André Filipe Silva Ferreira

ROBOT CONTROL BASED ON BIOSIGNALS

Dissertação de mestrado em Engenharia Electrotécnica e Computadores, orientada pelo Senhor Professor Rui Pedro Duarte Cortesão e apresentada ao Departamento de Engenharia Electrotécnica e Computadores da Universidade de Coimbra

Setembro 2014

· U C ·

UNIVERSIDADE DE COIMBRA

UNIVERSITY OF COIMBRA
FACULTY OF SCIENCES AND TECHNOLOGY
COMPUTERS AND ELECTRICAL ENGINEERING DEPARTMENT

· U C ·

Robot Control Based on Biosignals

Master Thesis in Computer and Electrical Engineering

André Filipe Silva Ferreira

September 2014

Supervisor:
Rui Pedro Duarte Cortesão

Jury:
Helder de Jesus Araújo
Paulo Jorge Carvalho Menezes

Acknowledgements

First and foremost i would like to thank my thesis advisor, Professor Rui Cortesão, for the opportunity to do a dissertation in the amazingly interesting area that is the world of medical robotics area, and for all the availability, motivation, help and insights provided.

A special thanks to PLUX® whose kit, bitalino Plugged, was used in this work and without it, nothing would be possible.

To all of my friends, with a special thanks to André Pinto, who designed the wonderful art in the cover, José Pedro and Mariana Francisco for all the time they spent testing, helping and criticising my work.

Finally, I am grateful to my family, who have always taken interest in my academic progress, and supported me along the way.

Resumo

Este trabalho tem como objectivo o desenvolvimento de uma plataforma modular onde um qualquer numero de gestos distintos, obtidos através de Surface Electromyography (sEMG), possam ser treinados e testados com a maior precisão possível, aplicando esses gestos ao controlo de um robô em tempo real. Um conjunto de seis movimentos foram treinados incluindo extensão do pulso, flexão do pulso, supinação do pulso, pronação do pulso, abdução e adução dos dedos da mão. Foi aplicada o algoritmo Principal Component Analysis (PCA) na matrix de características, obtida numa sessão prévia de treino, e modelos de classificação foram criados com recurso a Support Vector Machines (SVMs) e a Linear Discriminant Analysis (LDA). Quatro pessoas foram submetidas a duas sessões de teste, usando em cada uma delas um classificador diferente (SVM e LDA). Os testes consistiam numa variação de uma Fitts' Target Acquisition Task (FTAT) que involve mover um cursor num espaço pseudo-3D. Um robô 3D foi desenvolvido para testar o desempenho dos métodos num ambiente simulado.

Keywords: EMG, SVM, PCA, LDA, FTAT, Reconhecimento de Padrões .

Abstract

This work has as its goal the development of a modular platform in Matlab where any number of distinct sEMG gestures can be trained and tested with the maximum accuracy possible, and apply those gestures to control a real robot in real time. A set of six movements was trained including wrist extension and flexion, wrist supination and pronation, finger abduction and finger adduction. PCA was applied to the feature matrix, obtained previously, and patter recognition models were created using SVMs and LDA. Four subjects were submitted to two testing sessions each in which a different classifier was used (SVM and LDA) to train a model for the training data. The tests consisted on a variation of FTAT which involves moving a cursor in a pseudo-3D space. A 3D controlled robot was also built to test the performance of the system in a simulated environment.

Keywords: EMG, SVM, PCA, LDA, FTAT, Pattern Recognition .

Contents

1 Introduction **1**

2 From Thought to Movement **2**
- 2.1 The Human Motor System . . . 2
- 2.2 The Neuron as an I/O device . . . 4
- 2.3 The EMG Signal . . . 6

3 Pattern Recognition Myoelectric System **8**
- 3.1 Raw Data Processing . . . 8
  - 3.1.1 Data Collection . . . 8
  - 3.1.2 Data Pre-Processing . . . 9
  - 3.1.3 Data Segmentation . . . 10
  - 3.1.4 Muscle Contraction Detection . . . 13
- 3.2 Feature Extraction . . . 14
  - 3.2.1 Zero Crossings . . . 14
  - 3.2.2 Waveform Length . . . 14
  - 3.2.3 Slope Sign Changes . . . 14
  - 3.2.4 Skewness . . . 15
  - 3.2.5 Root Mean Square . . . 15
  - 3.2.6 Mean Absolute Value . . . 15
  - 3.2.7 Integral Absolute Value . . . 15
  - 3.2.8 Hjorth Parameters . . . 16
  - 3.2.9 AR Coefficients . . . 16
- 3.3 Dimensionality Reduction . . . 17
- 3.4 Classification . . . 19
  - 3.4.1 Linear Discriminant Analysis . . . 19
  - 3.4.2 Support Vector Machines . . . 21

4 Real Time Testing **25**
- 4.1 Data Extraction Considerations . . . 25
- 4.2 Database . . . 26
- 4.3 Six-Movement Classification . . . 26
  - 4.3.1 GUIs . . . 29

4.3.2 Testing Method 29
4.3.3 Testing procedure 29
4.3.4 Training Results 31
4.3.5 Classification and testing results 34
4.3.6 Robot Simulator 34

5 Conclusion and Future Work **39**
5.1 Conclusion 39
5.2 Future Work 39

Appendices **44**

A GUIs **45**

B Detailed Subject Test Results **51**

List of Figures

2.1 General organization of the Motor System in Humans. 3
2.2 Principal cortical domains of the motor system 3
2.3 The motor homunculus. 4
2.4 Connection between the brain and the lower motoneurons. 4
2.5 A typical action signal 5
2.6 A typical neuron anatomy 5
2.7 An action potential traveling along the bipolar electrodes. 6
2.8 A raw surface Electromyography (EMG) reading. 7

3.1 Block diagram of the EMG pattern recognition training system. 8
3.2 Amplitude and phase response of the highpass filter used. 9
3.3 Amplitude and phase response of the notch filter used. 10
3.4 Classification error compared to segment length in the steady and transient states. . 12
3.5 The two major windowing techniques. 12
3.6 Example read and detection on real time data 13
3.7 Model order and Root Mean Square (RMS) difference comparison. 17
3.8 Difference between model order and the time cost. 18
3.9 Example of two class densities. 20
3.10 Linear separating hyperplanes for the separable case. 22
3.11 Linear separating hyperplanes for the non-separable case. 23

4.1 Posterior and anterior views of the electrode placement on the left arm. 27
4.2 Six movements for classification 27
4.3 The Fitts interface. 30
4.4 Widths and locations of all possible targets. 30
4.5 Grid Seach of the SVMs using 5-fold cross-validation. 32
4.6 Testing accuracies for both classifiers per subject. 33
4.7 Averaged and normalised confusion matrix across all test subjects for SVM. The testing data was used to produce this result. 33
4.8 Averaged and normalised confusion matrix across all test subjects for LDA. The testing data was used to produce this result. 34
4.9 Completion times for both Degree of Freedoms (DOFs) and for both classifiers per subject. 37

4.10 Accuracy rates for both DOFs and for both classifiers per subject. 37
4.11 Overall completion times and target accuracy rates for both methods. 37
4.12 Robot simulator built within the Matlab environment. 38

A.1 The main GUI. 47
A.2 The options Graphical User Interface (GUI). 47
A.3 The tuning GUI for the Fitts test 48
A.4 The data analysis GUI. 48

List of Tables

3.1 Summary of the specifications of the bitalino kit. 9

4.1 Correspondence between electrode placement and the target muscles. 28
4.2 Correspondence between anatomical movements and the action for the robot to perform. 28
4.3 Target count of each testing session. 31
4.4 Performance metrics used. 32
4.5 Performance metrics across all subjects using the SVM classifier. 35
4.6 Performance metrics across all subjects using the LDA classifier. 36

A.1 Item numbers and a short description for the main GUI. 46
A.2 Item numbers and a short description for the options GUI. 49
A.3 Item numbers and a short description for the FTAT test tuning GUI. 49
A.4 Item numbers and a short description for the data analysis GUI. 50

B.1 Performance metrics for subject A using SVM. 52
B.2 Performance metrics for subject B using SVM. 53
B.3 Performance metrics for subject C using SVM. 54
B.4 Performance metrics for subject D using SVM. 55
B.5 Performance metrics for subject A using LDA. 56
B.6 Performance metrics for subject B using LDA. 57
B.7 Performance metrics for subject C using LDA. 58
B.8 Performance metrics for subject D using LDA. 59

Acronyms

AR Autoregressive. 16, 17, 49

CNS Central Nervous System. 2

DAGSVM Directed Acyclic Graph-Support Vector Machine. 24

DC Direct Current. 9

DOF Degree of Freedom. iii, iv, 31, 35, 37

ECG Electrocardiography. 6

EEG Electroencephalography. 16

EMG Electromyography. iii, 6--8, 13, 39, 40

EPSP Excitatory Post-Synaptic Potential. 4

FNS Functional Neuromuscular Stimulation. 1

FTAT Fitts' Target Acquisition Task. v, 29, 34, 39, 49

GUI Graphical User Interface. iv, v, 25, 29, 30, 45--50

IAV Integral Absolute Value. 15, 17

IPSP Inhibitory Post-Synaptic Potential. 4

LDA Linear Discriminant Analysis. v, 19, 20, 29, 34, 36, 39

MAV Mean Absolute Value. 15, 17, 31

MUAP Motor Unit Action Potential. 6

PCA Principal Component Analysis. 18, 19, 29

PMA Pre-Motor Area. 2

PPC Posterior Parietal Cortex. 2

RBF Radial Basis Function. 23

RMS Root Mean Square. iii, 15, 17

sEMG Surface Electromyography. 1

SMA Supplementary Motor Area. 2, 3

SSC Slope Sign Changes. 14, 17

SVM Support Vector Machine. iii, v, 22, 24, 29, 31, 32, 34, 35, 39

WL Waveform Length. 14, 17

XML eXtensible Markup Language. 26

ZC Zero Crossings. 14, 17

Chapter 1

Introduction

The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny ...' - Isaac Asimov

The first documented experiments regarding electromyographic signals were Francesco Redi's works in 1666. Redi discovered a highly specialized muscle of the Electric Eel generated electricity. Despite this early discovery it was not until late 1800s that the first actual recording of this activity was made, by a french scientist, Ètienne-Jules Marey, who coined the term electromyography. In 1944 Herbert Spencer Gasser and Joseph Erlanger received the nobel prize for their work with action potentials in nerve fibers.
Since its first clinical use in the 1960s the sEMG field has seen numerous improvements and with them new and wide applications. Among them we have: multifunction prosthesis, exoskeletons, wheelchairs, gait generation, grasping control, virtual keyboards, gesture-based interfaces, virtual worlds, robot control, diagnoses and clinical applications, such as Functional Neuromuscular Stimulation (FNS), and detection of preterm births based on uterine myoelectric signals. However, despite many advances and capabilities, myoelectric control has a significant distance from professional and commercial applications. It needs complementary interfaces to deal with all requirements for fine control and suffers a lack of sensory feedback. In a attempt to close this gap, this thesis compares two methods of classification applied to a test that resembles real world conditions and control.

Chapter 2

From Thought to Movement

2.1 The Human Motor System

Albeit somewhat artificial, an anatomical division of regions involved in the production of movement can be instructive in understanding the physiology of the human motor system (Figure 2.1). The brain stem and spinal cord can be viewed as the regions of the Central Nervous System (CNS) that contain the interneuronal-motoneuronal [1] networks responsible for the production of reflexive and rhythmic motor behaviours.

The primary motor cortex, or M1, is one of the principal brain areas involved in motor function. M1 is located in the frontal lobe of the brain, along a bump called the precentral gyrus (Figure 2.2). The role of the primary motor cortex is to generate neural impulses that control the execution of movement. Signals from M1 cross the body's midline (midsagittal plane) to activate skeletal muscles [2] on the opposite side of the body, meaning that the left hemisphere of the brain controls the right side of the body, and the right hemisphere controls the left side of the body. The amount of brain matter devoted to any particular body part represents the amount of control that the primary motor cortex has over that body part. For example, a lot of cortical space is required to control the complex movements of the hand and fingers, and these body parts have larger representations in M1 than the trunk or legs. This disproportionate map of the body in the motor cortex is called the motor homunculus (Figure 2.3).

The other regions of the cortex involved in motor function are called the secondary motor cortices. These regions include the Posterior Parietal Cortex (PPC), the Pre-Motor Area (PMA), and the Supplementary Motor Area (SMA). The PPC receives many and various inputs including visual, auditory, somesthetic, limbic and outputs motor signals. The posterior parietal areas send this information on to the premotor cortex and the supplementary motor area. The PMA lies just in front of (anterior to) the primary motor cortex. Although not fully understood it is believed to be involved in the sensory guidance of movement, and the control of the more proximal muscles

[1]A interneuron is a neuron which forms a a connection between other neurons, while a motoneuron is a neuron that originates in the motor region of the cerebral cortex or the brain stem, whose cell body is located in the spinal cord and whose fibber (axon) projects outside the spinal cord to directly or indirectly control muscles.

[2]A skeletal muscle is a form of striated muscle tissue that is voluntarily controlled. It is one of three major muscle types, the others being cardiac and smooth muscle. As their name suggests, most skeletal muscles are attached to bones by bundles of collagen fibbers known as tendons.

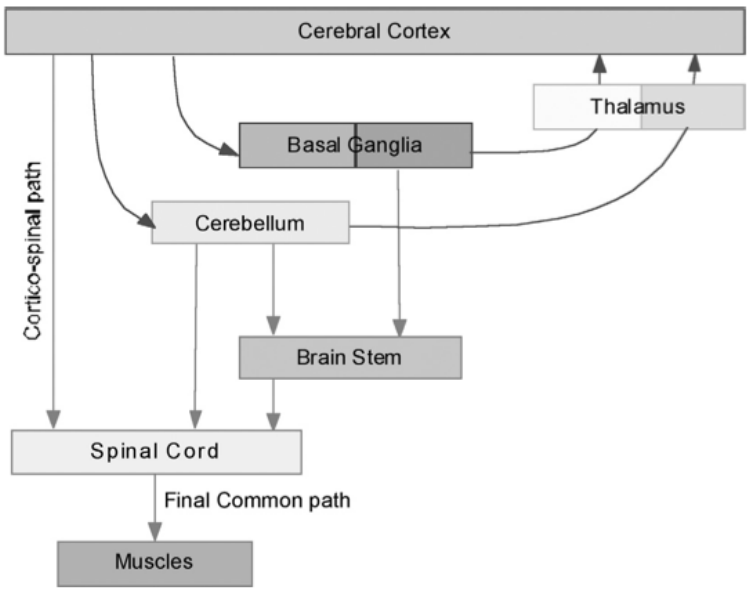


Figure 2.1: General organization of the Motor System in Humans.

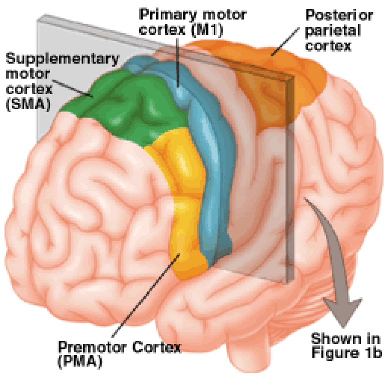

Primary motor cortex (M1)
Posterior parietal cortex
Supplementary motor cortex (SMA)
Premotor Cortex (PMA)
Shown in Figure 1b

Figure 2.2: Principal cortical domains of the motor system.

and trunk muscles of the body. The supplementary motor area lies above, or medial to, the premotor area, also in front of the primary motor cortex. Neurons in the SMA project directly to the spinal cord and may play a role in the direct control of movement. Possible functions attributed to the SMA include the postural stabilisation of the body, the coordination of both sides of the body such as during bimanual action, the control of movements that are internally generated rather than triggered by sensory events, and the control of sequences of movements.

Neurons in M1, SMA and premotor cortex give rise to the fibers of the corticospinal tract which is the only direct pathway from the cortex to the spine and is composed of over a million fibers. These fibers descend through the brainstem where the majority of them cross over to the opposite side of the body. The corticospinal tract is the main pathway for control of voluntary movement in humans but there are other motor pathways which originate from subcortical groups of motor neurons. The nerve axons travelling down the tract are referred to as upper motor neurons (refer to Figure 2.4). These axons travel down the tracts in the white matter of the spinal cord (which contain both white and grey matter) until they reach the vertebral level of the muscle that they will innervate.

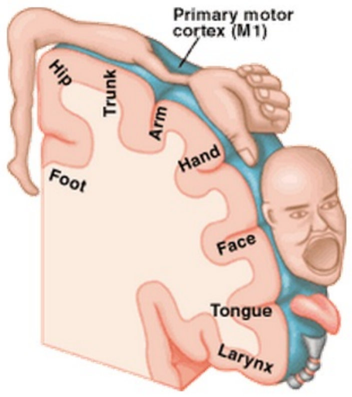


Figure 2.3: The motor homunculus in the primary motor cortex on the right.

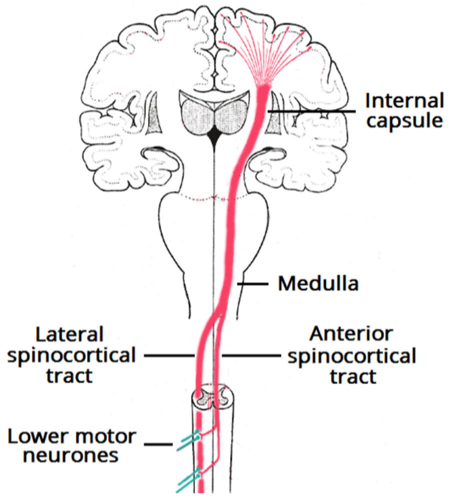


Figure 2.4: Connection between the brain and the lower motoneurons.

