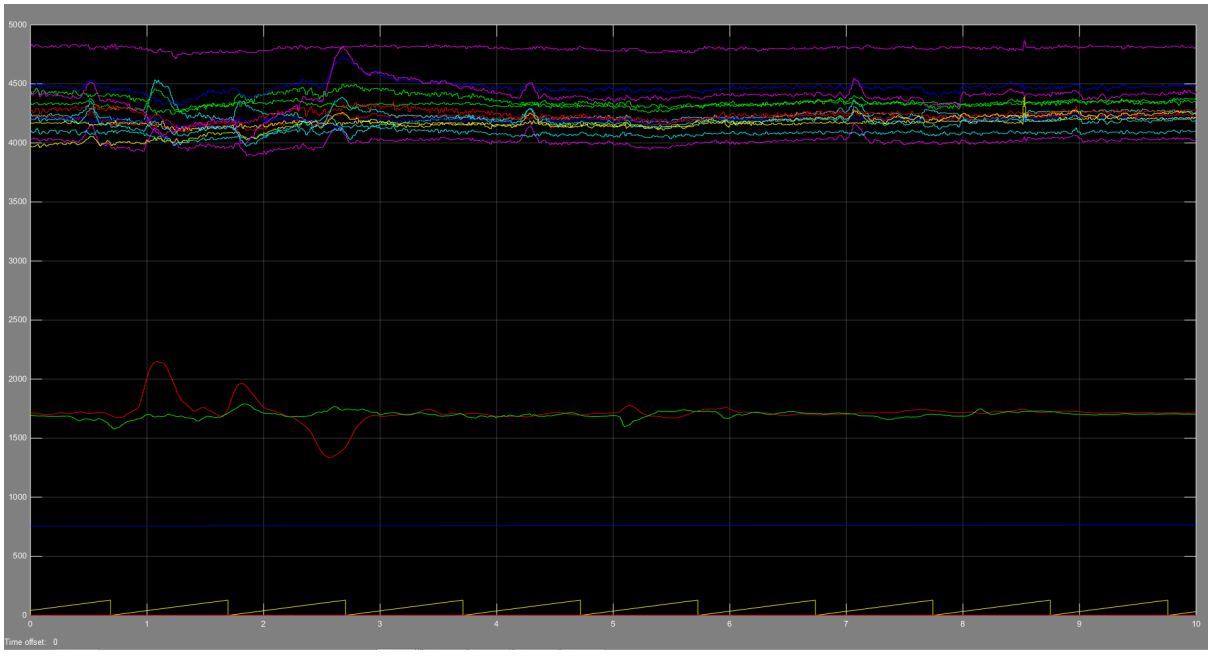


Figure 4.10: Scope displaying 10 seconds of real-time EEG data

Before the data can be passed to a neural network, processing of the raw EEG data into the frequency plane is needed. By doing this the frequencies can easily be extracted from the signal. The general process can be described as such:

1. Import datasets for processing

2. Extract relevant datafields
3. Find ΔV between electrode pairs
4. Remove DC-offset from all channels
5. Transform to frequency plane via FFT
6. Assemble the input matrix for the neural network

The datasets are saved in a struct of 200 individual fields and within each field is a 1281x23 matrix. By looping through all 200 fields one by one the required steps are performed before progressing to the next field. As the list above describes the potential between the electrode pairs must be found. By comparing the data to a Emotiv-supplied file [Jacob Thomas Redmond, 2013] meant for EEGLab identifying the 14 signal pairs is made easy and can be done without manual testing.

Listing 4.4: psudo matlab code from RawToNNet.m

```
S=load('matlab.mat');
target = [ones(1,100) zeros(1,100) ; zeros(1,100) ones(1,100)]; //Build
    target vector for neural network

for f = fieldnames(S)
channelRAW=S.(f{1})(1:1280, 3:16); //extract datachannels 3 to 16

    //Find potential between electrode pairs
    for i = 1:7
        channel(:,i) = channelRAW(:,i)-channelRAW(:,15-i);
    end

    //Remove DC from all channels by subtracting mean
    for i=1:7
        channel(:,i)=channel(:,i)-mean(channel(:,i));
    end

    // FFT
    FFTchannel = abs(fft(channel, N));

    //Build input vector for the neural network
    for i=1:7
        input(i, currentDataSet) = FFTchannel(k, i); //single frequency
        inputMean(i, currentDataSet) = mean(FFTchannel(deltaL:deltaH, i));
            //mean frequency between deltaL and deltaH
    end
    //Repeat above loop for every frequency/frequencyband

currentDataSet=currentDataSet+1;
end
```

4.4.2 Data verification

As an intermediate step between measurement and classification of the data, it is necessary to verify that the acquired data is valid. One way of doing this is to plot the data in a way they can be compared against each other. By using a scatter plot the data can be plotted against each other as data points of the "push" values and "meditation" values. By inspecting the data in a scatter plot a comparison of the two sets of data can be done to check for inequalities. When looking at the data, ideally, one would expect to see two separate clusters. Hence with the scatter-plots generated at 20Hz (beta), the group expected to see two clusters of data points at the low end of the "meditation" data and at the high end of the "push" data. As one can see in figure 4.11 the two cognitive states does not show itself to be easily distinguishable. However, there is an observable trend between all the plots. The "meditation" sets generally have lower magnitude than the "push" data sets, which coincides with the fact that beta waves are related to cognitive activity.

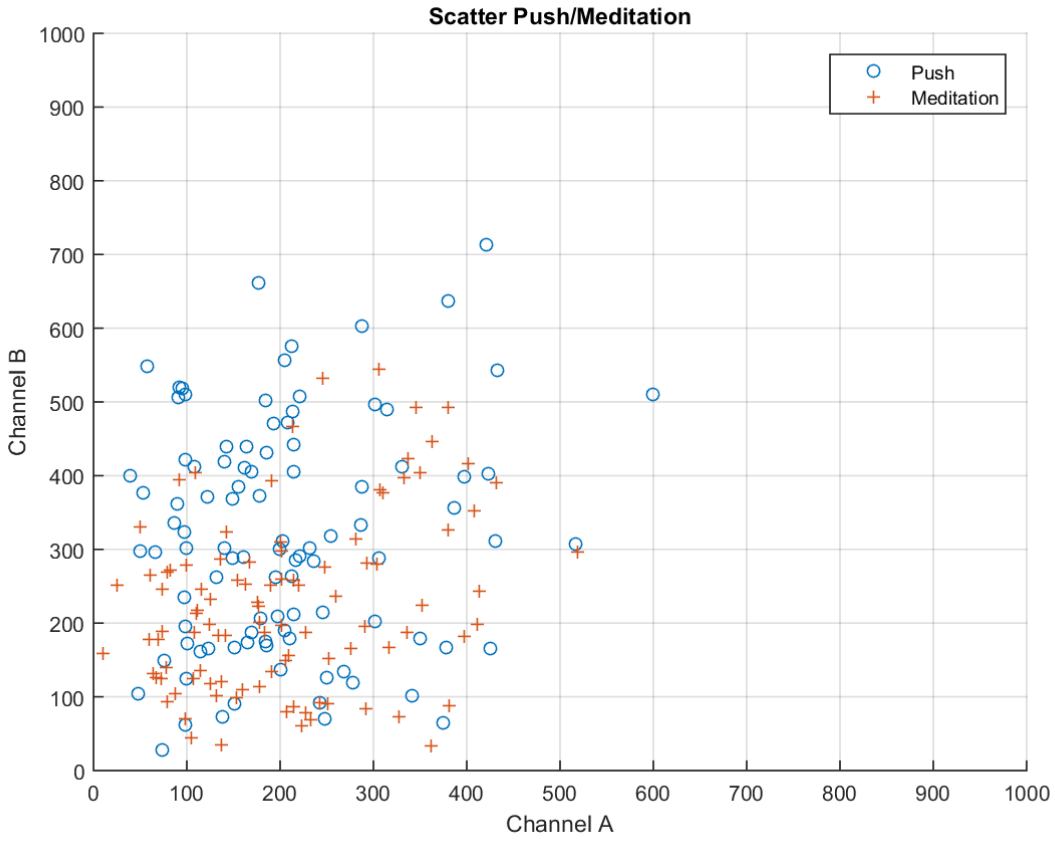


Figure 4.11: Scatter plot of channel 4 and 7 at 20Hz

4.4.3 Classification

Using the built-in neural network tool in Matlab (nprtool) [MathWorks, 2015], attempts were made to classify the collected EEG-data into the two initial groups of data: "push" and "meditation".