At this point, the axons synapse with lower motor neurons. The majority of axons do not directly synapse with lower motor neurons, but instead synapse with an interneuron that then synapses with a lower motor neuron.

Each motor neuron in the spine is part of a functional unit called the motor unit. The motor unit is composed of the motor neuron (or motoneuron), its axon and the muscle fibers it innervates. Smaller motoneuron typically innervate smaller muscle fibers. A motoneuron can innervate any number of muscle fibers, but each fibre is only innervated by one motoneuron which when fired makes all of its muscle fibers contract. The size of the motoneuron and the number of fibers that are innervated contribute to the force of the muscle contraction.

2.2 The Neuron as an I/O device

Just as atoms are the basic building blocks of matter, the neurons are the basic building unit of the nervous system. A typical neuron is composed by an axon, a cell body (soma) and dendrites (a dendritic tree). The dendrites can be considered the inputs of the neuron and the axon its output. The dendrites of a neuron have protrusions, or spines, that capture the neurotransmitters released by the presynaptic neuron at synapses. A synapse is a structure that allows the passing an electrical or chemical signal from a neuron (the presynaptic cell) to another cell (the postsynaptic cell). The generation of a spike in the neuron's membrane (action potential), is determined by the sum of its inputs (the postsynaptic potentials at its dendrites) which can have two types of presynaptic cells making synaptic contact with them: an excitatory cell which tends to changes the local potential of its postsynaptic cell to positive values, Excitatory Post-Synaptic Potentials (EPSPs), or an inhibitory which tends to lower the local postsynaptic potential, Inhibitory Post-Synaptic Potentials (IPSPs). The neuron's membrane, considering all the inputs (the positives and negatives), can gen-

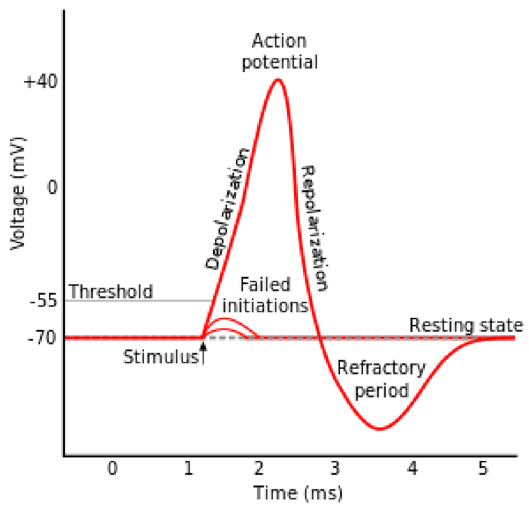


Figure 2.5: A typical action signal.

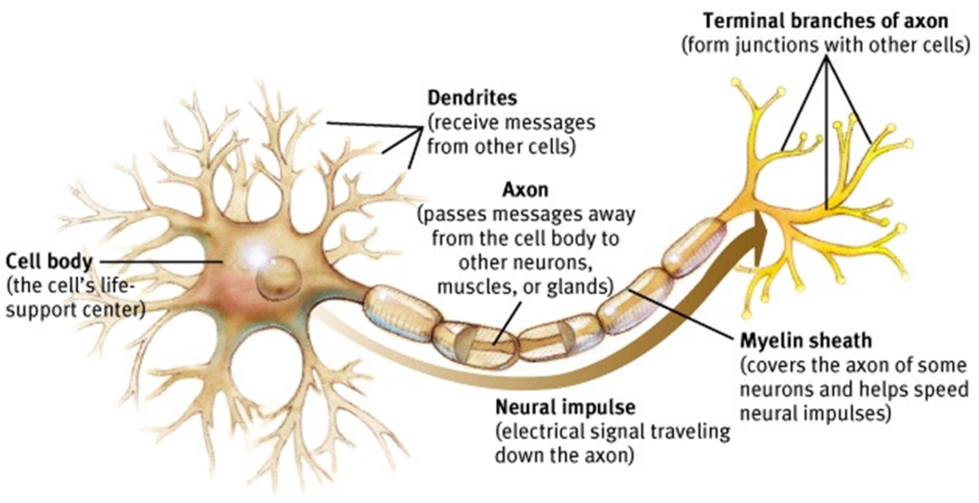

Terminal branches of axon
(form junctions with other cells)
Dendrites
(receive messages from other cells)
Axon
(passes messages away from the cell body to other neurons, muscles, or glands)
Cell body
(the cell's life-support center)
Myelin sheath
(covers the axon of some neurons and helps speed neural impulses)
Neural impulse
(electrical signal traveling down the axon)

Figure 2.6: A typical neuron anatomy.

erate a spike which is transmitted along the neuron's axon. This the action potential is governed by an all or nothing rule which dictates that a stimulus has to surpass a minimum threshold in order to be generated. The membrane's resting potential is situated around -70mV and has a threshold around -55mV. When this threshold is met the membrane's potential rises quickly, depolarizes, until it reaches its peak which is followed by a rapid decrease in the cell's membrane to its resting value (repolarizes) as it can be seen in Figure 2.5. The connection between the motoneurons and the muscle fibers is know as a neuromuscular junction which is, by definition, a synapse. This synapse, although being different from the synapses (chemical synapses) between neurons, has the same function: transmission and reception of chemical messages which are triggered by the arrival of an action potential.

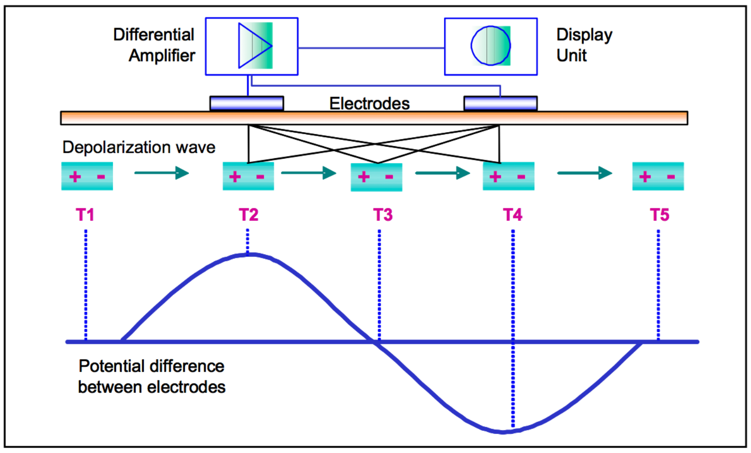


Figure 2.7: An action potential traveling along the bipolar electrodes.

2.3 The EMG Signal

The action potential repolarization and depolarization phases travel along the surface of the muscle fibers as a wave. This wave can be detected by placing bipolar electrodes (and a reference electrode in an area where there are no muscles underneath such as the wrist) longitudinal to the muscle. In the figure 2.7 the action potential reaches the first electrode at time T_1 and the potential difference increases as time increases until it reaches its peak value (depolarization) at T_2, followed by the repolarization at T_3. After T_3 the second electrode starts to receive the depolarisation part of the action potential which makes the electric potential rise to the symmetric peak value of T_2.

As it was previously mentioned a motor unit consists of many muscle fibers which are detected by the electrode pair as a superposed signal of the whole motor unit. This superposed signal consists of the sum of the magnitude of all the innervated fibers within the motor unit firing at different frequencies and magnitudes and it is called a Motor Unit Action Potential (MUAP). An unprocessed MUAP detected by a pair of electrodes is called a raw EMG signal. This signal is essentially composed by two states: the rest state, where the muscle is relaxed and the baseline can be seen, and the contraction state when a muscle is contracted. The baseline must be as low as possible in order to accurately distinguish between activity and rest. This rest value is dependant on the quality of the detection equipment as well as the skin preparation done before the readings, and must not be higher than $3-5$ µV, with $1-2$ µV being the target. The frequency of the EMG range between 6 and 500Hz with the most power concentrated between 20 and 150Hz. The EMG signal can be influenced by several factors which can be grouped in the following categories:

1. Skin Characteristics: As the electric conductivity vary with tissue type, thickness, physiological changes and temperature this introduces in the EMG signal a great degree of variability between subjects and even within the same subject and makes it impossible to analyse the EMG signal directly.

2. Cross Talk: The muscles in the neighbourhood of the electrode can influence its reading. Electrocardiography (ECG)[3] spikes can influence the reading as well, especially when the

[3]Electrocardiography is the recording of the electrical activity of the heart.

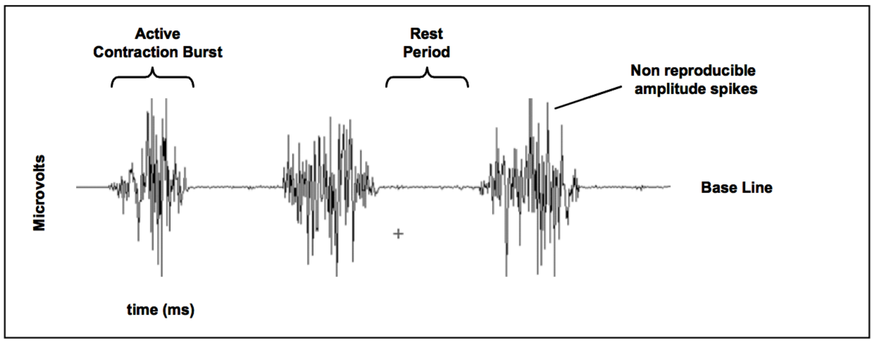


Figure 2.8: A raw surface EMG reading.

EMG is performed in the upper trunk, however they can be easily detected and its effect reduced by the used of post capture algorithms.

3. Changes between the muscle and the electrode site: Any change in distance or pressure during movement between the signal origin and detection will alter the EMG reading.

4. External noise: A noisy electrical environment will degrade the signal to noise ratio and it is typically caused by incorrect grounding of external devices (power hum).

Chapter 3

Pattern Recognition Myoelectric System

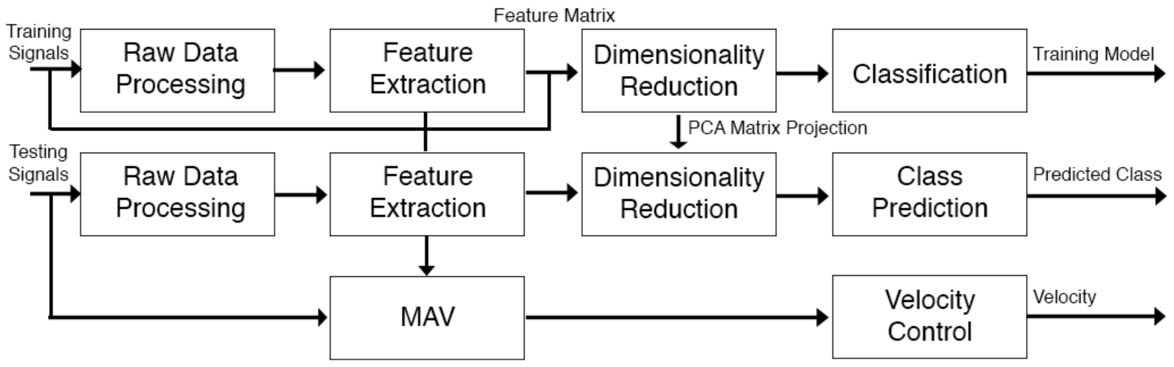


Figure 3.1: Block diagram of the EMG pattern recognition training system.

A typical pattern recognition myoelectric system can be divided into distinct modules which will be discussed in the following sections. A overview can be seen in fig. 3.1.

3.1 Raw Data Processing

Data captured by the EMG recorder needs to be preprocessed before it can be used. In this phase the data are amplified, filtered and sampled which is then fed to the next module.

3.1.1 Data Collection

Data were acquired using bitalino Plugged from PLUX® using 4 EMG sensors connected to 4 analog channels. This sensors possess a bandwidth of $10-400$Hz and a amplifier with gain 1000. After filtering and amplification the micro controller present on the board samples the data into a desired sampling rate between 1, 10, 100 or 1000Hz. As the frequency of EMG signal can range between 10 and 400Hz, a minimum sampling rate of 800Hz is required according to the Nyquist-Shannon sampling theorem. Thus, a sampling rate of 1000Hz was chosen to perform all data analysis. Data are transmitted by a bluetooth interface included in the board to a computer running Matlab to be processed.

V_{cc}	Sensor Range	EMG_g	n	Sampling Frequency
3.3V	$10 - 400$Hz	1000	10	1000Hz

Table 3.1: Summary of the specifications of the bitalino kit.

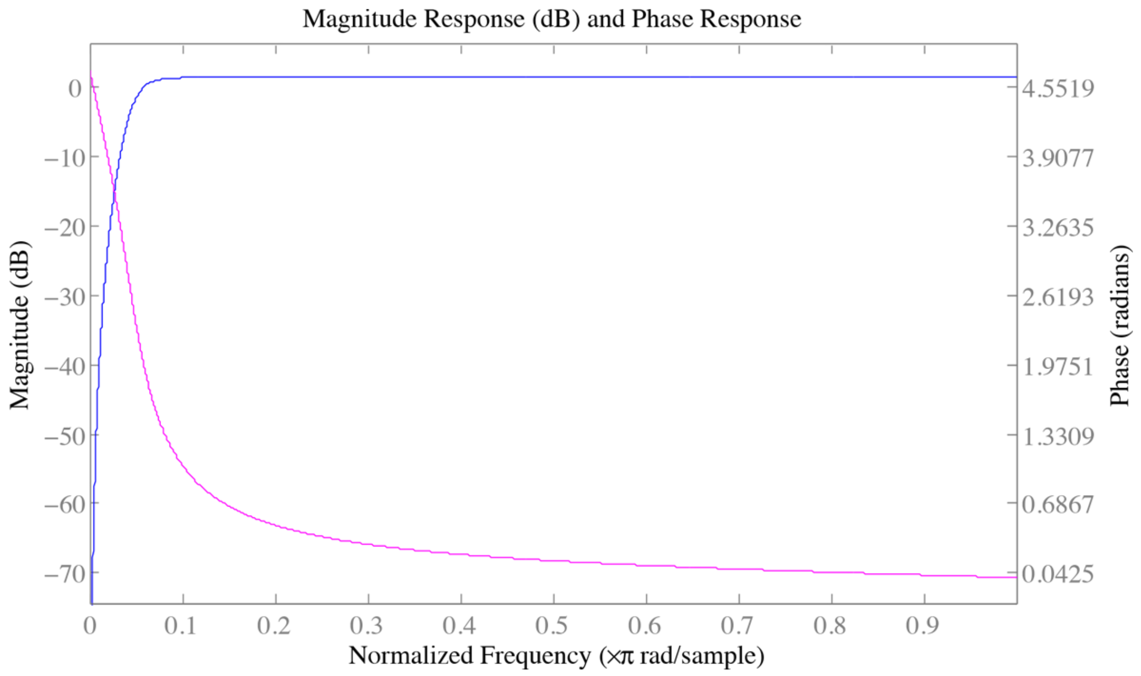


Figure 3.2: Amplitude (in blue) and phase (in magenta) response of the highpass 3^{rd} order butterworth filter used. From the amplitude plot, and using normalised frequency we obtain the highpass filter with cut-off frequencies from 0 to $\approx 0.06 \times 500 \approx 30$Hz.

3.1.2 Data Pre-Processing

The data are acquired by the electrodes, amplified, and sent to a Matlab routine. In this routine the data are converted to actual units (Volts) using the provided equation

$$EMG_v = \frac{V_{cc}\left(\frac{EMG_b}{2^n - 1} - \frac{1}{2}\right)}{EMG_g}, \tag{3.1}$$

where V_{cc} is the operating voltage in volts, EMG_b is the value read by the bitalino, n is the number of bits of the channel (2^n quantization levels) and EMG_g is the amplifier gain. The obtained values are then converted to microvolts, high pass filtered at 25Hz with a notch filter implemented to remove the 50Hz (ranging between 49 and 51Hz) line interference and its Direct Current (DC) component was eliminated by subtracting each channel mean to the signal. In the implementation of the filter the Matlab function *filtfilt* was used, which provides zero-phase digital filtering by processing the input data in both the forward and reverse directions .

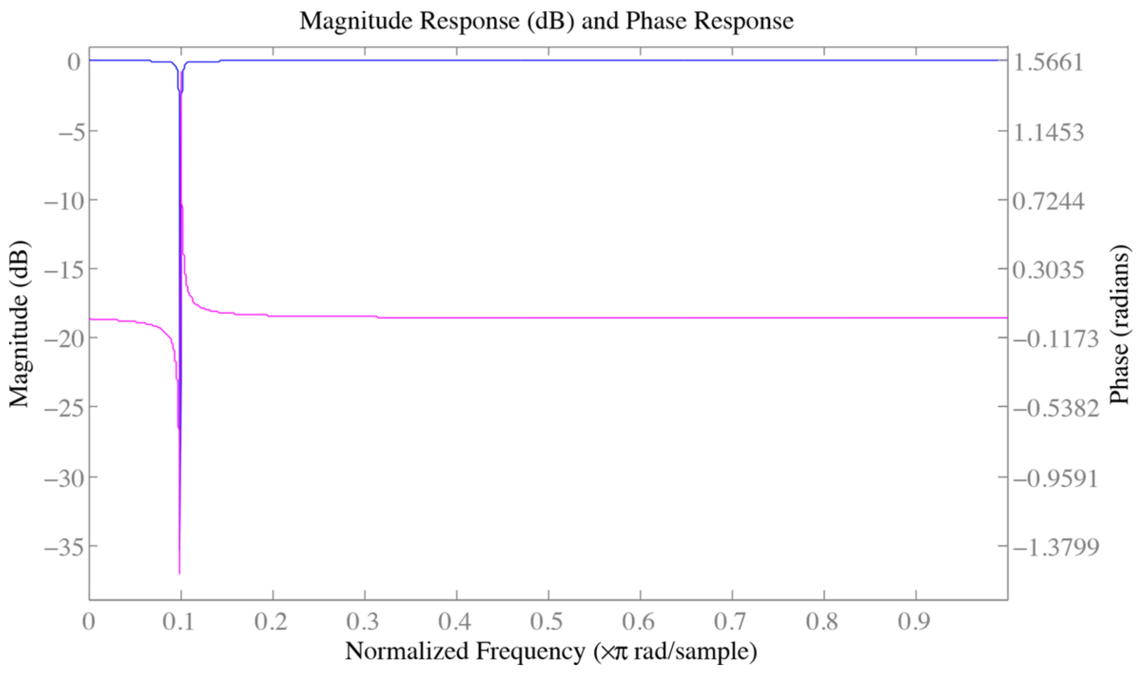


Figure 3.3: Amplitude (in blue) and phase (in magenta) response of the notch filter used. From the figure and using normalised frequency we get $0.1 \times 500 = 50$Hz which was the desired frequency for this filter.

3.1.3 Data Segmentation

Data needs to be time slotted before feature extraction, and due to real-time constraints an adjacent segment plus the required processing time should not exceed 300 ms. Furthermore, Englehart and Hudgins [3] highlighted that by applying continuous segmentation to a steady state signal, the segment length can be as low as 128 ms, or even 32 ms, without a significant decrease in classification accuracy ($< 8\%$).

A myoelectric signal (on muscle contraction) can be divided in two states: (i) a transient state which is comprised between the initial contraction to about 100 ms after onset and (ii) a steady state which takes place after the transient until the muscle contraction ends. Englehart et al. [4] highlighted that steady state data are classified more accurately than transient data, and the classification error increase less with shorter segments lengths with the former than the latter (Fig 3.4).

Another important aspect of data segmentation is the windowing technique. There are two major techniques in data windowing: adjacent and overlapping windowing. In the case of adjacent windowing, consecutive segments of a predefined length are used for feature extraction with a classification result emerging after a certain processing delay. In this case the processor is idle during the remaining time of the segment length (segment length - processing time). On the other hand, in overlapping windowing, the next segment slides over the previous segment by a predefined window step which is less than the total window segment and greater than the time that is required to compute a result. This technique produces more training data and outputs more classification results at the cost of computing time.

Continuous segmentation

Continuous segmentation or continuous classification was first presented by Englehart and Hudgins [3] . In this method, a dense and continuous stream of decisions is produced using multiple overlapping segments which uses the full capacity of the processing system. With this method we can have more classifications in a shorter period of time (fig 3.5b). A final decision based on these classifications is computed by a post-processing method. In this work the majority voting was chosen as such method.

Majority voting consists of taking the most common value of class decisions vector containing a desired number of samples, or in another words, taking the mode of the referred vector. The size of this vector can be defined by the odd number $2m+1$, which reduces the chance of the same number of occurrences for different classes. As such, m is the parameter that defines the acceptable delay for the system. The acceptable delay is the duration between the signal onset (muscle contraction) and the first generated decision by the method. If we define the minimum samples needed for a m-step decision as

$$M_s = W_s + (2m+1)W_t \tag{3.2}$$

where W_s is the window size and W_t is window step or increment, and defining the acceptable delay as T_d then we need at least n windows of EMG data to get a decision:

$$\begin{aligned} T_d &= nW_s && (3.3)\\ T_d &\geq M_s && (3.4)\\ T_d &\geq W_s + (2m+1)W_t && (3.5) \end{aligned}$$

Then for a given acceptable delay T_d we can have, at most, m-steps given by

$$0 < m \leq \frac{T_d - W_s - W_t}{2W_t}, \quad m \in \mathbb{N}^+ \tag{3.6}$$

or, expressing the previous equation in a calculable way

$$m = \left\lfloor \frac{T_d - W_s - W_t}{2W_t} \right\rfloor, \quad m > 0 \tag{3.7}$$

Then, to increase the number of available classications $2m+1$ in a fixed T_d we can only shorten the window size, W_s and W_t which can drastically lower the classification accuracy. However, Englehart and Hudgins [3] has shown that this effect can be counteracted by applying this method, producing faster decision results with very little difference in performance, allowing a lower response time for the system.

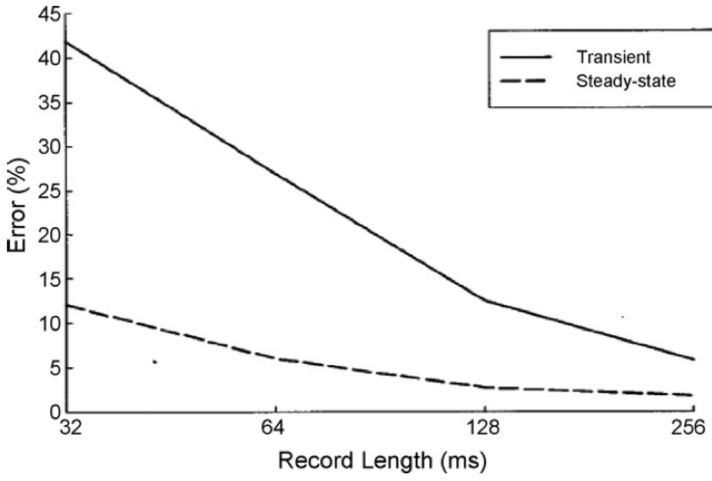
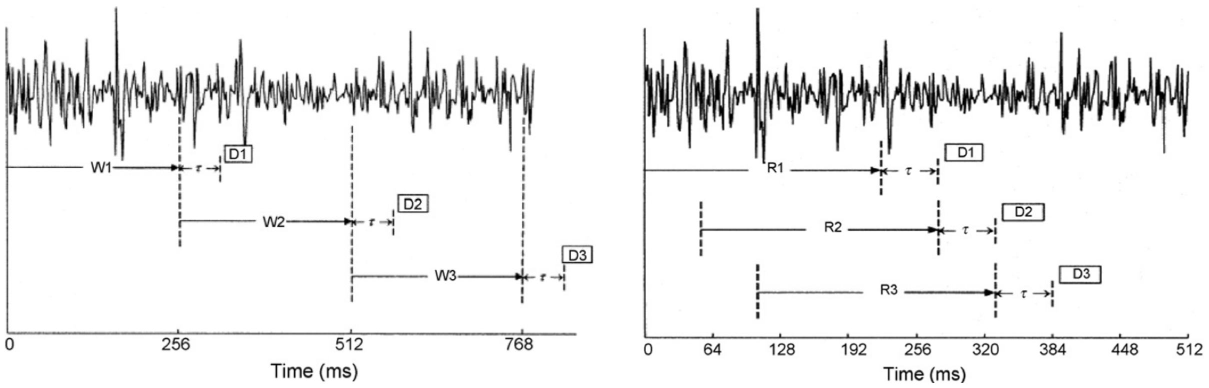


Figure 3.4: Classification error compared to segment length in the steady and transient states [4].



(a) Adjacent windowing (b) Overlapping windowing

Figure 3.5: The two major windowing techniques. [3].

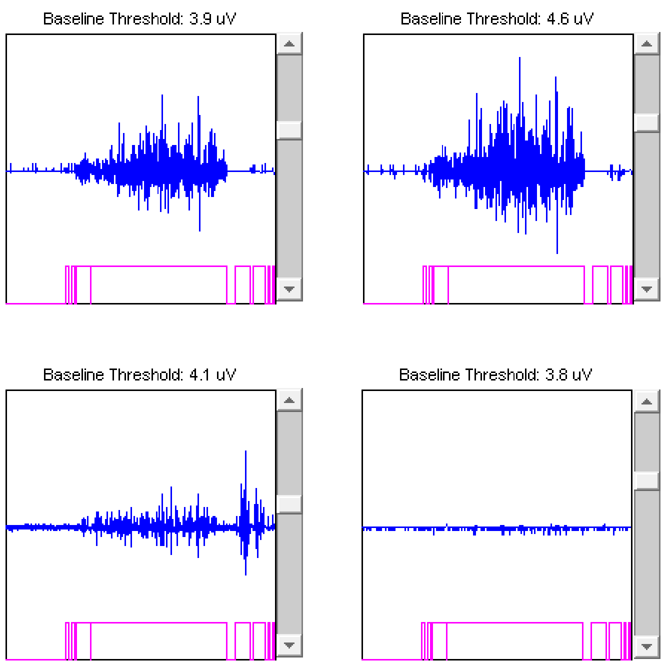

Baseline Threshold: 3.9 uV
Baseline Threshold: 4.6 uV
Baseline Threshold: 4.1 uV
Baseline Threshold: 3.8 uV

Figure 3.6: Example read and detection on real time data.

3.1.4 Muscle Contraction Detection

Estimation of the on-off timing of a muscle is crucial to a real time prediction EMG system. The most simple method to detect this timing is the single threshold method. This technique signals a muscle contraction when a fixed threshold is exceeded by a value the full rectified[1] data segment. However, this method has only one degree of freedom and led to unsatisfactory results as it gives a lot of false positives [15][17].

In order to overcome drawbacks of the single-threshold method mentioned a group of methods based on two thresholds were proposed by Bonato *et al* [16]. The double-threshold method proposed allows to decrease the number of false alarms and improves sensitivity of detection by increasing the number of degrees of freedom. For this method, a traditional single-threshold method is applied to the rectified data, but signal onset is detected only when at least a predefined number (the second threshold) of consecutive samples exceed the first threshold. This proposed approach is then defined by two parameters: the first threshold (from a traditional first-threshold method) and the second threshold (the number of consecutive samples exceeding the first threshold).

In this work the full wave rectification was performed by a moving average on the absolute values of the data. A moving average smooths the data and is calculated by convoluting the matrix composed of the actual (absolute) data plus the last samples with a mask. The mask used was a vector with 16 ones so the detection won't be to fragmented (fig. 3.6).

[1]Full Wave rectification consists of taking all the negative values into positive and smooth the resulting data using a averaging filter. In this work a moving average filter was implemented by using a convolution.

3.2 Feature Extraction

Directly feeding the myoelectric signal to a classifier is not feasible due to the large number of inputs and the randomness associated with the signal. Thus the data segment must be mapped to a lower dimension vector, the feature vector. Features for EMG signals can be divided in three categories: (i) time domain, which due to its computational simplicity it is the most popular in myoelectric classification, (ii) a frequency domain which is mostly used to study muscle fatigue and (iii) time-frequency domain which uses Fourier, Wavelet and Wavelet Packet transforms. As time domain features provide high classification results [5][7][8] with low computational cost they were used in conjunction with a frequency domain feature (AR coefficients). All features were extracted in each data segment.

3.2.1 Zero Crossings

Zero Crossings (ZC) is the number of times that the amplitude value of the EMG signal crossed $y = 0$ and it can be formulated as:

$$\mathbf{ZC} = \sum_{n=1}^{N-1} sgn(x_n \times x_{n+1}) \tag{3.8}$$

$$sgn(x) = \begin{cases} 1, & \text{if } x < 0 \\ 0, & \text{otherwise} \end{cases}$$

3.2.2 Waveform Length

Waveform Length (WL) is the cumulative length of the waveform over the data segment. It is given by:

$$\mathbf{WL} = \sum_{n=1}^{N-1} |x_{n+1} - x_n| \tag{3.9}$$

3.2.3 Slope Sign Changes

Slope Sign Changes (SSC) are similar to zero crossings and such, counts the changes between positive and negative slope. The threshold value prevents little fluctuations in the data segment to be accounted for. The definition is given by

$$\mathbf{SSC} = \sum_{n=2}^{N-1} f[(x_n - x_{n-1})(x_n - x_{n+1})] \tag{3.10}$$

$$f(x) = \begin{cases} 1, & \text{if } x >= \text{threshold} \\ 0, & \text{otherwise} \end{cases}$$

3.2.4 Skewness

Skewness is a measure of the asymmetry of the data around the sample mean. If skewness is negative, the data are spread out more to the left of the mean than to the right. If skewness is positive, the data are spread out more to the right. The skewness of the normal distribution (or any perfectly symmetric distribution) is zero. The skewness of a distribution is defined as

$$s = \frac{E(x-\mu)^3}{\sigma^3} \tag{3.11}$$

where μ is the mean of x, σ is the standard deviation and $E(t)$ represents the expected value of the quantity t. The built in MATLAB function was used to compute the skewness and can be described by the following equations:

$$s = \frac{\sqrt{n(n-1)}}{n-2} s_1 \tag{3.12}$$

$$s_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^3}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2}\right)^3} \tag{3.13}$$

which are the bias corrected formulas.

3.2.5 Root Mean Square

RMS , also known as the quadratic mean, is a statistical measure of the magnitude of a varying quantity. It is given by

$$\textbf{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2} \tag{3.14}$$

3.2.6 Mean Absolute Value

The Mean Absolute Value (MAV) is calculated by taking the average of the absolute value of the data segment, or in different wording, is the average value of a full-wave rectified data segment and thus is given by

$$\textbf{MAV} = \frac{1}{N}\sum_{n=1}^{N} |x_n| \tag{3.15}$$

3.2.7 Integral Absolute Value

Integral Absolute Value (IAV) is the summation of the absolute values of the data segment, and thus it is given by

$$\textbf{IAS} = \sum_{n=1}^{N} |x_n| \tag{3.16}$$

3.2.8 Hjorth Parameters

Hjorth introduced, in 1970, three parameters based on time domain properties [9][10]. They were intended as a clinically useful tool capable of describing quantitatively the characteristics of an electroencephalography[2] trace in terms of amplitude, slope, and slope spread. These parameters are named "Activity", "Mobility", and "Complexity". Activity (eq. 3.17) measures the variance of the amplitude of the signal. In the frequency domain, it can be conceived as the envelope of the power spectrum. Mobility (eq. 3.18) measures the ratio between the standard deviation of the slope and the standard deviation of the amplitude given per time unit; hence, it represents dominant frequency. Complexity (eq. 3.19) is a dimensionless parameter that quantifies any deviation from the sine shape as an increase from unit. It can be interpreted as a measure of the signal's bandwidth. These parameters can be formulated as [11]:

$$\mathbf{A} = \sigma_0^2 \tag{3.17}$$

$$\mathbf{M} = \frac{\sigma_1}{\sigma_0} \tag{3.18}$$

$$\mathbf{C} = \frac{\sigma_2 \sigma_0}{\sigma_1^2} \tag{3.19}$$

with σ_0 being the standard deviation of the data segment and σ_1, σ_2 the standard deviation of the first derivative and the second derivative of the data segment, respectively.

3.2.9 AR Coefficients

As different methods to calculate the Autoregressive (AR) coefficients give the same approximate coefficients, Burg's method was chosen to calculate the AR coefficients given its stability and reliability. Given a discrete signal x_n with $n \in [0, N]$, k coefficients a_n with $n \in [1, k]$ can be used to approximate the signal original values using a forward linear prediction, y_n, given by eq. 3.20 and a backward linear prediction, z_n, given by eq. 3.21.

$$y_n = -\sum_{i=1}^{k} a_i x_{n-i} \tag{3.20}$$

$$z_n = -\sum_{i=1}^{k} a_i x_{n+i} \tag{3.21}$$

In other words each y_n is a weighted sum of the k previous known values and each z_n is a weighted sum of the k next known values. The usual way to find the a_n coefficients is by minimising the sum of the squares of the error between the original and approximated values for the forward and backward linear predictions (eq. 3.22 - 3.23).

In Burg's method the main idea is to minimise the total sum of $F_k + B_k$. The detailed explanation of Burg's algorithm is beyond the scope of this work but for a complete explanation of this method

[2]Electroencephalography (EEG), is the recording of electrical activity along the scalp. EEG measures voltage fluctuations resulting from ionic current flows within the neurons of the brain.

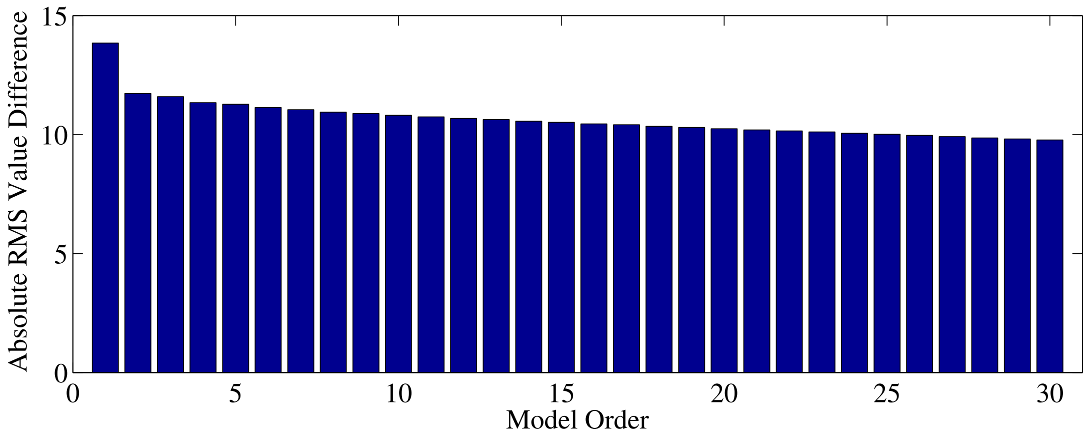


Figure 3.7: Model order and the difference (in RMS) between the original data and the reconstructed data using the model coefficients. The RMS tests were performed in 188,416 samples (1472 data segments of 128 ms each), averaged and subtracted with the real RMS value of the data segments. The time cost was performed in the same samples using the timeit function, available at http://www.mathworks.com/matlabcentral/fileexchange/18798-timeit-benchmarking-function

please refer to [13].

$$F_k = \sum_{n=k}^{N} (x_n - y_n)^2 \tag{3.22}$$

$$B_k = \sum_{n=k}^{N} (x_n - z_n)^2 \tag{3.23}$$

As the order of the AR model, k, influences the fidelity of the reconstructed signal, several tests were made to the data in order to get the maximum amount of fidelity to the original data while minimising the order of the model (fig. 3.7 - 3.8). Taking this results into account we can conclude that there is not a significant improvement with increasing order of the model and the computational cost increases linearly with the model order. Although the computational time between models order does not differ significantly, there is also not a significant improvement in model fidelity from a low level order from a higher one. Furthermore, the amount of data generated would be much larger with a higher order model without guaranteeing a better classification result. As such, a model of order 7 was chosen.