Having the recorded EEG-data in the frequency plane makes it easy to view the magnitude of a specific frequency. The first approach was to use 20Hz as the only defining feature, giving us an input matrix of 7×200 values. Using a single value in the beta band proved to be only a little bit more effective than random guessing in terms of data classification. As the 7×200 approach with a single frequency as input vector proved insufficient and further investigation was needed to determine what to use in the input matrix. Figure 4.12 illustrates what values were initially used in the ANN input matrix.

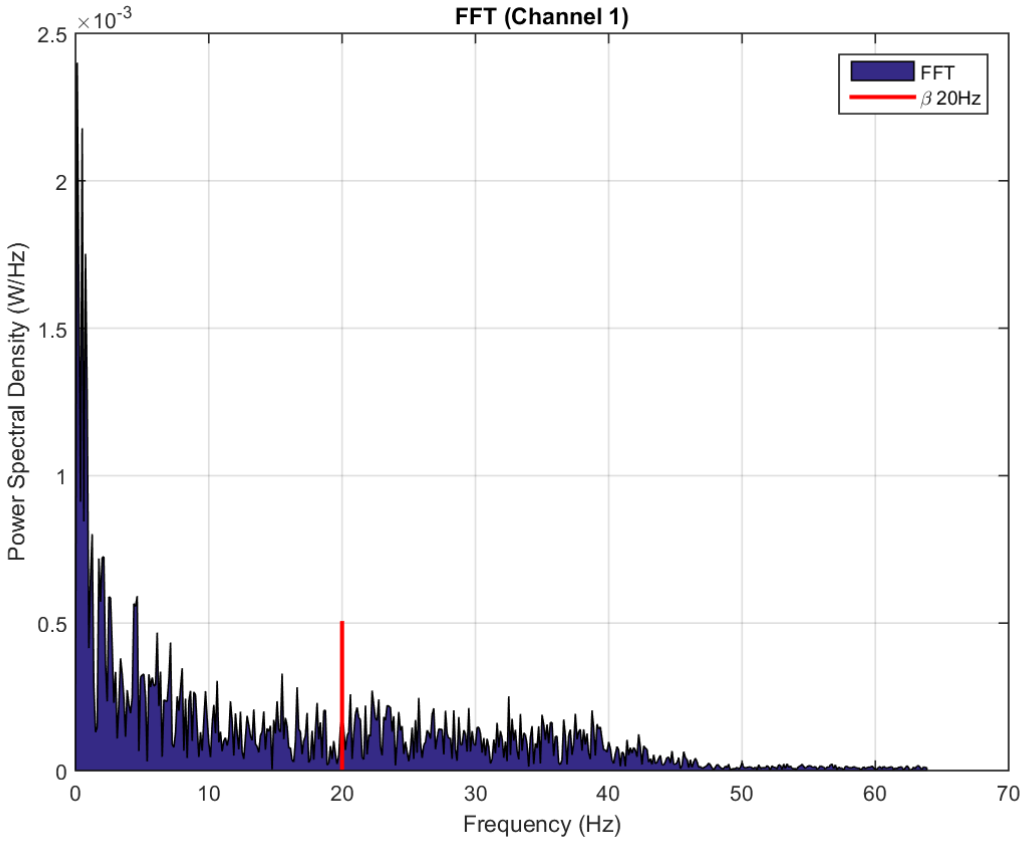


Figure 4.12: Illustration of ANN input data using only 20Hz

Due to the initial attempt of configuring the network having limited success it was decided that expansion of the input matrix was needed, expanding from 7×200 to a larger matrix by increasing the number of features. Instead using 6 frequencies, $6 \times 7 channels$ produced a number of features which were used and resulted in a larger 42×200 input matrix. Figure 4.13 illustrates using 8, 13, 16, 20, 25 and 30Hz. This method seemed to produce better results than the initial method of using a single frequency. Evaluating the performance

of the neural network can be hard due to a relatively low sample size. After the neural network is done training itself it evaluates its own performance using a set percentage of the 200 datasets. If this percentage is for example 10%, the error estimator only has $N = 20$ sets of data to work with. This in conjunction with random initialization values in the ANN makes the estimator have a high variance making it hard to objectively decide if one network configuration is better than the other. However, as the error estimation seemed to be generally lower than that of the single-frequency input, it was natural to experiment further using mean values.

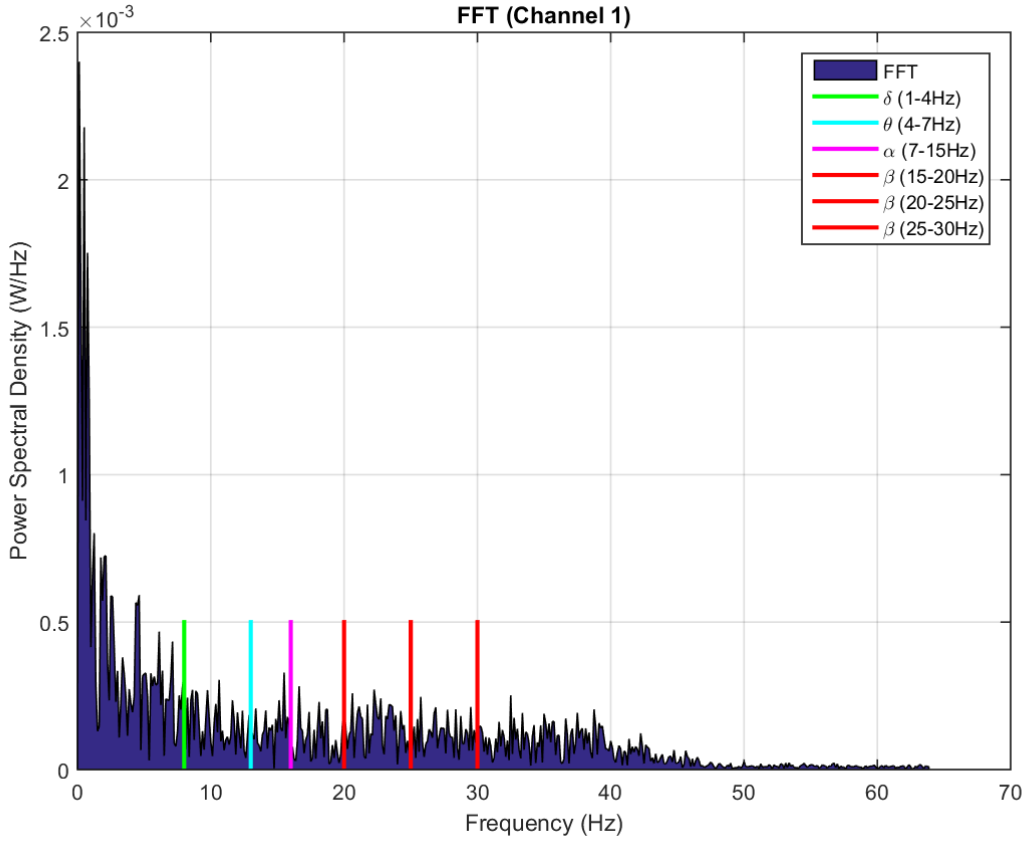


Figure 4.13: Illustration of ANN input data using several values

The previous experiment using 6 separate frequencies seemed to point in the right direction in terms of classification error. In an attempt to further reduce the networks classification error the mean values of 6 frequency bands were used as input. As a result, the input matrix consisted of the mean value of δ, θ, α, β_{LOW}, β_{MEDIUM}, β_{HIGH}. As an attempt to reduce loss of information by averaging values, the beta band was further divided into three sub-bands.