3.3 Dimensionality Reduction

Considering all the data features described in the previous section we have: ZC (1 feature), WL (1 feature), SSC (1 feature), Skewness (1 feature), RMS (1 feature), MAV (1 feature), IAV (1 feature), Hjorth parameters (3 features) and AR coefficients (7 features) which gives a total of 17×4 channels $= 68$ features. Although this is much less than the previous $200 \times 4 = 800$ variables, this is still a high number of values to feed to a classifier and it would be a very time consuming and inefficient task to do so.

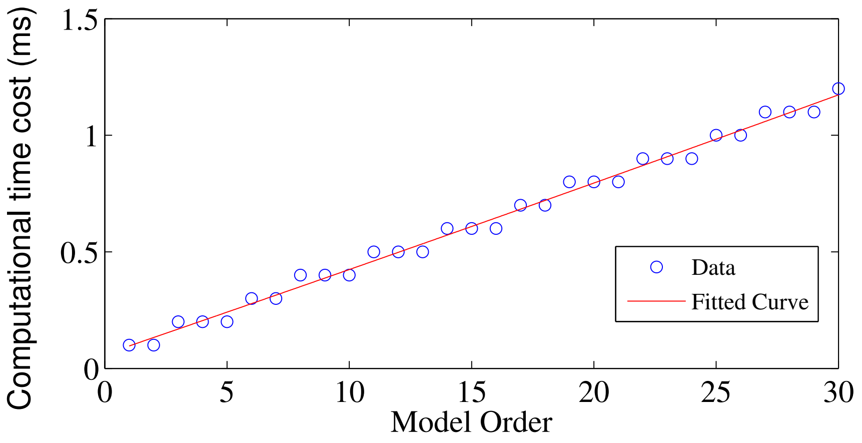


Figure 3.8: Difference between model order and the time cost.

In order to further reduce this feature vector to a more manageable size, a PCA algorithm is applied to the data before classification. PCA is one of a family of techniques for taking high-dimensional data, and using the dependencies between the variables to represent it in a more tractable, lower-dimensional form, without losing too much information. PCA is one of the simplest and most robust ways of doing such dimensionality reduction and it is used in numerous fields. The PCA steps are relatively simple and can be divided as the following [14]:

1. Organize the data features into a feature matrix $F_{n\times m}$, where n is the number of observations taken and m is the number of features. Thus, each row corresponds to a observation and each column is a feature of the observed data.

2. Subtract the mean from each feature column from the training data matrix, $F_{n\times 68}$, to obtain the matrix $Q_{n\times 68}$ [3].

3. Estimate the covariance matrix of the training data using

$$cov(Q) = \frac{1}{n-1}Q^TQ \tag{3.24}$$

4. Calculate the eigenvectors with the k highest eigenvalues from the $cov(Q)$ matrix (with k being the highest feature dimension wanted for classification) and form a feature vector (eq. 3.25).

$$\mathbf{FeatureVector} = (\mathbf{eigvector_1}, \quad \mathbf{eigvector_2}, \quad \mathbf{eigvector_3} \quad \dots \quad \mathbf{eigvector_k}) \tag{3.25}$$

5. Project the feature vector into the $Q_{n\times 68}$ to obtain the most k important feature columns .

$$\mathbf{FinalMatrix} = \mathbf{FeatureVector} \times Q_{n\times 68} \tag{3.26}$$

[3]In this work there is a additional step here, data scaling, which is not needed for the PCA variable reduction but it is needed to obtain higher classification results in the pattern recognition system.

3.4 Classification

A classification task usually involves separating data into training and testings sets. Each instance in the training set contains a target value, the class labels, and several attributes that belong to that class label (features or observed variables). The goal of any classification technique is to produce a model, based on the training data that predicts the target values of the test data given only the test data attributes.

3.4.1 Linear Discriminant Analysis

Discriminant analysis is a statistical technique to classify labels into mutually exclusive groups based on a set of features. LDA is closely related to PCA and factor analysis in that they both look for linear combinations of variables which best explain the data. LDA explicitly attempts to model the difference between the classes of data. PCA on the other hand does not take into account any difference in class, and factor analysis builds the feature combinations based on differences rather than similarities. Let the feature vector be X and the class label be Y. As LDA belongs to generative modelling, we try to estimate the within class density of X given the class label. Combined with the prior probability (unconditioned probability) of classes, the posteriori probability of Y can be obtained by the Bayes formula. Then, according to the Bayes rule, what we need is to compute the posterior probability:

$$\hat{P}(k \mid x) = \frac{P(x \mid k)P(k)}{P(x)} \tag{3.27}$$

where $\hat{P}(k \mid x)$ is the posteriori probability that a point x belongs to class k, $P(x \mid k)$ is the density of x conditioned on the class k, $P(x)$ is a normalisation constant, namely the sum over all classes of $P(x \mid k)P(k)$, i.e,

$$P(x) = \sum_{k=1}^{K} P(x \mid k)P(k) \tag{3.28}$$

and the priori probability $P(x)$ is the proportion of class k in x. This prior can be set in three ways: (i) uniformly, all classes have the same probability of appearing in the data,i.e, the prior probability of class k is 1 over the total number of classes, (ii) empirical, the proportion is set according to the number of appearances of class k in the training set, or (iii) by a specific numeric vector set by the user. Under LDA we assume that the density for x, given every class k is following a Gaussian distribution. The density formula for a multivariate Gaussian distribution[4] with mean μ_k and covariance matrix Σ_k at a point x is given by

$$P(x \mid k) = \frac{1}{\sqrt{(2\pi)^k |\Sigma_k|}} \exp\left(-\frac{1}{2}(x-\mu_k)^T {\Sigma_k}^{-1}(x-\mu_k)\right), \tag{3.29}$$

[4]The multivariate normal distribution is a generalisation of the univariate normal to two or more variables. It is a distribution for random vectors of correlated variables, each element of which has a univariate normal distribution. The multivariate normal distribution is parameterised with a mean vector, μ, and a covariance matrix, Σ. These are analogous to the mean μ and variance σ^2 parameters of a univariate normal distribution.

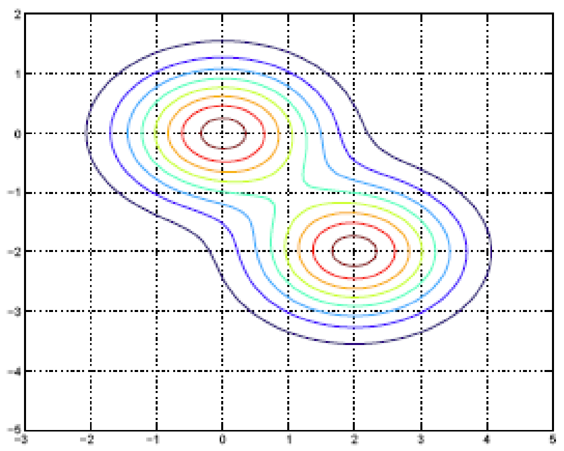


Figure 3.9: Example of two class densities.

where $|\Sigma_k|$ is the determinant of Σ_k, and Σ_k^{-1} is the inverse matrix of Σ_k. LDA assumes that all classes have the same covariance and only differ by their mean, i.e, $\Sigma_k = \Sigma$ for all classes. Since the covariance matrix determines the shape of the Gaussian density, in LDA, the Gaussian densities for different classes have the same shape, but are shifted versions of each other (different mean vectors). A example for two classes can be seen in figure 3.9.

The function used to fit a LDA model in Matlab was *ClassificationDiscriminant.fit*. The estimate of the class mean for weighted data are

$$\hat{\mu}_k = \frac{\sum_{n=1}^{N} M_{nk} w_n x_n}{\sum_{n=1}^{N} M_{nk} w_n} \tag{3.30}$$

where M_{nk} is a N-by-K class membership matrix that the element (n, k) is one if the observation n is from the class k and zero otherwise, and w_n is the weight of the observation n. By default this a vector of ones (no weights). For weighted data, assuming the weights sum to 1, the unbiased estimate of the pooled-in covariance matrix Σ is

$$\hat{\Sigma} = \frac{\sum_{n=1}^{N} \sum_{k=1}^{K} M_{nk} w_n (x_n - \hat{\mu}_k)(x_n - \hat{\mu}_k)^T}{1 - \sum_{k=1}^{K} \frac{W_k^{(2)}}{W_k}} \tag{3.31}$$

with

$$W_k = \sum_{n=1}^{N} M_{nk} w_n \tag{3.32}$$

is the sum of the weights for class k and

$$W_k^{(2)} = \sum_{n=1}^{N} M_{nk} w_n^2 \tag{3.33}$$

is the sum of square weights for class k.

The testing of offline data and the prediction of the class label for new data were done using the function *predict*. This function classifies so as to minimise the expected classification cost:

$$\hat{y} = \underset{y=1\,...\,K}{\arg\min} \sum_{k=1}^{K} \hat{P}(k \mid x) C(y \mid k) \tag{3.34}$$

where $\hat{y}$ is the predicted class, K is the number of classes, $\hat{P}(k \mid x)$ is the posterior probability of class k for observation x and $C(y \mid k)$ is k by k matrix with the costs of classifying an observation as y when its true class is k. By default, $C(i, j) = 1$ if $i \neq j$, and $C(i, j) = 0$ if $i = j$. In other words, the cost is 0 for correct classification, and 1 for incorrect classification.

3.4.2 Support Vector Machines

Support Vector Machines are among the most robust and successful classification algorithms [19, 21, 22, 23]. Support Vector Machines construct a hyperplane or a set of hyperplanes in a high (possibly infinite) dimensional space that has the largest distance to the nearest training data point of any class (functional margin), since in general the larger the margin, the lower the generalisation error[5].

Considering the training set, as a l pair of vectors (x_i, y_i), $x_i \in \mathbb{R}^n$, $i = 1, \ldots, l$ and $y_i \in \{1, -1\}^l$ vector which contains the class label, we want to find the maximum-margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. A hyperplane $\in \mathbb{R}^n$ is a set of points x_i that satisfy a linear equation

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = b \tag{3.35}$$

or in a more compact form,

$$w \cdot x - b = 0 \tag{3.36}$$

where w is the normal vector to the hyperplane and $b/\|w\|$ determines the offset of the hyperplane from the origin along the normal vector w. If the training data are linearly separable[6] then we can select two hyperplanes, H_1 and H_2, that separate the data, while maximising the distance between

[5]Generalisation error is a function that measures how well a learning machine generalises to unseen data (capacity of the machine). In an analogy a machine with too much capacity is like a botanist with a photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before; a machine with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree.

[6]Two sets are linearly separable if there exists at least one line in the plane with all of the points of one set on one side of the line and all the points of the other set on the other side.

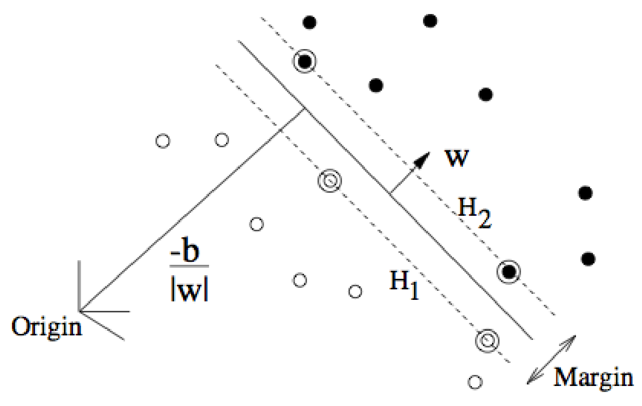


Figure 3.10: Linear separating hyperplanes for the separable case (H_1 and H_2). The support vectors are circled. Adopted from [21].

them. Thus, all data must satisfy the constraints:

$$w \cdot x_i - b \geq +1 \qquad \text{for } y_i = +1 \tag{3.37}$$

$$w \cdot x_i - b \leq -1 \qquad \text{for } y_i = -1 \tag{3.38}$$

These can be combined into one set of inequalities:

$$y_i(w \cdot x_i - b) \geq 1, \quad \forall i \tag{3.39}$$

As the distance between planes is $2/\|w\|$ we want to minimise $\|w\|$. The full, detailed explanation of the method from here is beyond the scope of this work but for a further explanation please refer to [21, 23].

For a linear non separable data, a hyperplane will be chosen so it splits the examples as cleanly as possible, while still maximising the distance to the nearest cleanly split examples. This optimisation problem is given by [23]:

$$\text{maximize} \qquad L_D \equiv \sum_i \alpha_i - \frac{1}{2}\alpha_i\alpha_j y_i y_j x_i \cdot x_j \tag{3.40}$$

$$\text{subject to} \qquad 0 \leq \alpha_i \leq C, \tag{3.41}$$

$$\sum_i \alpha_i y_i = 0 \tag{3.42}$$

With solution given by

$$w = \sum_{i=1}^{N_s} \alpha_i y_i x_i \tag{3.43}$$

where N_s is the number of support vectors. As a side note this optimisation problem only differs from the linear separable in α_i, which here has an upper bound C.

Originally (Vapnik and Lerner, 1963) SVMs were limited to linear classification. However, in 1992, Boser *et al* [24] suggested a way to create nonlinear classifiers by applying a *kernel trick*

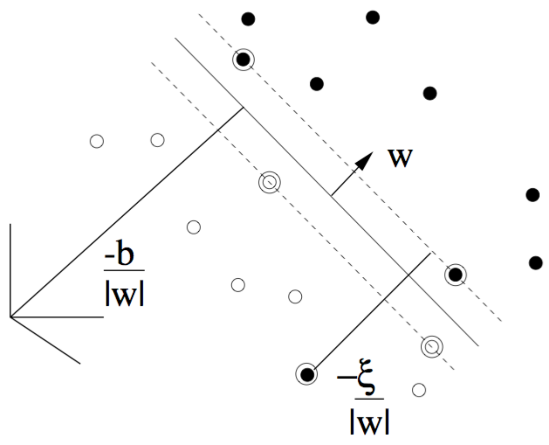


Figure 3.11: Linear separating hyperplanes for the non-separable case. Adopted from [21].

(originally proposed by Aizerman *et al* in 1964) to maximum-margin hyperplanes. This trick consists in mapping the data into to some other, possibly infinite dimensional, Euclidean space H using a function ϕ,

$$\phi : \mathbb{R}^d \mapsto H \tag{3.44}$$

As the training data only appears in the training problem as dot products (eq. 3.40) in H, the training algorithm only depends on the data through functions of the form $\phi(x_i)\cdot\phi(x_j)$. If a function $K(x_i, x_j) = \phi(x_i)\cdot\phi(x_j)$, a *kernel function* is used, we can use the algorithm without even explicitly knowing what ϕ is. The most popular kernels in the literature are the following:

- linear: $K(x_i, x_j) = x_i \cdot x_j$. This is the original kernel (eq. 3.40).
- polynomial with degree d: $K(x_i, x_j) = (\gamma x_i \cdot x_j + r)^d, \gamma > 0$.
- Radial Basis Function (RBF): $K(x_i, x_j) = e^{-\gamma\|x-y\|^2}$
- sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i \cdot x_j + r)$

with γ, r and d being the kernel parameters. In this work the RBF kernel was used as it has the ability to nonlinearly map samples and it is the most reasonable first choice for a kernel.

Grid-search and Cross-validation

The RBF kernel only has one parameter, γ, in addition to the penalty parameter of the error term, C (eq. 3.41). The optimal choice of these parameters is unique to a given set of input data and a parameter search must be done in order to find them. However, the optimal choice of these parameters is not the highest training accuracy per se, but the highest training accuracy that better predicts unknown data. As was discussed in the beginning of this section, one can divide the data into two sets, training and testing, and then obtain the parameters that score the highest testing accuracy, as this reflects best the performance of model when predicting unknown data. There is, however, a better method where all data is used as training data and can still prevent the over-fiting of the data (high training accuracy with low testing accuracy) known as cross validation.

In a v-fold cross-validation, the training data is divided in v subsets of equal size. Sequentially each one of these subsets is tested using the classifier, with the provided parameters, on the remaining $v - 1$ subsets. Thus, each instance of the whole training data is predicted once so the cross-validation accuracy is the percentage of data that is correctly classified.

In this work, and in most works as well, a grid-search on C and γ with cross-validation is used. Various pairs of (C, γ) are tried and the one with the highest cross-validation is picked. A good method to perform this grid-search is using exponentially growing sequences of C and γ (for example, $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$ and $\gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$). Although there are several more advanced methods, these methods take roughly the same amount of time as the grid-search to find good parameters (C, γ). Furthermore, these parameters are independent of each other, so the grid search process can be easily parallelised, while the advanced methods are typically a iterative process which can be hard to parallelise.