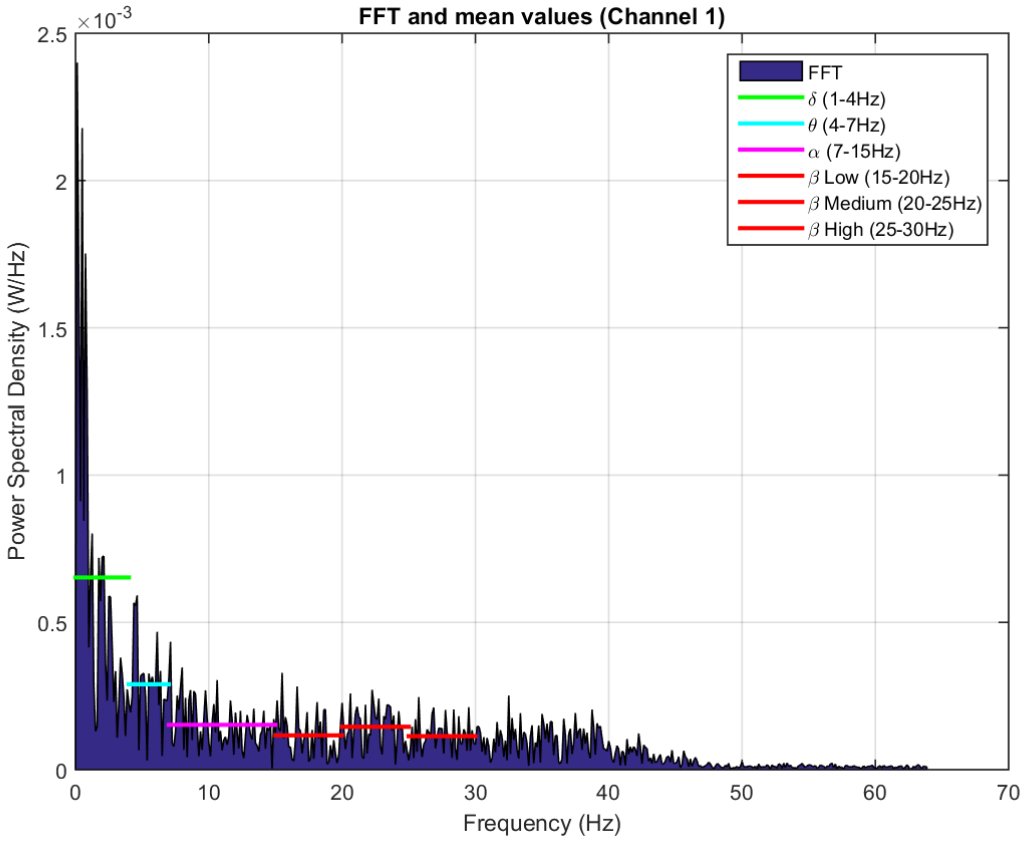


Figure 4.14: Illustration of ANN input data using mean values

Using several mean values in the input matrix the tools error estimator consistently returned classification errors ranging between 0-4% of the tested data sets, depending on the initialization values of the neural network.

There is no method of calculating the exact number of neurons in the hidden layer, but there are many rule-of-thumb methods which can help to find a starting point based on the size of input and the output of the neural network. The group decided to use the three rules-of-thumb listed below [Heaton Research, 2008]:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.

By using 2/3 the size of the input layer, plus the size of the output layer gives 30 hidden neurons. The group started with 30 hidden neurons as a reference point and tested the neural network with numbers close to 30 and ended up using 21 hidden neurons which gave the best result. In the creation of a neural network the user is prompted to choose how many datasets should be used in training, validation, and testing. Due to a relatively

low sample size it is important to choose a large enough percentage of data for testing. This is due to the fact that an error estimator will choose a certain percentage of data and run tests calculating the classification error for the network. If the amount of data set aside for testing is insufficient the variance of the estimator will be high due to low sample size. In this case the standard 15% was sufficient to consistently determine the performance of the network. After designing the input, choosing the number of neurons in the hidden layer, and the size of the output, the toolbox trained its neurons so that it would produce the correct output.

The final structure of the neural networks input matrix can be seen in figure 4.15. The figure demonstrates a graphical representation of the neural network, where the input consists of a matrix of size 42×200, 21 presents the number of neurons in the hidden layer, and the output consists of the two different cognitive functions "meditation" and "push".

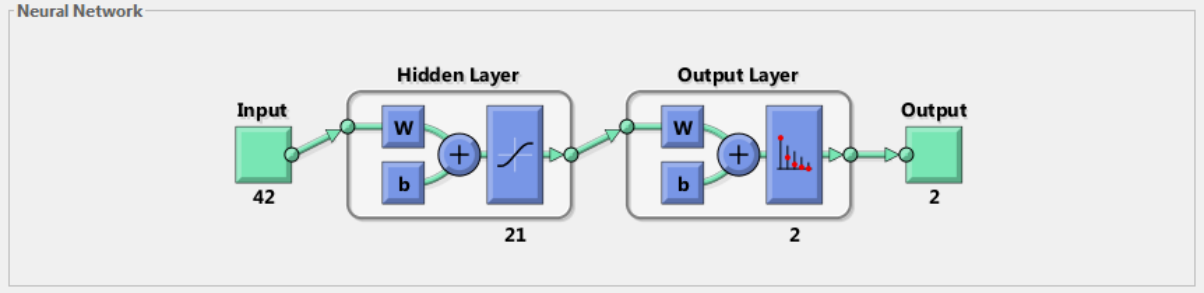


Figure 4.15: Final structure of the Neural Network

Figure 4.16 demonstrates the input matrix of the neural network. Each column represent one dataset where "meditation" is the first 100 columns and "push" being the last 100 columns. The channels for each of the frequency ranges $(\delta, \theta, \alpha, \beta_{LOW}, \beta_{MEDIUM}, \beta_{HIGH})$ are represented as rows in the matrix, resulting in 42 rows by 200 columns.

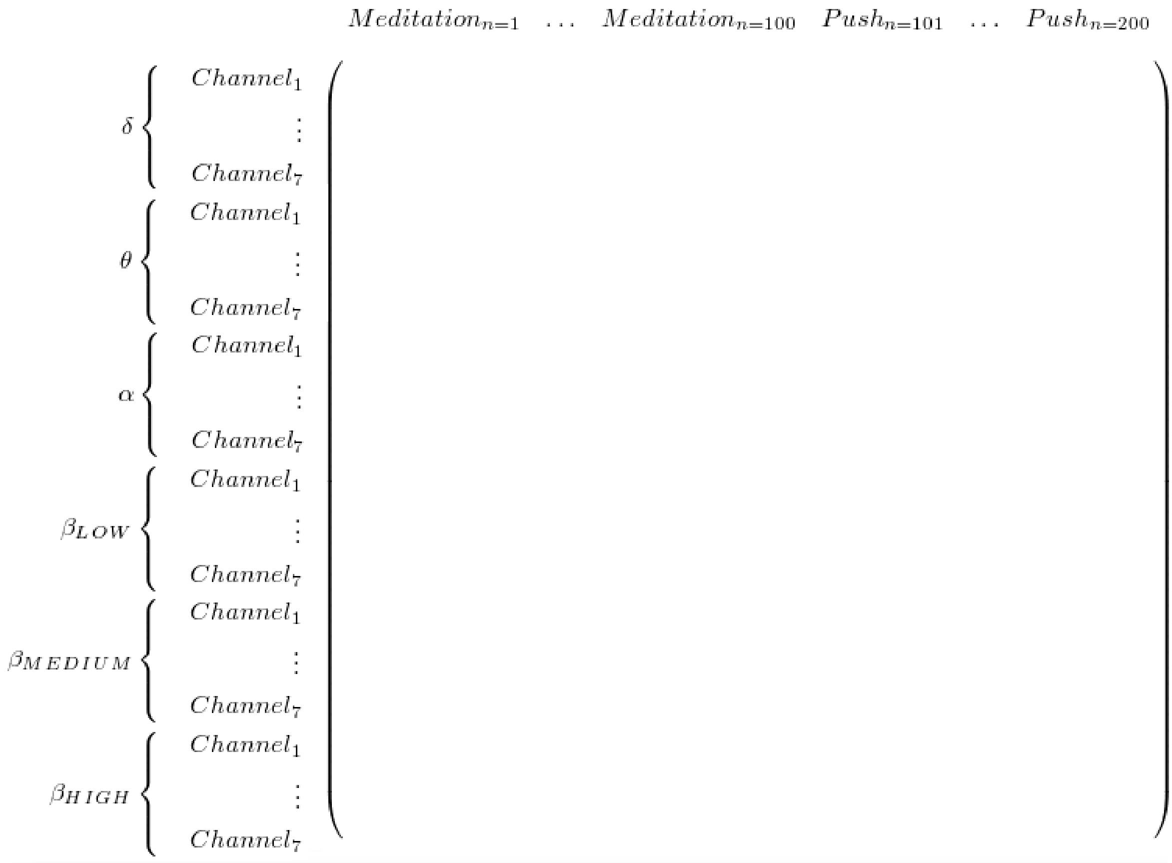


Figure 4.16: The input matrix for the neural network

4.5 Important Scripts

The behaviour found in the game is governed and controlled by scripts, these are written in C#. Each script is responsible for a specific behaviour or category of behaviours, some stand alone, others are tied together to create the desired features.

4.5.1 Control

InputHandler.cs is the script that takes in information from both the keyboard and the Emotiv API and the inputs are used to control the wheels of the chair. This creates the desired behaviour regarding movement. The first thing the script does is to find all the wheels of the chair, in other words initializing the necessary components. This is shortly followed by looping through the methods responsible for checking the inputs from either the headset or the keyboard.

Listing 4.5: InputHandler.cs

```
void Start ()
{
                WheelCollidersRR = GameObject.Find
                    ("Player/Character/BoxCollider/WheelCollidersRR")
                .GetComponent<WheelCollider> ();
                WheelCollidersLR = GameObject.Find
                    ("Player/Character/BoxCollider/WheelCollidersLR")
                .GetComponent<WheelCollider> ();
                WheelCollidersRF = GameObject.Find
                    ("Player/Character/BoxCollider/WheelCollidersRF")
                .GetComponent<WheelCollider> ();
                WheelCollidersLF = GameObject.Find
                    ("PlayerCharacter/BoxCollider/WheelCollidersLF")
                .GetComponent<WheelCollider> ();
                taggedTargets = GameObject.Find
                    ("TaggedTargets").GetComponentsInChildren<Transform> ();
                seeker = GetComponent<Seeker> ();
                treshold = 0.05f;
                forwardGain = 50;
}
```

The main loop in the script checks the inputs by the methods EmotivInput() and KeyboardInput(). The autopilots override this through an if - statement like this. The autopilot makes use of a different control approach than the keyboard control, instead of applying a force to the wheel axis, an invisible force is applied to the wheelchair itself. It works as if a rope is pulling it through the course.

Listing 4.6: InputHandler.cs

```
void Update ()
       {
                EmotivInput ();
                KeyboardInput ();
                if (!IsAutopilot) {
                        WheelchairRotation ();       //steering
                        WheelchairForward (); //apply torque
                }
       }
```

Through the plugin the headset control looks like this where h and v represent the horizontal and vertical axis on the keyboard.

Listing 4.7: InputHandler.cs

```
float CognitivPush = EmoCognitiv.CognitivActionPower [1];  //push
                float CognitivLeft = EmoCognitiv.CognitivActionPower [5];
                    //left
                float CognitivRight = EmoCognitiv.CognitivActionPower [6];
                    //right

                if (CognitivPush > treshold) { //push
                        v = CognitivPush * 10;
                }

                if (CognitivLeft > treshold) { // left
                        h = -CognitivLeft * 10;
                }

                if (CognitivRight > treshold) { // right
                        h = CognitivRight * 10;
                }
```

4.5.2 Timer System

Three scripts are running together to make the timer system work properly, StopWatch.cs, HighScore.cs and Timer.cs. The StopWatch.cs script is responsible for making sure the GUI is updated when the timer starts and counting the time as the timer is ticking. The seconds are displayed in the header of the GUI while the info box gives the user information on what to do. The update function looks like Listing 4.8. Afterwards the time is sent to the HighScore.cs script.

Listing 4.8: StopWatch.cs

```
void Update () {
                if(timerStarted){
                        time = time + Time.deltaTime;
                        headerText.displayTime((int)time);
                        info.setInfoText("Timer started, get to the finish line
                            as quickly as possible");
                }
        }
```

Timer.cs is attached to the triggers placed in the environment which are responsible for detecting whenever the player passes the trigger and when that happens it calls the StopWatch.cs script to start the timer. The different triggers are tagged either as a type "on" or "off". Unity has a method called OnTriggerEnter(), which is used to script the behaviour when entering a trigger.

Listing 4.9: Timer.cs

```
void OnTriggerEnter(Collider other){

        if ((other.gameObject.tag == "Player")) {
                findMyAreaTag();
                if(type.Equals("On")){
                        scoreSystem.setLevel(tag);
                        watch.StartTimer();
                }
                if(type.Equals("Off")){
                        scoreSystem.setLevel(tag);
                        watch.StopTimer();
                }
        }
}
```

HighScore.cs is responsible for determining which area the current time belongs to, whether it's level 1 or 2 and so on. It also has the task of determining whether the current score is better than the previous scores. When the user surpasses the best time a notification appears up in the header. Listing 4.10 demonstrates how a switch case is used.

Listing 4.10: HighScore.cs

```
        switch(level){
        case 1:
                if(time<bestTime1){
                        bestTime1 = time;
                        headerText.newBestTime("Best! : " +
                            bestTime1.ToString());
                        info.setInfoText("New best time on level " +
                            level);
                }
                break;
        case 2:
                if(time<bestTime2){
                        bestTime2 = time;
                        headerText.newBestTime("Best! : " +
                            bestTime2.ToString());
                        info.setInfoText("New best time on level " +
                            level);
                }
```

4.5.3 Area Detection and GUI Scripts

As mentioned, the game environment is divided into five different areas, each one corresponding to one of the five different levels. The TriggerHandler.cs script is responsible for detecting where the player is currently located and the GUI uses this information to figure out what information to display to the player. The TriggerHandler.cs script finds and stores the reference to the GUI game objects thus getting access to the scripts associated with them. This allows for the TriggerHandler.cs script to pass the "areaCode"-variable to those scripts which in turn display information to the player accordingly.

Listing 4.11: TriggerHandler.cs

```
    void Start(){

            findMyAreaCode();
            info = GameObject.Find("Canvas/Information/Info")
            .GetComponent<InfoScript>();
            header = GameObject.Find
                ("Canvas/Information/Header").GetComponent<HeaderScript>();
    }
    void OnTriggerEnter(Collider other){
            if(other.gameObject.tag == "Player"){
                    info.setAreaCode(myArea);
                    header.setAreaCode(myArea);
            }
    }
```

The HeaderScript.cs and InfoScript.cs simply store the text that should be displayed to the player as strings. The "areaCode"-variable is gathered from the TriggerHandler.cs script and used in a switch case that determines what information to display.

Listing 4.12: HeaderScript.cs

```
private string areaOne = "Area one";
    private string areaTwo = "Area two";
    etc.
    /*
     * Sets a new area code if the player enters a new area,
     * this is then displayed in the header.
     * This method is called by the TriggerHandler script.
     */
    public void setAreaCode(int n){
            if(n == areaCode){
                    return;
            }
            areaCode = n;
            if(!stopwatch.getTimerStarted()){
                    switch(areaCode){
                    case 1 :
                            print (areaOne);
```

```
                        headerText.text = areaOne;
                        break;
                case 2 :
                        print (areaTwo);
                        headerText.text = areaTwo;
                        break;
                etc.
                        }
```

4.5.4 AI

There are a lot of scripts responsible for the AI in the game such as the walking pedestrians and the autopilot system. These scripts have been imported from an Internet source and have been from there incorporated into the project. Two scripts are written by this group, AIpather.cs and AIpederstrian.cs, and they make use of the A* algorithm in the context of this game. Listing 4.13 demonstrates how the path is calculated in the code written by the group.