Since doing a exhaustive grid search can be very time consuming it is recommended that a coarse grid-search is done first and then fine tune the accuracy around the best region by using a finer grid search.

Multi-class SVMs

SVMs are, inherently, binary classifiers, i.e. they can only separate two distinct classes. However, a problem with more than two classes can be divided in several binary problems.There are several methods proposed in the literature, including one-vs-all and one-vs-one and Directed Acyclic Graph-Support Vector Machine (DAGSVM) [27]. The library used was libSVM [25] which uses one-vs-one method. The choice of this method is based on the results of [19]. In this approach, each class is compared to each other class. A binary classifier is built to discriminate between each pair of classes, while discarding the rest of the classes. This requires building $k(k-1)/2$ binary classifiers, where k is the number of different classes. When testing a new example, a voting is performed among the classifiers and the class with the maximum number of votes wins.

Chapter 4

Real Time Testing

In this chapter the procedures and results of real time testing will be discussed. Different sets of prediction methods were tested and each will be studied in the late sections.

4.1 Data Extraction Considerations

The accuracy of the classification, and more importantly its testing accuracy, of any set of movements can be degraded by a poor conditioning and procedures during the training phase [1, 2]. To minimise this issues a number of measures were devised.

Skin impedance: Low skin impedance can greatly lower the baseline amplitude (i.e., noise). To lower the impedance of the skin, a textile towel embed with alcohol can be used to remove any dirt or sweat from the contact area.

Baseline thresholds: A wrong baseline threshold can either give a lot of false positives or fails to detect a valid muscle contraction. In order to address this issue, before each training the optimal thresholds are estimated for each channel by asking the test subject to relax his arm as much as possible, without moving it from the rest position, for 5 seconds and then taking the maximum value of the moving average of the data taken during the last 3 seconds (the first 2 seconds are ignored). Furthermore, during the training session each movement was only recorded after a button is pressed in the GUI of the software to prevent unintentional recording of unwanted data.

Muscle fatigue: Muscle fatigue during the training phase can lower the accuracy of real time testing. To minimise this issue between each trial, which is composed of one of each movement, 10 seconds of rest is given.

Accuracy decay: As stated in [6] the accuracy of a trained model decays with time. This is due not only to physiological changes in the muscle tissue, but also to electrode displacement, skin changes, ambient noise, temperature. A simple way to counter this is to train a new model every few hours, although a more effective and elegant solution has been proposed by [6]. It consists on including a model correcting adaptation to account for these variations, a method which is already being used in the speech recognition community. For these reasons, the two different tests were taken in separate days instead of the same day.

4.2 Database

After every live training session data were stored in the disk for further analysis and model training. This data consists of both raw and feature data.

The raw data are divided in plain text (*.txt*) files, one for each movement, and one additional eXtensible Markup Language (XML) file (*info.xml*) which saves important information relative to the recording. This information comprises channels baseline, gesture set code and the number of training trials for the gesture set. In each of the text files each column corresponds to a channel (the first column to the first channel, the second column to the second channel and so on) and each line correspond to the analog (raw) value read in each channel. Furthermore, the time difference between each line is exactly 1 ms (the sampling rate is 1000 samples/s) and only the valid data are stored in these text files, i.e., data recorded when there was detected a muscle contraction and the record button in the GUI was pressed. From this raw data is possible to, afterwards, extract features with the desired channels, window size (W_s) and window step (W_t).

Data relative to the features extracted in real time are stored in a special Matlab (*.mat*) file. This file comprises three variables: the extracted features matrix, a column vector containing the corresponding class labels and a structure with information relative to this extraction. In the feature matrix, in each line corresponds to one observation relative to a pre-set time window (W_s) and each column is a feature of said time window. In this work, as was referred in section 3.3, 17 features were used for each channel and so the feature matrix has the structure

$$\textbf{fMatrix} = \begin{bmatrix} w_{1,1} & \cdots & x_{1,18} & \cdots & y_{1,35} & \cdots & z_{1,52} & \cdots & z_{1,68} \\ w_{2,1} & \cdots & x_{2,18} & \cdots & y_{2,35} & \cdots & z_{2,52} & \cdots & z_{2,68} \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \end{bmatrix} \tag{4.1}$$

where w is the feature data extracted from the first channel, x from the second, y from the third and z from the fourth. As each line corresponds to an observation then a line f_i and a class label l_i form n pairs of vectors (f_i, l_i), $f_i \in \mathbb{R}^n$, $i = 1, \ldots, n$ with $l_i \in \{1, N_{gestures}\}^n$ and with n being the number of observations taken.

The information variable consists of channels baseline, gesture set code, W_s, W_t and the number of gestures, $N_{gestures}$.

4.3 Six-Movement Classification

A possible way to control a robot with gestures consists in the movements described in figure 4.2. The possible corresponding motions to each gesture can be seen in table 4.2.

The placement of the electrodes can be seen in figure 4.1 and their respective targeted muscles in table 4.1.

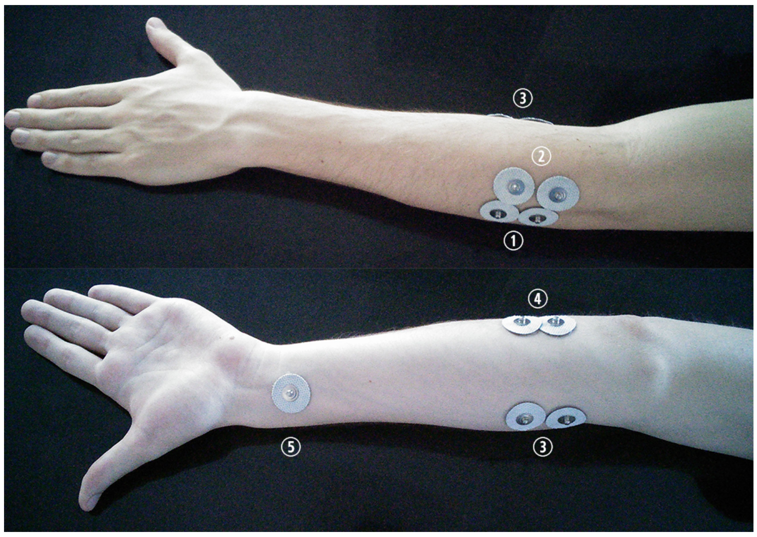


Figure 4.1: Posterior and anterior views of the electrode placement on the left arm.

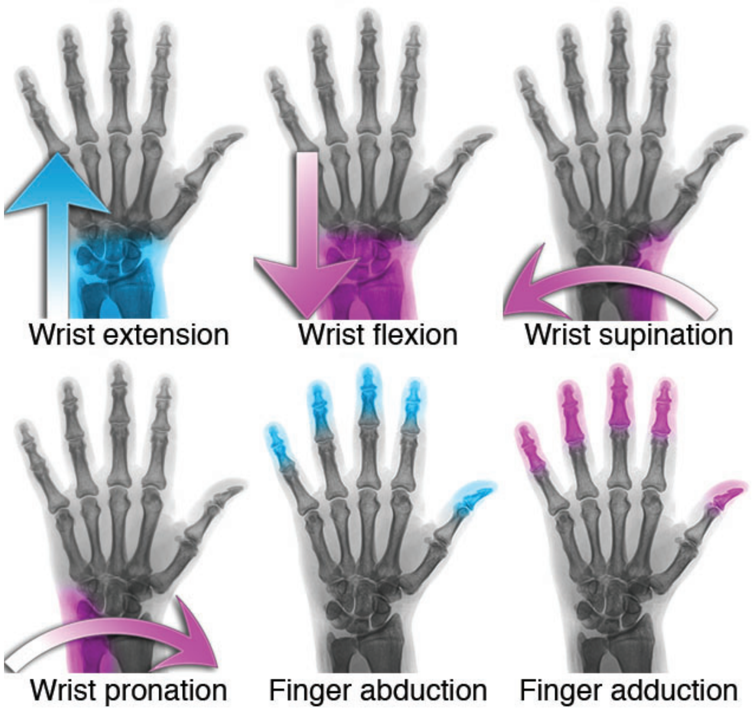


Figure 4.2: Array of six movements for classification.

Electrode Number	Targeted Muscle
1	Extensor digiti minimi
2	Extensor digitorum
3	Flexor carpi radialis
4	Flexor carpi ulnaris
5	Reference Electrode

Table 4.1: Correspondence between electrode placement and the target muscles. Muscles were located by palpation of the arm while the subject performed the movements.

Anatomical term	Anatomical motion	Robot Motion
Wrist extension	Bend hand up	Robot bends upward
Wrist flexion	Bend hand down	Robot bends downward
Wrist supination	Rotate hand to the left, palm facing up	Robot wrist rotates to the left
Wrist pronation	Rotate hand to the right, palm facing down	Robot wrist rotates to the right
Finger abduction	Open the hand, with the fingers spread apart	Robot gripper opens
Finger adduction	Close the hand, with the fingers close together	Robot gripper closes

Table 4.2: Correspondence between anatomical movements and the action for the robot to perform.

4.3.1 GUIs

In order to keep data more organised and displayed in a more intuitive way a set of GUIs were designed in Matlab using the GUIDE tool. A detailed summary of all the GUIs and screenshots can be seen in the annex A.

4.3.2 Testing Method

Although offline classification provided a general idea to the real accuracy of the system, a more real world test was needed. Inspired by the work done in [32] a FTAT was built. Each trial was composed by two phases. The goal of the first phase was to control a cursor in pseudo three-dimensional cartesian space and to steer it from the centre of the screen to a circular target that appeared with a given radius at a given location. Once near or on the target, the cursor radius must be reduced by the subject in order to fit inside the circular target. The second phase is similar to the first phase but now the cursor must be moved from the centre location to a more distant one and its radius must be enlarged in order to contain the target circle. The idea behind this test is to provide a simulation of a robot picking up a object in a cartesian space, transport it and place it in another location.

Subjects were instructed to move the cursor as fast as possible from its initial position in the circle in the centre of the space into the target circle and to dwell inside the target for a full second to complete target acquisition. A trial failure was determined by (i) a time-out, (i.e., when target acquisition in either phase took longer than 30 seconds); (ii) by excessive overshooting (i.e., when the cursor moved into and immediately out of the target more than 4 times; such targets were considered too difficult), or (iii) when cursor movement was uncontrolled and the cursor bumped into the borders of the test space.

4.3.3 Testing procedure

In order to get a broader and better view of the classification accuracy across different subjects, testing sessions were proposed. Each session was initiated by a training procedure, where each motion was repeated 4 times in a loop (all motions were asked in a row, followed by a rest period of one minute, and repeated again 3 times), with the $W_s = 200$ ms and $W_t = 50$ ms (75% segment overlap). After the training session the data were shuffled and separated into training data (80%) and testing data (20%). Training data were then scaled to the interval to have zero mean and unit variance, which improves classification accuracy, greatly improves training time, and removes the scaling step of PCA which then follows, performing a dimensionality reduction to only the top 15 features (data corresponding to the eigenvectors with the greater 15 eigenvalues). The resulting data were trained using SVM or LDA depending on the test. The SVMs were trained with the optimal parameters from a grid-search procedure with limits being $C \in [1, 50]$ and $\gamma \in [0.01, 0.3]$ with 7 values in each range, which gives $7^2 = 49$ tests. LDA classification accuracy is not very dependent on parameters, so no optimisation was performed.

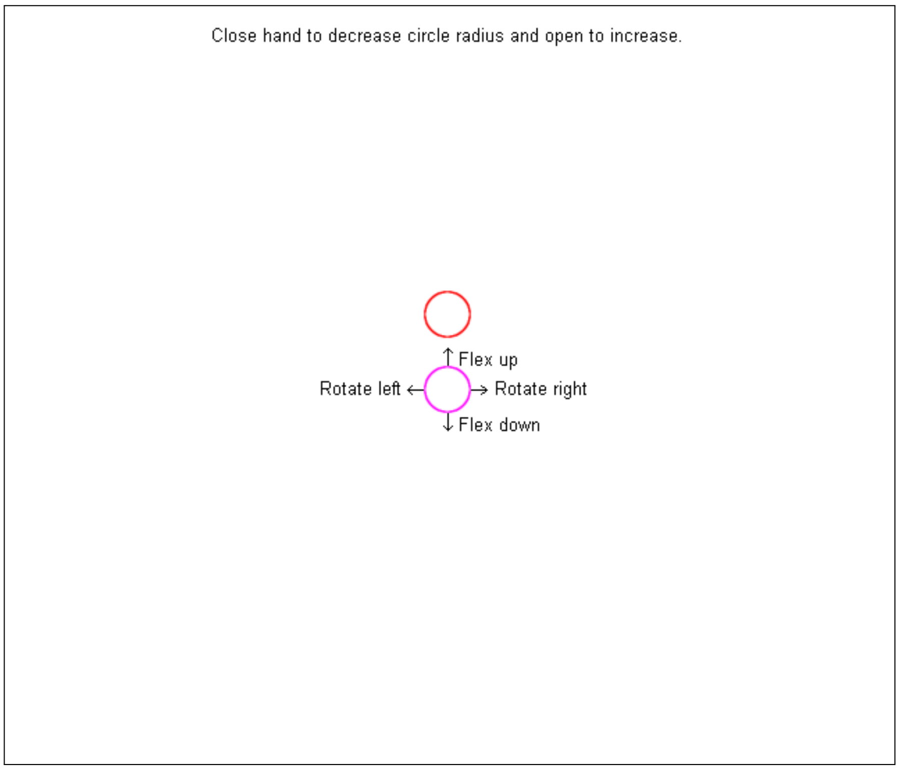


Figure 4.3: The Fitts interface. A detailed explanation and a screenshot of the tuning GUI can be found in the annex A.

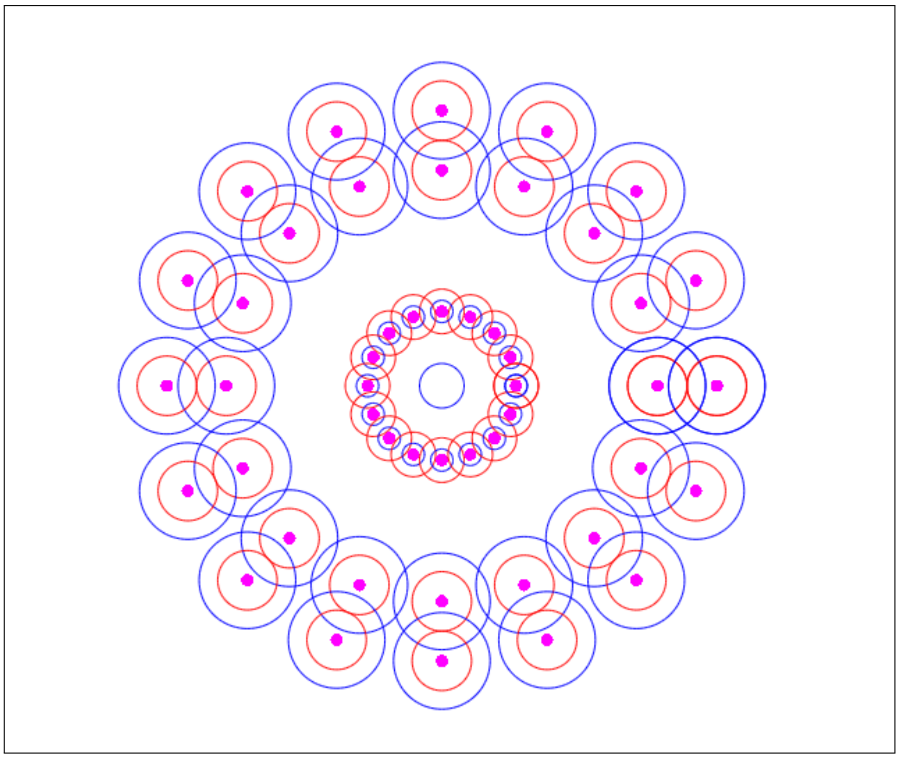


Figure 4.4: Widths and locations of all possible targets, in red, and the positional circle in the centre in blue. The positional circle has a maximum and minimum radius and its drawn in the figure around the targets. The GUI radius widths are as follows: (i) the full magenta circles are 0.75pts; (ii) the targets are 3pts in the inner layer and 4pts in both the outer layers, and (iii) the positional circle default radius is 3pts. The maximum and minimum radius are 6.5pts and 1.5pts respectively.

Ring	Number of 1 DOF targets	Number of 2 DOF targets	Total number of targets
Inner ring	4	4	8
First outer ring	2	2	4
Second outer ring	2	2	4
Totals:	8	8	16

Table 4.3: Target count of each testing session. Each ring target position was completely randomised and independent of each other and of the other rings (from the available positions seen in 4.4). Furthermore, the outer ring asked was also completely randomised, i.e., after the inner target test was complete, the subject could be asked to move the cursor to the second outer ring in consecutive trials.

Two tests in two different days were conducted to test the classification accuracy versus real world performance versus classifier. Each test consisted of a path from the centre to a inner circle and then from the centre to one of two outer ring circles, asked in a random order. A test session was composed of 16 targets (8 trials) and the target count can be seen in table 4.3 for both DOF.

After training either classification method, subjects were first allowed to familiarise themselves with the system by moving the cursor at various speeds. In this training subjects could test the cursor at low or high speed to verify good control. Speed was controlled by proportional control using the MAV (section 3.2.6) of the extracted data segment for all channels, which scales approximately proportional with the force applied [1]. A velocity control was then applied in order to smooth out possible misclassifications, i.e, the last predicted gesture speed was proportional to how many of the last m gestures were the equal to it, with m being a configurable parameter.