Listing 4.13: AIPedestrian.cs

```
if(path == null){
                        return;
                }
                if(currentWaypoint >= path.vectorPath.Count){
                        return;
                }
                prevLoc = curLoc;
                curLoc = transform.position;
                Vector3 rotVec = prevLoc-curLoc;
                transform.rotation = Quaternion.Lerp (transform.rotation,
                    Quaternion.LookRotation(rotVec*rotMod), Time.fixedDeltaTime
                    * lookSpeed); //rotate to heading direction

                Vector3 dir =
                    (path.vectorPath[currentWaypoint]-transform.position).normalized
                    * speed * Time.fixedDeltaTime;
                characterController.SimpleMove(dir);
                if(Vector3.Distance(transform.position,
                    path.vectorPath[currentWaypoint]) < maxWaypointDistance){
                        currentWaypoint++;
                        print (currentWaypoint);
                }
                if(Vector3.Distance(transform.position,
                    curTarget.transform.position) < maxWaypointDistance+0.7f){
                        newTarget();
                }
```

5 Results

5.1 The Levels Within the Environment

Having completed the main part of this project the result is an open world game environment where a player can navigate through various levels using the Emotiv Epoc headset for the purpose of helping train the user's skill in the various cognitive functions required. The cognitive functions required for each level vary, for instance at the start of the environment at level 1 the player only needs to use "push" as a cognitive command to complete a dragrace, but later in the following levels different cognitive commands and combinations are required. This chapter explains what each level consists of and its intended purpose.

5.1.1 Level 1

At level 1 the user finds himself placed in front of a drag race track (figure 5.1). The GUI will inform the user to move towards the start line. When the user crosses it the GUI information block changes to "Timer started, get to the finish line as quickly as possible". The GUI header changes from "Level 1" to a timer counting the seconds after crossing the start line. Now the user has to focus on the pre-calibrated "push" command to move forward. When crossing the finish line for the first time the header changes to inform the user that this was his best score so far. To try again the user could either move manually to the start line, or the user could simply hit "1" on the keyboard. It is not yet possible to review a user's high score but the game informs the user every time a high score is beaten. If you try to take a right from the start position and drive alongside the dirt road next to the track you would hit an invisible wall. This is to avoid the user accidentally turning off any timer before the corresponding "timer start"-trigger has been activated. Also if the user tries to get back to the start position by doing the drag race track in reverse, the user would be stopped by an invisible wall for the same reason.

Figure 5.1: Level 1, dragrace

5.1.2 Level 2

Level 2 is designed to train the user's ability to switch between commands, in this case the commands are "left", "right", and "push". The level is designed as a narrow track separated by steel blocks on a parquet floor. The track is quite long and can be quite tiring to train in. The same mechanics present in level 1 apply in level 2, which means it is not possible to do the track in reverse. If the user wishes to try again the user would have to push the "2" button on the keyboard in order to return to the starting line.

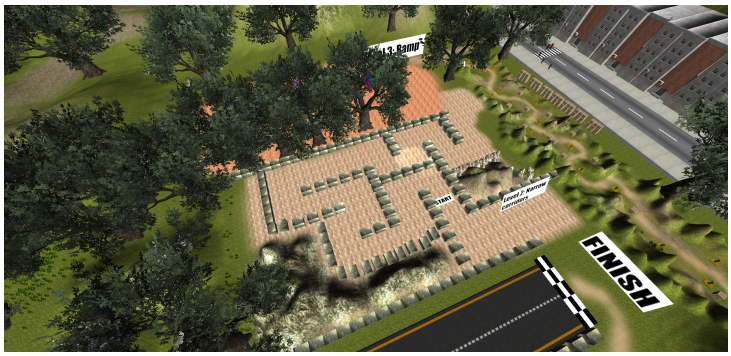


Figure 5.2: Level 2, labyrinth

5.1.3 Level 3

Level 3 demonstrates one of the safety mechanisms built into the control system. The wheelchair knows its own tilt angle relative to the ground and can be achieved in reality by using a gyroscope or an accelerometer. Whenever the wheelchair has a tilt greater than a configurable number of degrees safety breaks activate if no command is given by the user. This level is begins at an incline before leveling off and descending to the finish line. The user would have focus on moving forward without being distracted by the fact that they could potentially fall off or begin to roll backwards.

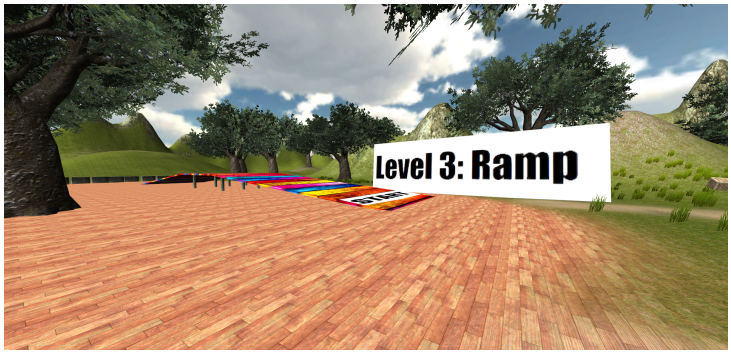


Figure 5.3: Level 3, ramp

5.1.4 Level 4

This level requires more caution when navigating. Cones are laid out alongside the dirt path which the user must steer through in order to reach the finish line. However, these cones have triggers attached to them so that if the user touches one a set of explosions are activated that will propel the wheelchair and surrounding crates into the air. The game is then reloaded and high-scores are lost. The idea behind this is to see whether the user can perform under pressure. When the finish line is reached the user will be transported to the starting line automatically.

Figure 5.4: Level 4, labyrinth

5.1.5 Level 5

This level is more or less a sandbox in which the user can move freely without caution or pressure. The level is littered with moving pedestrians coded to follow random paths. The idea is that the user can practice using different commands in an unrestricted environment to improve their skills with no ramifications resulting from mistakes. In addition, this level was also a testing ground for implementation of the A algorithm used in the wheelchair autopilot.

Figure 5.5: Level 5, street

5.2 Autopilot

The autopilot system works sufficiently within the environment by allowing the user to move around without constantly focusing on a single thought. The autopilot allows the user to move in order from level 1 to level 5. If the user tries to go in reverse from level 3 to level 1 some problems regarding the timer may occur. There is only one allowed moving direction between level 1 and level 3 due to invisible walls as a result of limitations in the timer system.

When the "pull" command is initiated the menu selections light up notifying the user of their selectability. The commands "left" and "right" will now toggle the autopilot GUI menu's selectable selections while the "push" command initiates them. When the wheelchair reaches it's target it will stop and roll to a halt. If the user activates the "pull" command again the wheelchair will revert back to manual mode. Figure 5.2 demonstrate the GUI menu with its five different destinations.

Figure 5.6: The deactivated autopilot GUI

5.3 Steering and Control

After training in the Emotiv training program is complete and the user is readily calibrated, the virtual wheelchair can be controlled. Thinking "push" will command the wheelchair to move forward whereas "rotate left" and "rotate right" are used for turning. In order to stop moving the user just needs to return to a neutral state of mind.

The chair has two safety features: collision avoidance and rolling prevention. If the user is on a collision course with any object tagged as "collidable" the safety brakes will engage whenever the player gets too close. Most objects within the environment are "collidable" except for terrain and moving objects. Brakes are also engaged to avoid unwanted rolling which can occur when the chair's angle relative to horizontal ground reaches a certain threshold and no control command is present.

5.4 Neural Network

The neural network was able to discern meditation from actively thinking "push" with a success rate of 0-4%. This neural network is developed by the neural network toolbox available in Matlab. As presented in Figure 5.3 some of the samples are used to train the network, some are for validating the results, and the rest remain for testing the network.

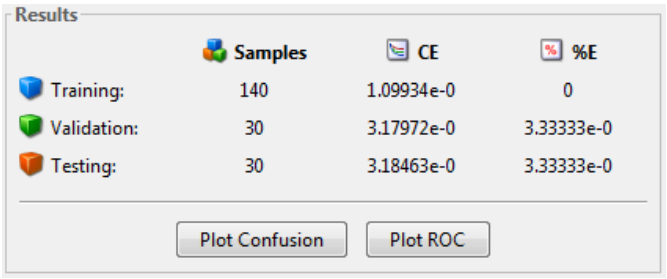


Figure 5.7: Neural networks result

Figure 5.4 shows the best validation performance, here 0.087107 at 18th epoch, and presenting the validation performances bottom-point. Lower value means less probability for false predictions.

Having fewer epochs means the network learns in small repetitions. Performance indicates the final cross-entrophy achieved. Lower value is associated with higher network accuracy.

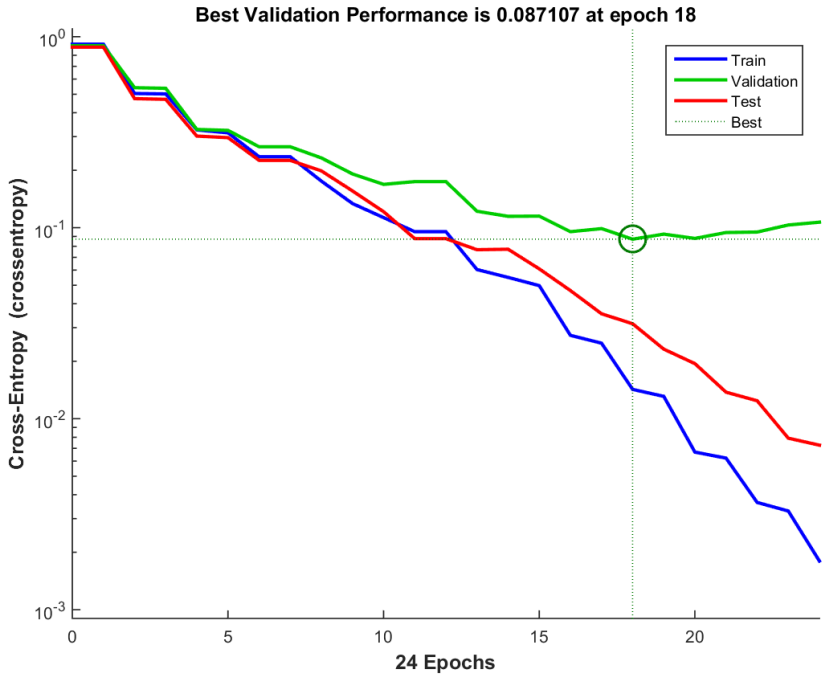


Figure 5.8: The neural networks performance

6 Discussion

This report proves that it is possible to control a virtual wheelchair using the Emotiv Epoc headset, although not without difficulty. A training program, much like the one this project has produced, is considered necessary in order to aid the user in controlling the wheelchair. The Oculus Rift VR headset actually makes the training progress easier for those who don't experience motion sickness with the use of VR. Notably, the group discovered the main problem with BCI is learning to think in such a way that the software is able to detect the user's configured commands. It was quickly realized that switching between different thoughts rapidly was not easy. Some users have shown in different media channels, like YouTube, that they can master these techniques using the same technology utilized in this project. By proper training the group believes it's possible to control a machine through EEG signals. As a result it is believed that those afflicted with certain ailments such as ALS can benefit from this technology.

6.1 Significant Choices

Throughout the project period the group had to make several important decisions regarding the methods that should be used, the software that should be used and other tools that should be used for the various tasks.

6.1.1 Communication

The first important decision involved how to make Emotiv and Unity work together. The previous project made use of a UDP server-client based application written in C# which was tested by the group but it was ultimately deemed inefficient to have a third application running. There is a plugin that is available for Unity which gives the developers the same possibilities as server-client approach with the added benefit of being a more direct way of interconnecting the two applications. The group decided that using a plugin was the route to take, the only thing to consider was that the solution might not work with newer versions of Unity, since the plugin was made by Emotiv in 2011. Therefore future developers should take precautions when implementing the plugin into future Unity versions if the plugin has not been updated.

6.1.2 Control Method

The group discussed various approaches when it came to controlling the wheelchair, the limitation of having a maximum of 4 commands available forced the group to simplify the problem. In a physical wheelchair you can maneuver forwards, backwards in addition to turning. The group had to limit movement to forward, left and right. This means that in order to turn the wheelchair would have to come to a complete stop before turning and that moving in reverse was impossible. Through lengthy discussion the group came up with several approaches ranging from having multiple operation modes to having different screens attached that could be used as an interface. In the end it all became way too convoluted and most of these ideas were discarded at an early stage.

6.1.3 Design

At first the group envisioned a game with selectable levels in a full-screen GUI, similar to how a typical arcade game works. But when it came to training someone to use this in reality, the group decided that an open world feeling would be better and less confined. Doing this would help minimize the gap between game environment and reality as much as possible and at the same time it gave the user the opportunity to move around freely and exploring the environment. After the user has become accustomed to the controls crossing the start line of a level is all it would take to initiate the start of the timer. The group views this freedom as a better way of designing a game instead of spending large amounts of time in menus selecting which level to be in.

6.1.4 Signal Processing

Since the group wanted to do something with the raw data in a very limited time frame, some decisions had to be made on how to efficiently approach the project. Trying to achieve BCI from the ground up was quickly discarded as impossible, but recording different EEG readings and looking at how they differ using techniques such as signal processing and machine learning was more probable.

6.2 Communication

From a more technical point of view different ways of communication between Unity and the Emotiv software have been tested and considered. First, a UDP server-client system was implemented an approach developed by a former project group. The reason this solution was developed was because their work began before the official plugin even existed. Due to this, the previous group was forced to develop their own method of communication between the Emotiv API and Unity. Implementing this solution was a quick way of getting started with the project. By establishing a working connection early on the group learned of the difficulties and possibilities of controlling a character via

BCI. Seeing as this method was a workaround the group considered using the official Unity plugin. Unfortunately by using the workaround the group encountered problems with Unity where the program would crash upon exiting play-mode. Little effort was put into trying to solve this, as a transition to the fully functional, and official plugin was a more natural progression. As an added bonus, it removes the need for running a third program between Unity and Emotiv. This leads to a more seamless and developer friendly experience. This plugin should be maintained and updated as Unity is updated because a typical problem in software is incompatibility when updates and newer versions are released. As of now, it works with Unity version 4.6. Unity 5 was released in March 2015 which sparked interest in migrating the project to the new Unity version. Limited testing shows that the plugin will work in Unity 5 as well, although some limitations apply. The most common of these limitations is that the 32-bit version of Unity 5 must be used, since Unity handles plugins have changed. [Unity Technologies, 2015]

6.3 Control and Safety Concerns

Simplicity is regarded as a key element when introducing new technology to potential users, therefore many different approaches to control have been considered, ranging from swapping between multiple screens and choosing multiple modes of control. Ultimately the group decided on the simplest approach. With the sensors available today in conjunction with some clever algorithms, sufficient AI should be developed allowing the user control over the wheelchair with AI responsible for safety.

6.3.1 Safety

In the virtual world the safety issues regarding the wheelchair have been highly simplified, by placing an invisible sphere around the wheelchair that detects possible collisions and a virtual accelerometer that detects the angle between the wheelchair and the ground, safety is not an issue. The purpose of the angle detection is to simply engage emergency brakes if needed. To achieve the same level of safety in the real world an array of sensors would be needed which, although possible, this would be more expensive than an ordinary wheelchair. In a real scenario the software can potentially detect the wrong command, which can in turn cause real danger. A proper safety system is required if this technology is ever going to be used in the real world. A possible solution would be to have distance sensors, possibly with cameras detecting obstacles and a system that overrides the user when necessary. Having a timing function that makes sure the user would have to focus for a given amount of seconds before the command is actually executed has also been considered. It would be in conjunction with the collision avoidance system and work as a double safety barrier for the user. Another possible approach to safety would be having engaged breaks as a default in which the user could be released by initiating a control command allowing movement. This would also stop the wheelchair more efficiently when going back to neutral. These breaks could also be gradually released as a thought initiates a command, giving the user a certain delay between initiating a command and actually

moving the wheelchair.