Real time classification was done using a single window classification method, i.e., a class decision is produced based on a single data segment (the data segment corresponding to $0-199$ms).

To comprehensively evaluate each class prediction strategy, a set of four quantitative performance metrics that can be seen in table 4.4 were tracked.

4.3.4 Training Results

As SVMs requires two parameters to train a model, and the classification accuracy can vary widely when using different parameters, a grid search was performed using a 5-fold cross-validation technique on 80% of the training data with the remaining 20% used for testing when the best parameters were found. The results of this cross-validation can be seen in figure 4.5.

Metric	**Units**	**Description**
Completion rate	%	Ratio of successfully completed targets to total number of targets.
Overshoot	%	Ratio of overshoots to number of targets. An overshoot was defined by a cursor entry into and exit of the target circle before the dwell time was accomplished.
Reaction time	Seconds	Time between target appearance and first move of the cursor.
Completion time	Seconds	Time between target appearance and task completion. This metric does not include failed test results times.

Table 4.4: Performance metrics used in each test session.

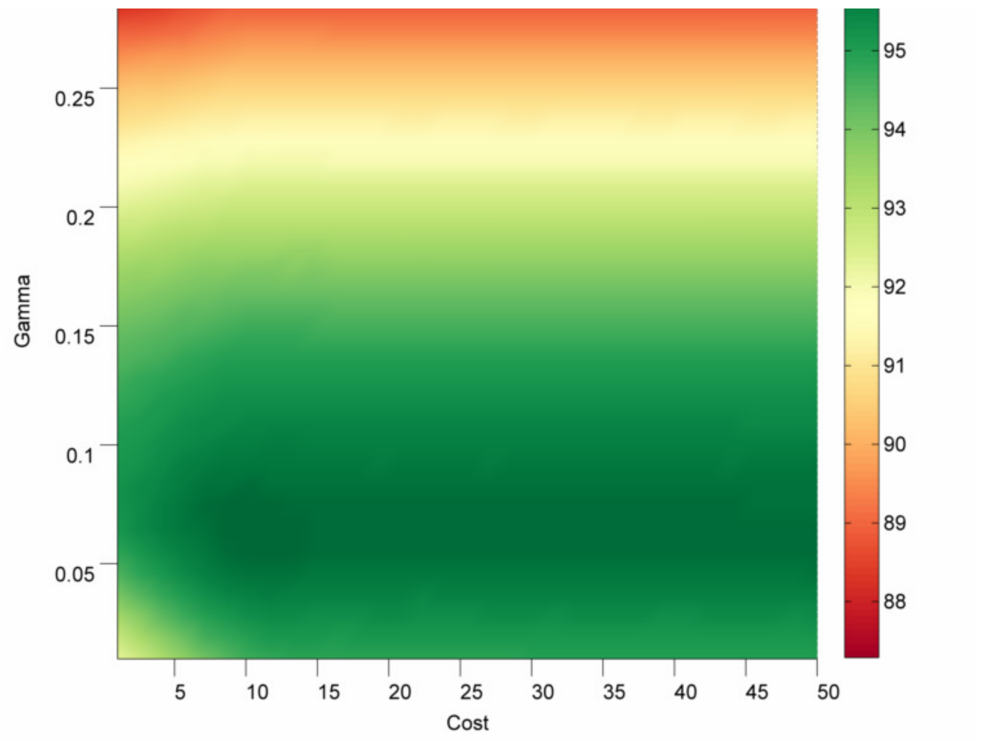


Figure 4.5: Mean Grid Seach of the SVMs using 5-fold cross-validation across all subjects. The optimal parameters are very well defined for this array of movements and are situated in the region around $C \in [8, 50]$ and $\gamma \in [0.05, 0.08]$

.

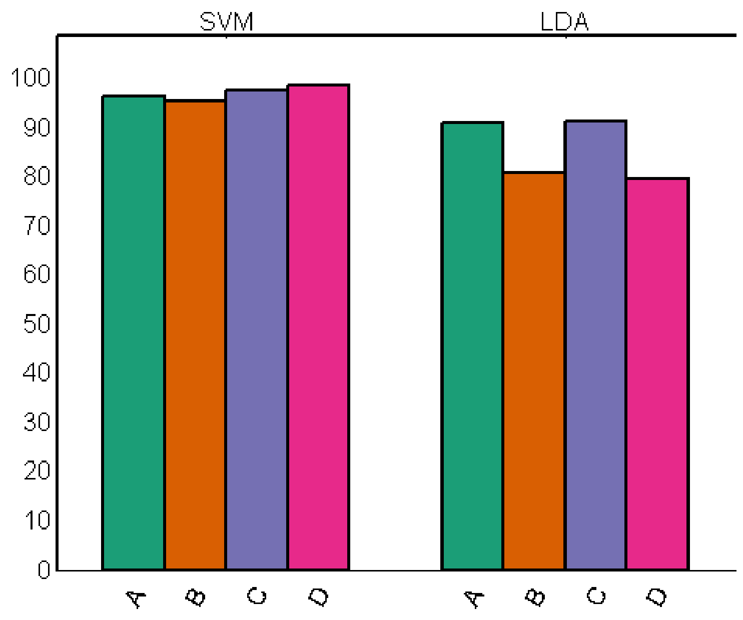


Figure 4.6: Testing accuracies for both classifiers per subject.

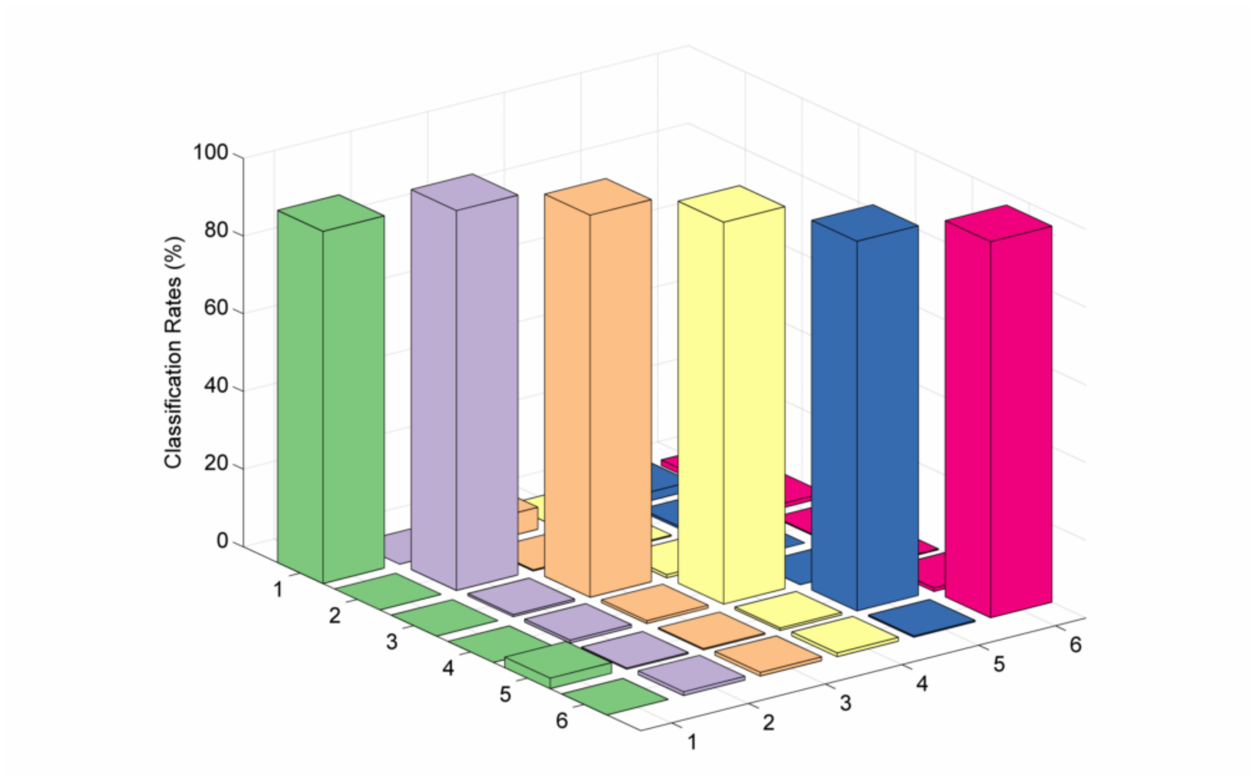


Figure 4.7: Averaged and normalised confusion matrix across all test subjects for SVM. The testing data was used to produce this result.

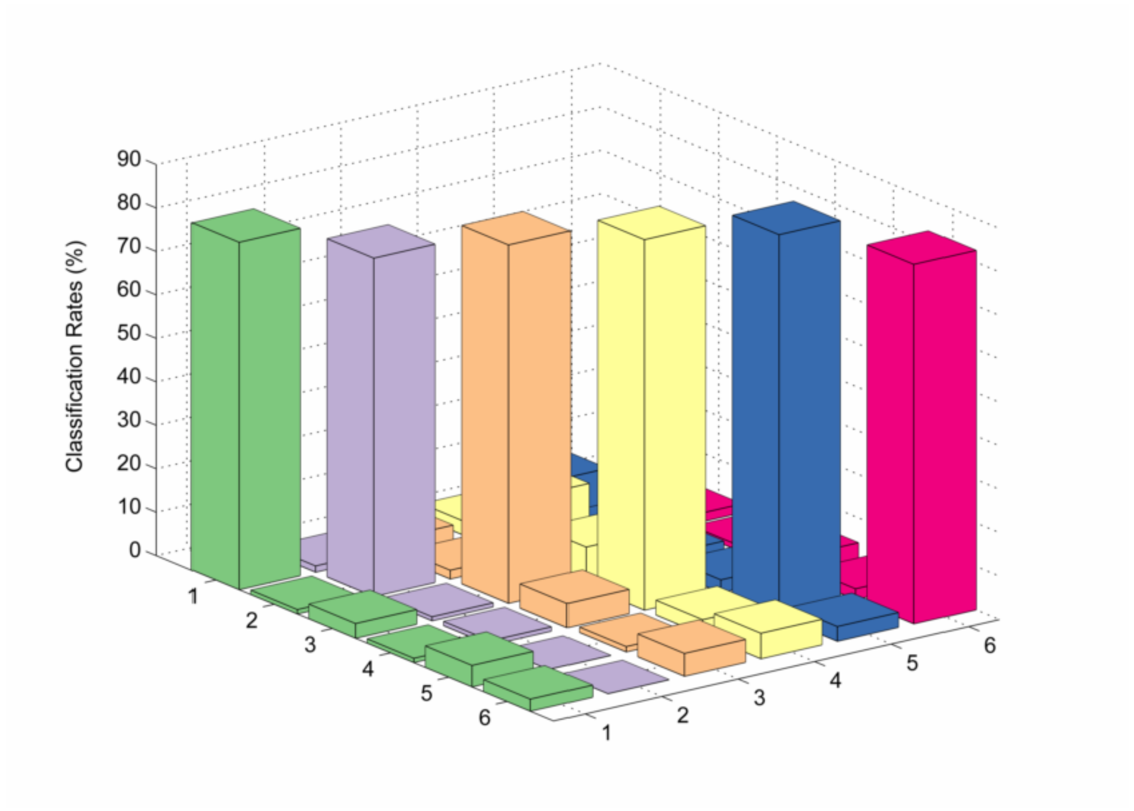


Figure 4.8: Averaged and normalised confusion matrix across all test subjects for LDA. The testing data was used to produce this result.

4.3.5 Classification and testing results

A total of four body able subjects were submitted to testing (one female and 3 males with ages 22.5 ± 1). The results of the testing can be seen in the table 4.5 and in table 4.6. Despite having lower testing accuracy (fig. 4.6), LDA scored higher results in the FTATs (figure 4.11b) while having a much lower standard deviation. Furthermore, all test subjects reported that the control of the cursor using LDA was much more smooth and had less misclassifications of motions. On the subject of completion rate, in test results of subject C using SVM, despite having very high testing accuracy (97.46 %), when the FTAT test was in course the subject reported that cursor wouldn't move according to the motion commanded, and that it was very hard to complete the test. After the test was over, LDA was applied to the same training data and this classifier could classify the motions with a much better degree of accuracy than that of SVM. At the time of this writing, the problem is still not evident and being so, more tests would be needed to identify its source.

4.3.6 Robot Simulator

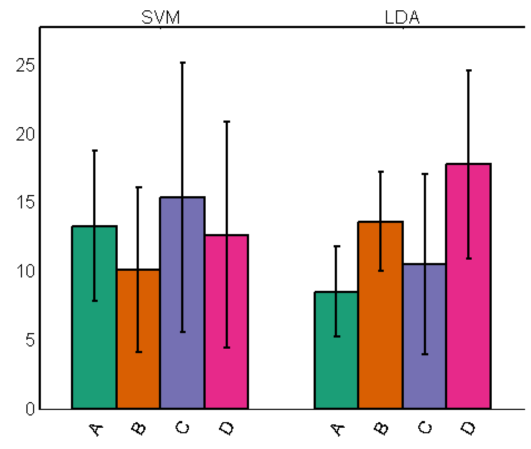
A robot simulator was built in order to test the applicability of the motions to a real robot. This robot used same velocity control used in the FTAT tests and the available motions can be seen in the table 4.2.

DOF	Subject	Reaction time	Completion time	Overshoot	Completion rate
1 DOF					
	A	1.30 ± 0.31	13.27 ± 5.44	12.5	100
	B	1.12 ± 0.14	10.09 ± 5.99	37.5	100
	C	1.29 ± 0.42	15.37 ± 9.81	87.5	25
	D	1.47 ± 0.62	12.61 ± 8.21	75	50
Sub Total					
		1.29 ± 0.41	12.19 ± 6.26	53.13 ± 34.42	68.75 ± 37.5
2 DOF					
	A	1.02 ± 0.36	14.02 ± 4.77	25	100
	B	1.17 ± 0.45	11.61 ± 2.46	0	100
	C	1.10 ± 0.48	18.84 ± 7.87	125	37.5
	D	1.45 ± 0.46	19.21 ± 7.34	0	75
Sub Total					
		1.184 ± 0.45	15.07 ± 5.93	37.5 ± 59.51	78.13 ± 29.54
Totals		1.24 ± 0.43	13.72 ± 6.19	45.31 ± 45.78	73.44 ± 31.65

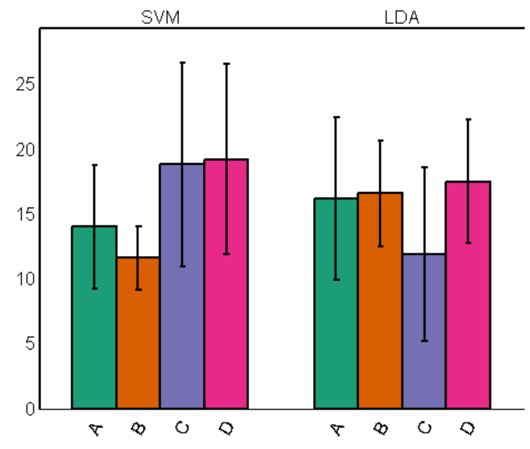
Table 4.5: Performance metrics across all subjects using the SVM classifier.

DOF	Subject	Reaction time	Completion time	Overshoot	Completion rate
1 DOF					
	A	1.26 ± 0.2	8.49 ± 3.26	12.5	100
	B	0.59 ± 0.47	13.59 ± 3.61	37.5	100
	C	0.96 ± 0.72	10.47 ± 6.56	12.5	87.5
	D	1.01 ± 0.90	17.75 ± 6.84	62.5	87.5
Sub Total					
		0.96 ± 0.64	12.47 ± 6.06	31.25 ± 23.94	93.75 ± 7.22
2 DOF					
	A	0.95 ± 0.53	16.21 ± 6.25	12.5	100
	B	1.02 ± 0.58	16.59 ± 4.10	125	75
	C	1.20 ± 0.49	11.92 ± 6.72	25	87.5
	D	0.51 ± 0.45	17.51 ± 4.79	25	100
Sub Total					
		0.92 ± 0.55	15.61 ± 5.74	46.88 ± 52.42	90.63 ± 11.97
Totals		0.94 ± 0.6	14.02 ± 6.06	39.14 ± 38.64	92.19 ± 9.3

Table 4.6: Performance metrics across all subjects using the LDA classifier.

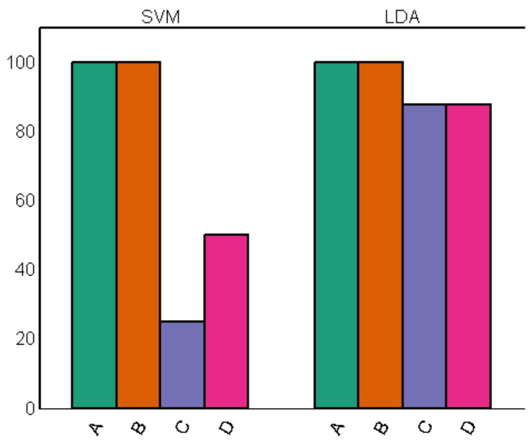


(a) 1 DOF

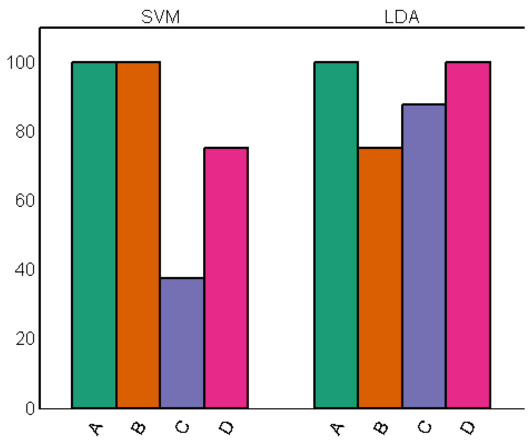


(b) 2 DOF

Figure 4.9: Completion times for both DOFs and for both classifiers per subject.