6.3.2 Autopilot

Another problem that needs consideration is autopilot and or semi-autopilot. Imagine having to move straight ahead for an extended period of time, the user would need to focus on the same thought for, possibly, several minutes. This could be exhausting for the user and diminish many of the potential benefits. Having the opportunity to choose a given destination would make the user experience simpler. As of now, it is almost impossible to use more than four different commands at a time, the fewer the commands the easier, and sometimes better. There needs to be three commands to control the most basic functions, forward, rotate left and rotate right. This leaves only one function which the group uses to toggle autopilot with pathfinding through waypoints, whereas GPS would be used in a real life application. This type of feature would allow the user to choose a path through a screen attached to the wheelchair. After having chosen a path, the wheelchair would simply find the way by itself. It would require the user to be able to do minor adjustments while all safety systems are engaged and working. It should be possible to stop, turn and do necessary adjustments to avoid any safety hazards. To sum it up, a lot of desired functionality is limited by only having four different commands, if a proper autopilot system were developed in conjunction with AI, it could be possible to explore a great number of desirable features without compromising simplicity. A thing to consider in the future would be how to use this technology in conjunction with already available technologies like EyeGaze, which allows the user to control their wheelchair with eye movement. For instance, controlling the settings and different control modes with eye movement would be a substantial step forward and this would allow for more sophisticated and possibly less strenuous control. A feature that has not been implemented but that the group regards as helpful is the ability to lock a command in order to avoid having to concentrate for extended periods of time when moving in a straight line. Locking a command could be a selection in the autopilot menu and would benefit when having to travel long distances. For example if the forward command could be represented as a selectable arrow then selecting it would initiate the command which would continue running until the user deactivated it. To stop the locked command the user would simply toggle out of the autopilot by thinking "pull".

6.4 Ideas Behind the Environment

Considering the environment that is built by the group, each level has its own purpose. For instance level 1 is designed to makes the user train by using the forward function of the wheelchair only. A timer function is added to give the user feedback on how well the user is doing. After doing this several times the user should be able to improve their score and hence the ability to focus to one thing for an extended period of time. Level 2 is designed to make the user switch between commands, rotate left and rotate right, this has proven itself to be one of the most difficult tasks. By spending a lot of time in

this environment it seems very reasonable, after sufficient training, that it will become easier to maneuver and get through the track without bumping into any of the walls or getting stuck. The next level is all about managing slopes. In level 3 moving on the slope and making sure the user is not rolling backwards on a slope are key elements in improving safety. The safety system should be handling those issues but the group considers it valuable to train on these skills nonetheless. Level 4 is all about precision, making a mistake here is critical. With an added element of stress the user is able to experience a simulation of a stressful situation and learn how to manage. Being able to control the wheelchair in a noise environment where errors could possibly be dangerous is important for the safety of the user. The fifth and last level mixes everything up, the user should be able to maneuver nicely, avoiding obstacles and cross the street in a reasonable amount of time. This level is also filled with distractions that the user should practise avoiding. Distraction is a topic that should be considered, since the wheelchair is a thought controlled device, being distracted could possibly mean loosing control. This fact puts safety at the top of the list if real world application is ever going to be considered.

6.5 Experiments

Emotiv is highly secretive when it comes to how they are detecting and separating different commands on a signal processing level. In light of this fact the group attempted to reverse engineer their technology. Using Simulink it is possible to access the raw unfiltered data coming from the headset which allows the use of algorithms and filtering methods on the signal which could possibly tap into the secret behind their work. At the very least it might be possible to get a sense of which direction is needed to go in order to classify signals into separate mental commands. It seems like it is very difficult to discern a neutral state of mind from a focused mind merely by viewing which areas of the brain are active, and which frequencies are present at any given amplitude. As seen in the result the group had three readings of a neutral mindset and three readings of a mind that was visualizing pushing. No consistent difference was observed. In one of the readings the group could see a high activation in the beta readings, but since it was inconsistent with the other readings, there is no way of concluding that there was a causal effect. It seems like a more heavy duty technical approach is needed, an approach that makes use of sophisticated pattern recognition. A toolbox is available in Matlab that handles many of these things.

Through sampling 100 data sets of "push" and 100 data sets of "meditation", the group acquired 200 data sets to feed into a neural network. The network should classify the data according to the intensity of the beta band. At first, specific frequencies where fed into the system, which produced a result slightly better than random guessing. This could be due to the fact that more data was needed or that simply looking at one specific frequency is prone to missing the target. If you are just looking at 20 Hz, when the information you are looking for could be anywhere from 13 - 30 Hz, it seems obvious that a lot of information is missing. Success was achieved when mean values where used, this allows bands of frequencies to be fed into the system. When each band, alpha, theta, delta and

three bands of beta were used, the classification error dropped to 0-4%. Another way of collecting information into groups like this would be to integrate the data from one frequency to another. The group expected similar results from this technique, but it has not been tested. Integrals are a more accurate way of quantifying the relative size of each data section, but mean values are viewed as satisfactory in this case. However, it does prove that neural networks can play a role in classifying mental states. This reveals that the activity sought after is divided into the whole spectrum of frequencies, and thus dividing all the frequencies into bands and quantifying their relative intensity lets the neural network classify them with successful results. It becomes apparent that a neural network indeed is able to classify different types of EEG signals.

It should be noted that the EEG signals used where expected to be as different as possible, meditation with eyes closed and focusing with visualising a complex image are two completely different mental functions. When it comes to real time BCI this group suspects, from using the technology, that a pattern is stored during calibration, and clever algorithms calculate how similar the user's current brainwaves are to the sample acquired from calibration. This difference is quantified, probably through some algorithms utilizing complicated mathematics that the group was unable to investigate due to time constraints. This happens in real time, so a host of different techniques is probably used. State vector machines and LDA tecnhiques are most likely a part of the process as well. The calculation of this difference in real time lets the software calculate how well the user is focusing, giving the user's thoughts a power level instead of just being an on and off switch. The group was able to classify sampled signals through a neural network, but with the nature of the signals the group can only say that a neural network is able to classify completely different EEG readings. Using a neural network to discern between signals produced making different imagery in the mind such as "left" and "right" which most likely are similar to "push" and "meditation", but have not been tested and can not be commented upon. We speculate that a neural netowrk could very well be part of Emotivs technology, but Emotiv does not want to unveil their classified information.

6.6 Possible Future Applications

The inspiration behind this application stems from the condition known as ALS (amyotrophic lateral sclerosis), also known as Lou Gehrig's disease. This condition gradually paralyses the sufferer, making it difficult to near impossible to control an electric wheelchair. Being completely immobilized removes an important sense of freedom, being able to move around at will is an essential part of feeling human. When this ability gradually dissolves many loose hope and end up in despair. In the light of the technology available today the group believes it is possible to safely and easily control an electric wheelchair, using a user's thoughts in a better way than what is now possible [Megalingam et al., 2013]. This could greatly increase the quality of life for those who suffer by restoring a certain sense of freedom in their lives. This is the main future application foreseen by this project. If a wider scope is applied, any machine that is controlled by a human can possibly be controlled through EEG, since this technology is very new and therefore not yet as refined as it can be, today's applications is not sufficiently

satisfactory.

Hopefully, in the coming years, better algorithms that detect brainwave patterns will be developed and therefore make it easier to use. For now, stringent training is required to make the technology work in a satisfying manner.