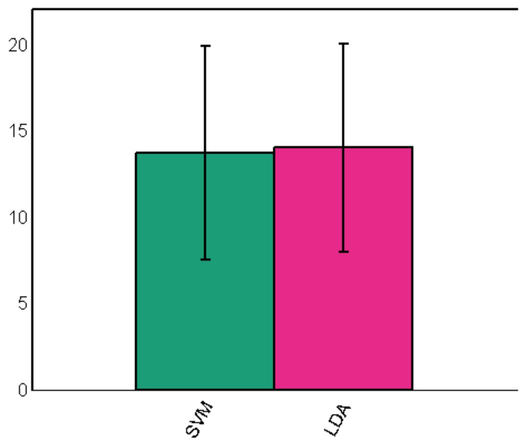


(a) 1 DOF

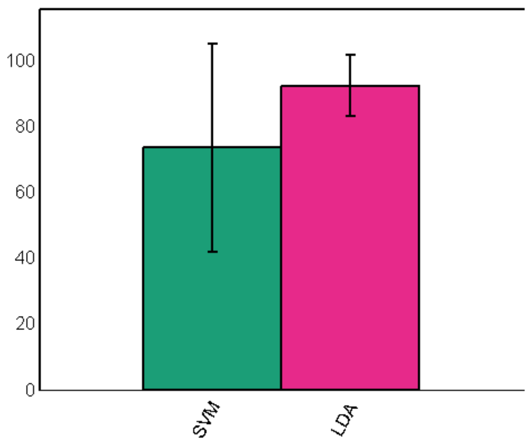


(b) 2 DOF

Figure 4.10: Accuracy rates for both DOFs and for both classifiers per subject.



(a) Completion time



(b) Completion rate

Figure 4.11: Overall completion times and target accuracy rates for both methods.

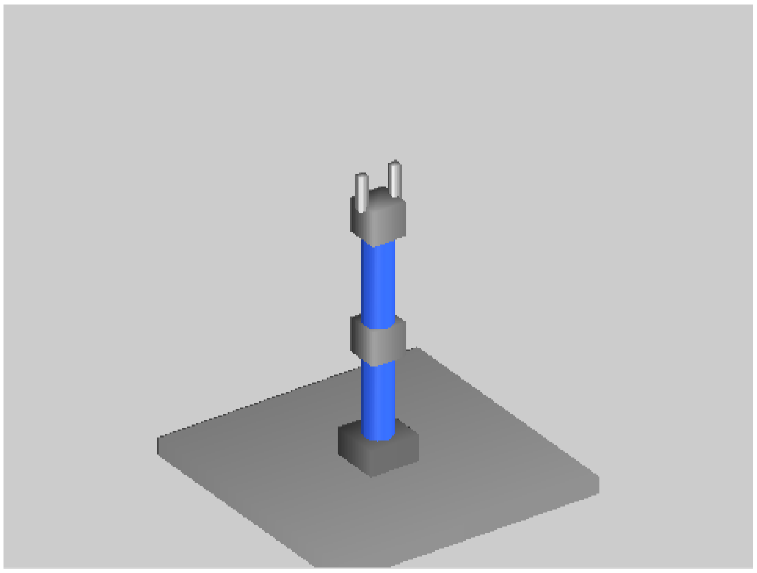


Figure 4.12: Robot simulator built within the Matlab environment.

Chapter 5

Conclusion and Future Work

In this chapter we discuss the final results of the applied methods in 5.1 and in section 5.2 outline some methods that we would like to try in future works.

5.1 Conclusion

The fundamental goal of this thesis was to test the possibility of controlling a robot using real time classification . Based on this goal, a platform was developed where any given set of features can, not only be trained, but tested by a model that simulates real world conditions. Our methods strongly suggest that such control is indeed possible with a high degree of accuracy.

Although offline classification accuracies are important and provide a general idea of a possible real world performance, they do not evaluate its applicability in real conditions. With this in mind a FTAT test was devised that tried to replicate a more real situation by a completion of a task, and proved to be not only challenging but also an excellent measure of applicability.

In direct comparison with both classification methods, LDA has proven to be the a stronger candidate than SVM to such task. Although SVM scored higher offline training accuracies than LDA, its capability at classifying new data was worse than LDA, and in the case of test subject C, it was unable to predict any movement reliably. The source of this issue with SVMs was not clear and further testing would be needed to identify it.

With an average completion rate of 92% in the FTAT tests, using LDA, and with an average completion time of 14 seconds, the proposed methodology can be considered a successful approach to myoelectric signal decoding.

5.2 Future Work

In the near future, we would like to test the continuous classification method using majority voting. This method, if successful, can greatly reduce the window segment length used for classification (200ms) and improve the overall responsiveness of the system.

One of the major difficulties of applying any classification method based on EMG signals in commercial applications is the accuracy decay that occurs over time. Some studies suggest that this accuracy decay is around 3.3% per hour, which would require a retrain multiple times in a day, rendering EMG control very difficult to implement in a real world scenario. A study in this area would be a very interesting step to take, and would bring this type of control closer to its many possible commercial applications.

We used a relative small number of people in this thesis. As a possible improvement, a more numerous and age diversified test subjects could be asked to take the tests. Also, only non-disabled people have participated on this study, and as myoelectric control can be applied to disabled people as well, a more broad study could be done. And since future applications could be a devices able to help in physiotherapy or EMG controled prosthesis, the study of ill arm signals should be taken into account.

Although robot control was simulated in a virtual environment, controlling a real robot in real time, using our methodology, would also be a very interesting step.

Bibliography

[1] P. Konrad, The ABC of EMG: A Practical Introduction to Kinesiological Electromyography, (2006), Noraxon, Available at http://www.noraxon.com/docs/education/abc-of-emg.pdf (Accessed: July 2014).

[2] Dennis Tkach, He Huang, Todd A Kuiken, Study of stability of time-domain features for electromyographic pattern recognition, Journal of NeuroEngineering and Rehabilitation 2010, 7--21

[3] K. Englehart, B. Hudgins, A robust, real-time control scheme for multi- function myoelectric control, IEEE Trans. Biomed. Eng. 50 (7) (2003) 848--854.

[4] K. Englehart, B. Hudgins, P.A. Parker, A wavelet-based continuous classification scheme for multifunction myoelectric control, IEEE Trans. Biomed. Eng. 48 (3) (2001) 302--310.

[5] Rami N. Khushaba, Sarath Kodagoda, Diaki Liu, Gamini Dissanayake, Muscle computer interfaces for driver distraction reduction, Computer methods and programs in biomedicine 110 (2013) 137--149.

[6] K.R. Wheeler, Device control using gestures sensed from EMG, Soft Computing in Industrial Applications, SMCia/03, Proceedings of the 2003 IEEE International Workshop (2003), 21--26 .

[7] B. Hudgins, P. Parker, R. Scott, A new Strategy for multifunction myo-electric control, IEEE Trans. Biomed. Eng. 40 (1) (1993) 82--94.

[8] Angkoon Phinyomark, Chusak Limsakul, Pornchai Phukpattaranont, A Novel Feature Extraction for Robust EMG Pattern Recognition, Journal of Computing, Volume 1, Issue 1, December 2009, 71--80, ISSN: 2151-9617.

[9] B. Hjorth, EEG analysis based on time domain properties, Electroencephalography and Clinical Neurophysiology, vol. 29, no. 3, Sept. 1970, pp. 306-310, doi:10.1016/0013-4694(70)90143--4.

[10]] B. Hjorth, The physical significance of time domain descriptors in EEG analysis, Electroencephalography and Clinical Neurophysiology, vol. 34, no. 3, Mar. 1973, pp. 321-325, doi:10.1016/0013-4694(73)90260-5.

[11] M. Mouz6-Amady, F. Horwat, Evaluation of Hjorth parameters in forearm surface EMG analysis during an occupational repetitive task. Electroencephalography and clinical Neurophysiology 101 (1996) 181--183.

[12] Hjorth B. Time domain descriptors and their relation to a particularmodel for generation of EEG activity, CEAN – Computerized EEG analysis. Stuttgart: Gustav Fischer Verlag; 1975. p. 3–8.

[13] Cedrick Collomb, Burg's Method, Algorithm and Recursion, November 2009, Available at http://ccollomb.free.fr/ (Accessed: July 2014).

[14] Smith, L. I. (2002), A tutorial on principal components analysis , Technical report, Cornell University, USA.

[15] S. Micera, G. Vannozzi, A. M. Sabatini and P. Daro, Improving detection of muscle activation intervals, IEEE Engineering in Medicine and Biology Magazine, 20(6), (2001), 38--46.

[16] P. Bonato, T. D'Alessio and M. Knaflitz, A statistical method for the measurement of muscle activation intervals from surface myoelectric signal during gait, IEEE Trans. on Biomedical Engineering, 45(3), (1998), 287--299.

[17] Winter DA, Pathologic gait diagnosis with computer-averaged electromyographic profiles, Arch Phys Med Rehab 1984, 65:393-398.

[18] Kerem Tuncay Özgünen, Umut Çelik, Sanlı Sadi Kurdak, Determination of an optimal threshold value for muscle activity detection in EMG analysis, Journal of Sports Science and Medicine (2010) 9, 620--628.

[19] Chih-Wei Hsu, Chih-Jen Lin, A Comparison of Methods for Multi-class Support Vector Machines, IEEE Transactions on Neural Networks 13 , no. 2 (2002): 415--425.

[20] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, A Practical Guide to Support Vector Classification, Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (2003).

[21] Christopher J.C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2 (1998): 121--167.

[22] Mohamed Aly, Survey on multiclass classification methods, Neural networks (2005): 1--9.

[23] Corinna Cortes, Vladimir Vapnik, Support-vector networks, Machine Learning, pages 273--297, 1995.

[24] B. E. Boser, I. M. Guyon, V. N. Vapnik, A Training Algorithm for Optimal Margin Classifiers, in David Haussler, ed., Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT'92) , ACM Press, Pittsburgh, PA, USA , pp. 144--152, 1992.

[25] Chih-Chung Chang, Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[26] Ryan Rifkin, Aldebaro Klautau, Parallel networks that learn to pronounce english text, Journal of Machine Learning Research, pages 101--141, 2004.

[27] J. Platt, Sequential minimal optimization: A fast algorithm for training support vector machines, (1998).

[28] Ashkan Radmand, Erik Scheme, Peter Kyberd, Kevin Englehart, Investigation of optimum pattern recognition methods for robust myoelectric control during dynamic limb movement, Conference paper.

[29] A. Ameri , E. Scheme , E. Kamavuako , K. Englehart and P. Parker, Real time, simultaneous myoelectric control using force and position based training paradigms, IEEE Trans. Biomed. Eng., vol. 61, no. 2, pp.279 -287 2013.

[30] Young AJ, Smith LH, Rouse EJ, Hargrove LJ, A comparison of the real-time controllability of pattern recognition to conventional myoelectric control for discrete and simultaneous movements, J Neuroeng Rehabil 2014, 11:34--44.

[31] Guanglin Li, Electromyography Pattern-Recognition-Based Control of Powered Multifunctional Upper-Limb Prostheses, Advances in Applied Electromyography, Prof. Joseph Mizrahi (Ed.), ISBN: 978-953-307-382-8, InTech, DOI: 10.5772/22876, 2011. Available from: http://www.intechopen.com/books/advances-in-applied-electromyography/electromyography-pattern-recognition-based-control-of-powered-multifunctional-upper-limb-prostheses (Accessed: July 2014).

[32] Wurth and Hargrove: A real-time comparison between direct control, sequential pattern recognition control and simultaneous pattern recognition control using a Fitts' law style assessment procedure. Journal of NeuroEngineering and Rehabilitation 2014 11:91.

[33] Bitalino Manual. Available at http://www.biosignalsplux.com/downloads/bitalino_manual/manual.html (Accessed: September 2014).

Appendices

Appendix A

GUIs

Item No.	Description
1	File menu. it gives access to *test setup*, *options* and *data analysis*. • *test setup*: tests current bitalino setup for conditions analysis. • *options*: gives the options GUI seen in fig. A.2. • *data analysis*: Gives the Data Analysis GUI seen in fig. A.4.
2	Save model menu. Gives access to *from live data*, and *from saved data*. • *from live data*: used get features of a certain moveset. It involves asking the user for the moveset motions in order and getting features from time windows. Before training, it needs to be provided with a codename, a window size and a windows step. It saves the raw signals, the feature matrix, the labels matrix and various info about the train parameters. • *from saved data*: used to retrain a saved raw model according to a any window size and step, so the subject won?t have to train again with a different window.
3	Log window. Provides various information to the user.

4	Train Model. Trains a SVM/LDA model based on a selected feature/label matrix with a parameter search method chosen by the user in the case of the choice being SVM. This methods are: • Search in a predefined range, with step between values given by the user. Lower steps imply more tests. • Search in a given range: user inputs a search range for the gamma and cost variables, as well as the number of elements in each range (10^2 tests) and the top elements to show in a table. • Input directly the cost and gamma variables. In the range methods a graph is shown to display the accuracy vs parameters. The missing values between tests are filled by interpolation.
5	A trained gesture is asked in random order for a given number of tests. This is can be used to determine the real world accuracy of the trained model in the first milliseconds of a gesture. The end results are saved and can be visualised and compared in the data analysis GUI (fig. A.4).
6	Control a 3d simulated robot with trained gestures.
7	Displays the current gesture the user needs to perform when training the model, or when testing in (5).
8 - 11	Shows the individual processed signal of each channel, plus a square wave to show a muscle contraction. An example can be seen in 3.6.
12 - 15	Slider used the adjust the baseline threshold,i.e., the threshold that decides when a channel is experiencing muscle contraction.
16	Indicators of muscle activity on each channel.
17	Toggle button to start recording the signals or start the test when applicable.

Table A.1: Item numbers and a short description for the main GUI.

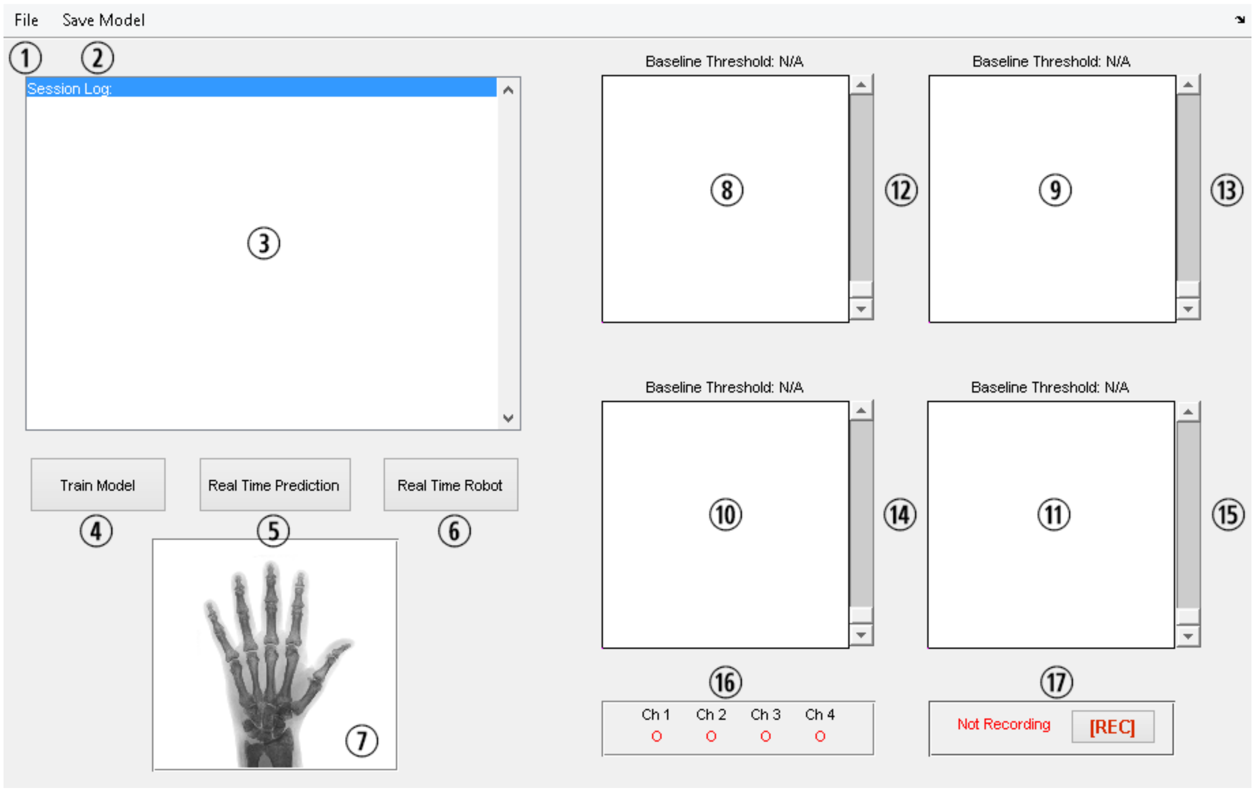
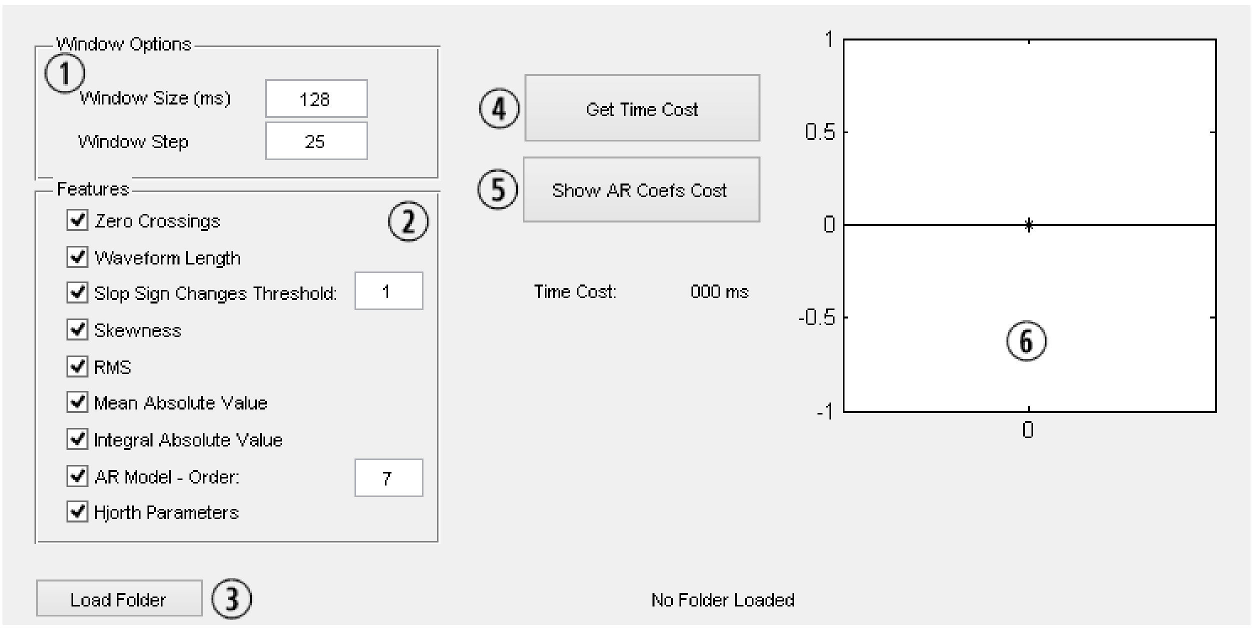


Figure A.1: The main GUI.