6.7 Learning Outcomes

The fulfillment of this project has brought with it a host of useful experiences. Game development which was up until this point a subject untouched, has now become something that the group has become familiar with. Also the application of AI in such a context has become apparent. But most of all, learning to work together and getting to know each others strengths and weaknesses. The group views this as the most useful experience regarding future work, especially when a person work on a skill they are naturally good at, the results can be potentially astounding. Naturally the group learned some basic neuroscience to understand how EEG works, this also relates to how the brain works. This knowledge is future oriented and has the potential to open the mind to ideas not yet thought of.

The group has also been working with neural networks related to pattern recognition. The concept of simplifying difficult concepts through abstraction, turning the level of complexity to a conceivable one and at the same time obtaining the desired functionality, gives a good insight into the essence of engineering. Understanding how these pattern recognition techniques relate to each other in different fields is viewed by the group as valuable insight. Another interesting insight is the fact that any specific problem can be solved in a number of different ways, it's not just about solving a problem, it's about finding the best solution amongst many.

Last but not least, maybe even the most essential part of any project, is planning. Through planning and visualizing the path to success achievements can be made. In addition, properly managing the group and it's tasks carefully is essential to any successful project. This is definitely a valuable lesson that will be carried with us in future work.

7 Conclusions

1. Successful communication between the EEG headset and Unity was established through a plugin, this works the way the group wanted it to. Other methods were tested, but this solution was the most satisfying. The only drawback seems to be that the plugin has to be maintained and updated to ensure compatibility with future versions of Unity, the group can not guarantee that this will be done. At this point, the plugin allows access to all the features provided by Emotiv and thus allowing the Unity game to be controlled through the mind.
2. The literature study gave the group a good foundation in regards of understanding the how and why of EEG, it laid the groundwork for being able to handle the raw signals. It also revealed that similar things have been done before, many have used the Emotiv SDK to achieve BCI in their own applications. Also, many papers exist on BCI, explaining how it works and how it's done.
3. Looking at the environment, the group can, with confidence, conclude that it's a decent place for a user to explore and develop their abilities. The Oculus Rift might help by making the experience feel more real, and by doing so, potentially shorten the training process. Although this seems highly dependant on the individual, some experience motion sickness after prolonged use of the VR headset (Oculus Rift). It should be said the even though the group is satisfied with the result regarding the game, it should be polished. When it comes to performance, bugs and overall finish there is still room for improvement.
4. This project has been uploaded in its entirety on GitHub, the source code can be found along with the Unity game. This allows others to pick up on this work and do whatever they want with it. All the code is documented, and by following this report, it should be relatively easy to perform a continuation of this work.
5. Using Matlab and Simulink with an EEG toolbox, the group was able to record raw EEG signals consisting of 100 samples of "push" commands, and 100 samples of "meditation" commands. Through FFT, filtering, scatter plots and neural networks, the group was able to distinguish these signals. Although, this is not real time processing, only when noisy signals were removed manually, did the neural network have any chance of separating the two. Focusing on a mentally demanding task produces beta waves at a higher magnitude than relaxing or meditating. That means that by only looking at polar opposites did the group have any chance of distinguishing the two different mental states.

All in all the group reached its aim and successfully accomplished the various tasks. The supervisors and the group are happy with the results, with this the group can conclude that a lot has been learned and important experiences have been made.

References

[LaTeX, 2015] LaTeX (2015). LaTeX - A document preparation system. http://www.latex-project.org/.

[Biomedresearches, 2006] Biomedresearches (2006). EEG History. http://www.biomedresearches.com/root/pages/researches/epilepsy/eeg_resources.html.

[Bjørlykke, oint] Bjørlykke, H. (PowerPoint). Artificial Neural Networks. Power point presentation.

[Department of Clinical Neurophysiology, 2015] Department of Clinical Neurophysiology, EA 2683, C. L. F. (2015). The A* Algorithm. http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.

[Emotiv,] Emotiv. Emotiv Software Development Kit(user manual).

[Emotiv, 2014a] Emotiv (2014a). Emotiv Comparison.

[Emotiv, 2014b] Emotiv (2014b). Simulink EEG Importer. https://www.emotiv.com/store/product_112.html.

[F.Kurose and W.Ross, 2013] F.Kurose, J. and W.Ross, K. (2013). *Computer Networking*. Pearson, Edinburgh Gate.

[Games, 2006] Games, R. B. (2006). Intracerebral study of gamma oscillations in the human sensorimotor cortex. http://www.ncbi.nlm.nih.gov/pubmed/17071239.

[Granberg,] Granberg, A. Astar pathfinding website. http://arongranberg.com/astar/.

[Heaton Research, 2008] Heaton Research (2008). The Number of Neurons in the Hidden Layers. http://www.heatonresearch.com/node/707.

[Hu, 2013] Hu, F. (2013). EEG Circuit Design ppt.

[Jacob Thomas Redmond, 2013] Jacob Thomas Redmond (2013). Emotiv Forum. http://emotiv.com/forum/messages/forum4/topic3373/message16873/#message16873.

[Luis Fernando et al., 2012] Luis Fernando et al. (2012). Brain Computer Interfaces, a Review .

[Lécuyer et al., 2005] Lécuyer et al. (2005). Brain-Computer Interfaces, Virtual Reality, and Videogames.

[MathWorks, 2015] MathWorks (1994-2015). Neural Network Toolbox. http://se.mathworks.com/products/neural-network/.

[Megalingam et al., 2013] Megalingam, R., Thulasi, A., and Krishna, R. (2013). Thought Controlled Wheelchair Using EEG Acquisition Device.

[Microsoft, 2015] Microsoft (2015). Visual studio homepage. https://www.visualstudio.com/.

[Nagel, 1995] Nagel, H. (1995). The Biomedical Engeneering Handbook.

[Negnevitsky, 2005] Negnevitsky, M. (2005). *Artificial Intelligence*. Pearson Education Limited, Edinburgh Gate, England.

[Repovš, 2010] Repovš, G. (2010). Dealing with Noise in EEG Recording and Data Analysis. http://ims.mf.uni-lj.si/archive/15(1)/21.pdf.

[ShareLaTeX,] ShareLaTeX. ShareLaTeX. https://www.sharelatex.com/university?ref=footer.

[Slaven Cvijetic, 2013] Slaven Cvijetic (2013). What are Gamma Brain Waves? How to produce more Gamma Waves with Meditation.

[Teplan, 2002] Teplan, M. (2002). Fundamentals of EEG measurement.

[The MathWorks Inc, 2015a] The MathWorks Inc (1994-2015a). MATLAB The Language of Technical Computing. http://se.mathworks.com/products/matlab/.

[The MathWorks Inc, 2015b] The MathWorks Inc (2015b). Simulink. http://se.mathworks.com/products/simulink/.

[Trans Cranial Technologies Ltd., 2012] Trans Cranial Technologies Ltd. (2012). 10/20 System Positioning Manual.

[Unity Technologies, 2015] Unity Technologies (2015). Unity 5 Upgrade Guide. http://docs.unity3d.com/Manual/UpgradeGuide5-Plugins.html.

[Vaque, 1999] Vaque, T. (1999). The history of EEG Hans Berger: Psychophysiologist. A Historical Vignette. http://www.brodmannarea.info/readings/berger_jn3-2.pdf.

[Wikipedia, 2006] Wikipedia (2006). Wave patterns. http://en.wikipedia.org/wiki/Electroencephalography.

[Xcessity Software Solutions,] Xcessity Software Solutions. EPOC Simulink EEG Importer. http://xcessity.at/downloads/EpocSimulinkImporter_update1.zip.

A GitHub

The project in its entirety can be found on GitHub, the whole game environment in addition to the final Matlab project regarding raw EEG data. Two of the repositories contain Unity projects, one with Oculus Rift support, and one without. In the last repository all documents are gathered, this makes it easy to find key information. For simplicity each repository has an accompanying readme-file that will help you with setup.

This link gets you to the repositories: github.com/RolfHjdal/

A.1 Repositories

Bachelor-EEG-Unity

Bachelor-EEG-Unity-Oculus

Bachelor-EEG-Matlab

Bachelor-EEG-Documents