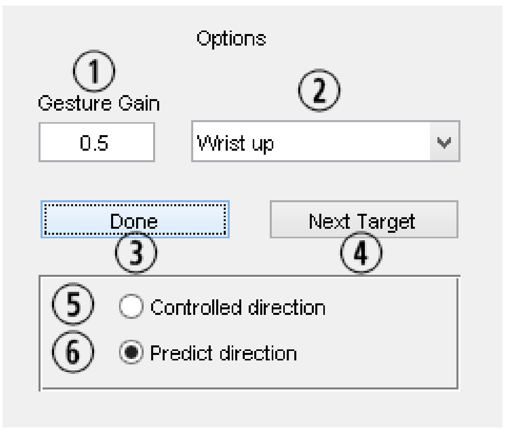
Window Size (ms): 128
Window Step: 25

- ✓ Zero Crossings
- ✓ Waveform Length
- ✓ Slop Sign Changes Threshold: 1
- ✓ Skewness
- ✓ RMS
- ✓ Mean Absolute Value
- ✓ Integral Absolute Value
- ✓ AR Model - Order: 7
- ✓ Hjorth Parameters

Time Cost: 000 ms

No Folder Loaded

Figure A.2: The options GUI.

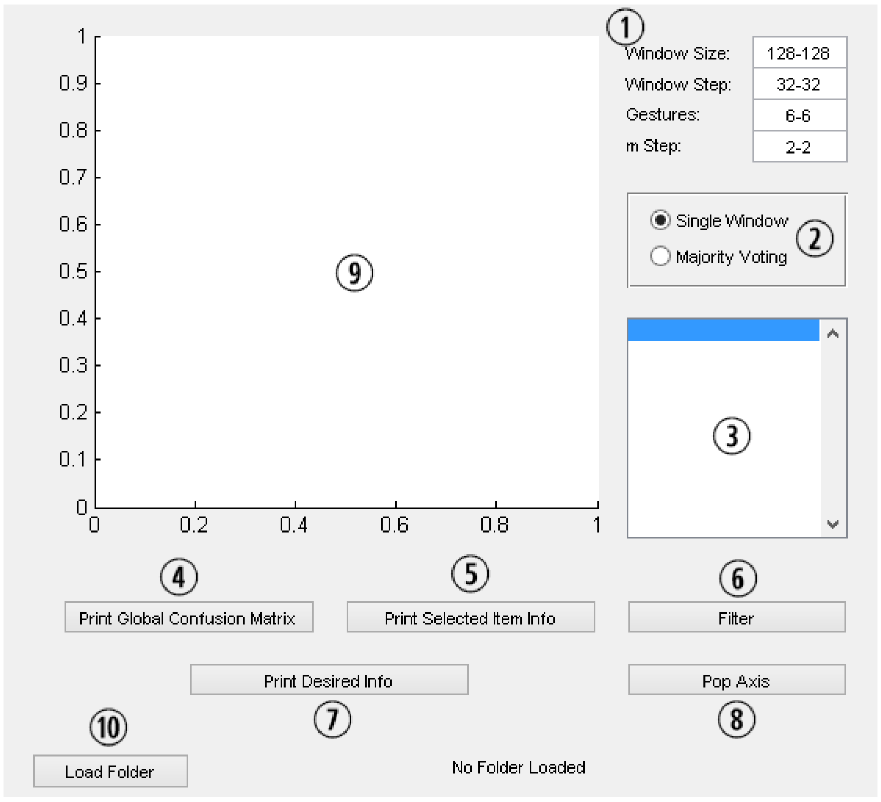


Gesture Gain: 0.5

Wrist up

- Controlled direction
- ● Predict direction

Figure A.3: The tuning GUI for the Fitts test .



Window Size:	128-128
Window Step:	32-32
Gestures:	6-6
m Step:	2-2

- ● Single Window
- Majority Voting

No Folder Loaded

Figure A.4: The data analysis GUI.

Item No.	Description
1	Select window size and window step.
2	Toggle features on/off.
3	Load a folder to get metrics from. Must contain raw data from a previous training session.
4	Get time cost of the selected features using the selected raw data folder.
5	Load a folder to get metrics from. Must contain raw data from a previous training session.
6	Get a histogram showing the AR model order vs the difference in RMS between the original signal and the reconstructed one using the AR chefs. It also displays the time cost of each model order.
7	Display the results from the buttons 4 and 5.

Table A.2: Item numbers and a short description for the options GUI.

Item No.	Description
1 - 2	Tune the gain for each individual feature. A feature is selected from the drop-down menu (2), and its gain can be adjusted in (1).
3	Start the test session with the tuned values.
4	Draw the next target. This is used to experiment cursor control. A limit of 4 targets can be drawn in the Fitts' GUI and the target order is completely independent of the ones in the actual test session.
5	Controlled direction mode. It allows the cursor to move with only the selected motion in (2) and with the gain in (1). The purpose of this mode is to fine tune the velocity control of each motion.
6	Prediction mode. This is the final tweak of the cursor tuning. It predicts the motion done by the user with all the tuned gains. It allows the subject to get a overall feel of cursor control and decide, if it needs more tuning or not.

Table A.3: Item numbers and a short description for the FTAT test tuning GUI.

Item No.	Description
1	Filtering options ranges.
2	Choose method to display.
3	Displays the names of the currently filtered results.
4	Print the global confusion matrix and displays it has a bar graph in (9), for all items filtered (i.e., the items in the list (3).
5	Print the selected item info in the list (3) and displays its confusion matrix in (9).
6	Filter the folder selected in (10) with the parameters in (1) and (2).
7	Print desired info about the selected axis. This can be displayed (in 9) as 2D data or 3D data. The options for each axis include window size, window step, m steps, classification accuracy and testing accuracy.
8	Create a new figure with the data of the axis (9). This can be useful to extract plots.
9	Axis in which data can be plotted.
10	Select a folder for analysis.

Table A.4: Item numbers and a short description for the data analysis GUI.

Appendix B

Detailed Subject Test Results

DOF	Reaction (s)	Time Taken (s)	Overshoots (%)	Target Ring	Status
1 DOF					
	1.50	5.21	0	Inner	Succeeded
	1.15	6.89	0	Inner	Succeeded
	1.14	12.82	0	Inner	Succeeded
	1.27	16.27	1	Inner	Succeeded
	1.03	12.82	0	Outer 1	Succeeded
	1.13	22.59	0	Outer 1	Succeeded
	1.15	15.39	0	Outer 2	Succeeded
	1.99	14.19	0	Outer 2	Succeeded
Metrics					
	1.3 ± 0.31	13.27 ± 5.44	12.5	-	100
2 DOF					
	0.13	8.11	0	Inner	Succeeded
	1.16	11.87	0	Inner	Succeeded
	1.13	10.47	0	Inner	Succeeded
	1.26	10.04	0	Inner	Succeeded
	1.09	17.81	1	Outer 1	Succeeded
	1.12	13.42	0	Outer 1	Succeeded
	1.13	20.19	0	Outer 2	Succeeded
	1.15	20.21	1	Outer 2	Succeeded
Metrics					
	1.02 ± 0.36	14.02 ± 4.77	25	-	100

Table B.1: Performance metrics for subject A using SVM.

DOF	Reaction (s)	Time Taken (s)	Overshoots (%)	Target Ring	Status
1 DOF					
	1.14	5.07	0	Inner	Succeeded
	1.03	13.92	0	Inner	Succeeded
	1.01	4.83	0	Inner	Succeeded
	1.03	5.68	0	Inner	Succeeded
	1.05	17.20	1	Outer 1	Succeeded
	1.45	9.02	1	Outer 1	Succeeded
	1.12	19.61	1	Outer 2	Succeeded
	1.09	5.42	0	Outer 2	Succeeded
Metrics					
	1.12 ± 0.14	10.09 ± 5.99	37.5	-	100
2 DOF					
	1.65	7.46	0	Inner	Succeeded
	1.26	9.28	0	Inner	Succeeded
	1.27	14.42	0	Inner	Succeeded
	1.44	11.89	0	Inner	Succeeded
	0.14	14.22	0	Outer 1	Succeeded
	1.16	13.60	0	Outer 1	Succeeded
	1.15	10.62	0	Outer 2	Succeeded
	1.28	11.42	0	Outer 2	Succeeded
Metrics					
	1.17 ± 0.45	11.61 ± 2.46	0	-	100

Table B.2: Performance metrics for subject B using SVM.

DOF	Reaction (s)	Time Taken (s)	Overshoots (%)	Target Ring	Status
1 DOF					
	1.69	24.04	2	Inner	Failed
	1.02	15.14	0	Inner	Failed
	1.14	30.06	3	Inner	Failed
	2.13	22.30	0	Inner	Succeeded
	1.11	30.17	1	Outer 1	Failed
	1.25	8.43	1	Outer 1	Succeeded
	0.83	30.20	0	Outer 2	Failed
	1.16	30.19	0	Outer 2	Failed
Metrics					
	1.29 ± 0.42	23.82 ± 8.24	87.5	-	25
2 DOF					
	1.15	16.52	0	Inner	Failed
	1.03	10.70	2	Inner	Succeeded
	1.85	26.00	5	Inner	Failed
	1.10	19.42	1	Inner	Succeeded
	1.13	9.99	0	Outer 1	Failed
	0.13	26.40	2	Outer 1	Succeeded
	1.40	30.02	0	Outer 2	Failed
	1.01	18.6	0	Outer 2	Failed
Metrics					
	1.1 ± 0.48	19.71 ± 7.35	125	-	37.5

Table B.3: Performance metrics for subject C using SVM.

DOF	Reaction (s)	Time Taken (s)	Overshoots (%)	Target Ring	Status
1 DOF					
	1.27	30.05	0	Inner	Failed
	1.39	6.49	1	Inner	Succeeded
	2.07	23.69	3	Inner	Succeeded
	1.70	6.27	0	inner	Succeeded
	0.13	13.99	0	Outer 1	Succeeded
	1.82	30.03	2	Outer 1	Failed
	1.40	19.20	0	Outer 2	Failed
	1.99	30.03	0	Outer 2	Failed
Metrics					
	1.47 ± 0.62	19.97 ± 10.17	75	-	50
2 DOF					
	1.73	30.02	0	Inner	Failed
	1.47	26.07	0	Inner	Succeeded
	1.64	11.89	0	Inner	Succeeded
	1.46	8.91	0	Inner	Succeeded
	0.39	24.99	0	Outer 1	Succeeded
	1.83	24.60	0	Outer 1	Succeeded
	1.73	18.81	0	Outer 2	Succeeded
	1.31	30.03	0	Outer 2	Failed
Metrics					
	1.45 ± 0.46	21.92 ± 7.97	0	-	75

Table B.4: Performance metrics for subject D using SVM.

DOF	Reaction (s)	Time Taken (s)	Overshoots (%)	Target Ring	Status
1 DOF					
	1.21	9.84	0	Inner	Succeeded
	1.35	4.45	0	Inner	Succeeded
	1.2	13.41	0	Inner	Succeeded
	1.13	4.2	0	Inner	Succeeded
	1.44	10.82	0	Outer 1	Succeeded
	0.97	8.02	1	Outer 1	Succeeded
	1.20	6.6	0	Outer 2	Succeeded
	1.61	10.61	0	Outer 2	Succeeded
Metrics					
	1.26 ± 0.2	8.49 ± 3.26	12.5	-	
2 DOF					
	1.49	8.42	0	Inner	Succeeded
	0.12	12.83	0	Inner	Succeeded
	0.12	27.67	0	Inner	Succeeded
	1.19	12.7	0	Inner	Succeeded
	1.13	19.20	0	Outer 1	Succeeded
	1.12	21.99	1	Outer 1	Succeeded
	1.10	13.64	0	Outer 2	Succeeded
	1.34	13.2	0	Outer 2	Succeeded
Metrics					
	0.95 ± 0.53	16.21 ± 6.25	12.5	-	

Table B.5: Performance metrics for subject A using LDA.

DOF	Reaction (s)	Time Taken (s)	Overshoots (%)	Target Ring	Status
1 DOF					
	1.09	11.27	2	Inner	Succeeded
	0.24	11.92	0	Inner	Succeeded
	0.14	9.22	1	Inner	Succeeded
	1.00	9.67	0	Inner	Succeeded
	0.98	14.59	0	Outer 1	Succeeded
	0.13	15.41	0	Outer 1	Succeeded
	0.12	18.41	0	Outer 2	Succeeded
	1.03	18.19	0	Outer 2	Succeeded
Metrics					
	0.59 ± 0.47	13.59 ± 3.61	37.5	-	100
2 DOF					
	1.82	19.44	0	Inner	Succeeded
	1.01	13.28	3	Inner	Succeeded
	0.60	15.32	5	Inner	Failed
	1.83	11.65	0	Inner	Succeeded
	0.14	14.79	0	Outer 1	Succeeded
	0.73	30.00	1	Outer 1	Failed
	0.93	17.79	0	Outer 2	Succeeded
	1.09	22.58	1	Outer 2	Succeeded
Metrics					
	1.02 ± 0.58	18.11 ± 5.94	125	-	75

Table B.6: Performance metrics for subject B using LDA.

DOF	Reaction (s)	Time Taken (s)	Overshoots (%)	Target Ring	Status
1 DOF					
	1.68	5.21	0	Inner	Succeeded
	0.12	9.63	1	Inner	Succeeded
	1.64	5.84	0	Inner	Succeeded
	0.12	24.64	0	Inner	Succeeded
	1.16	9.21	0	Outer 1	Succeeded
	1.59	7.99	0	Outer 1	Succeeded
	1.24	4.62	0	Outer 2	Failed
	0.12	10.80	0	Outer 2	Succeeded
Metrics					
	0.96 ± 0.72	9.74 ± 6.42	12.5	-	87.5
2 DOF					
	1.05	13.67	0	Inner	Succeeded
	0.13	22.84	1	Inner	Succeeded
	1.64	4.64	0	Inner	Succeeded
	1.62	6.01	0	Inner	Succeeded
	1.16	11.64	0	Outer 1	Succeeded
	1.15	6.80	0	Outer 1	Succeeded
	1.59	8.60	0	Outer 2	Failed
	1.28	17.81	1	Outer 2	Succeeded
Metrics					
	1.20 ± 0.49	11.5 ± 6.33	25	-	87.5

Table B.7: Performance metrics for subject C using LDA.

DOF	Reaction (s)	Time Taken (s)	Overshoots (%)	Target Ring	Status
1 DOF					
	1.46	15.44	3	Inner	Succeeded
	1.19	23.20	0	Inner	Succeeded
	2.7	13.28	2	Inner	Succeeded
	0.95	29.54	0	Inner	Succeeded
	0.12	30.00	0	Outer 1	Failed
	0.12	10.60	0	Outer 1	Succeeded
	1.40	12.19	0	Outer 2	Succeeded
	0.12	20.00	0	Outer 2	Succeeded
Metrics					
	1.01 ± 0.90	19.28 ± 7.68	62.5	-	87.5
2 DOF					
	0.33	15.70	1	Inner	Succeeded
	0.22	16.09	0	Inner	Succeeded
	0.37	11.95	0	Inner	Succeeded
	1.3	20.20	0	Inner	Succeeded
	0.12	26.8	1	Outer 1	Succeeded
	1.14	12.20	0	Outer 1	Succeeded
	0.12	18.79	0	Outer 2	Succeeded
	0.48	18.41	0	Outer 2	Succeeded
Metrics					
	0.51 ± 0.45	17.51 ± 4.79	25	-	100

Table B.8: Performance metrics for subject D using LDA.