CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF ELECTRICAL ENGINEERING

DIPLOMA THESIS

Ing. Néstor Mauricio Caro Sánchez

# Gesture classification based on electromyography

**Department of Cybernetics**

Project supervisor: **Ing. Petr Posik, Ph.D.**

Prague

May 2016

**Czech Technical University in Prague**
**Faculty of Electrical Engineering**

**Department of Cybernetics**

# DIPLOMA THESIS ASSIGNMENT

| | |
|---|---|
| **Student:** | Néstor C a r o |
| **Study programme:** | Cybernetics and Robotics |
| **Specialisation**: | Robotics |
| **Title of Diploma Thesis:** | Gesture Classification Based on Electromyography |

**Guidelines:**

The purpose of the work is to identify hand gestures based in the Electromyography raw signals provided from the Myo armband. First replicating the gesture dictionary provided by the manufacturer and then expanding the amount of recognizable gestures through various feature extraction and pattern recognition techniques.

1. Learn the principles of electromyography with relation to the Myo armband device.
2. Study relevant methods of signal processing, and machine learning.
3. Use the device SDK to create a training data set of signals and their correct classification.
4. Create and evaluate a classifier reproducing the behavior of the software provided with the device.
5. Expand the vocabulary of gestures, collect data for them and label them manually.
6. Train and evaluate a new classifier for the expanded gesture set.

**Bibliography/Sources:**

[1] Xing, K., Yang, P., Huang, J., Wang, Y., & Zhu, Q. (2014). A real-time EMG pattern recognition method for virtual myoelectric hand control. Neurocomputing, 136, 345-355.
[2] Pan, L., Zhang, D., Liu, J., Sheng, X., & Zhu, X. (2014). Continuous estimation of finger joint angles under different static wrist motions from surface EMG signals. Biomedical Signal Processing and Control, 14, 265-271.
[3] Boyali, A., & Hashimoto, N. (2016). Spectral Collaborative Representation based Classification for hand gestures recognition on electromyography signals. Biomedical Signal Processing and Control, 24, 11-18.

**Diploma Thesis Supervisor:** Ing. Petr Pošík, Ph.D.

**Valid until:** the end of the summer semester of academic year 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
**Head of Department**

prof. Ing. Pavel Ripka, CSc.
**Dean**

Prague, January 5, 2016

*Abstract*

The purpose of the work is to identify different hand poses based in the Electromyography raw signals provided from a Myo armband, using various signal processing, feature extraction and pattern recognition techniques. First we will replicate the gesture dictionary provided by the manufacturer, and then we will explore different hand poses that are compatible with the basic dictionary of gestures, and then expanding the amount of recognizable gestures for the tested classifiers.

# Table of Contents

## Table of Figures

## List of Abbreviations

APL – Applied Physics Laboratory

BCI – Brain Computer Interfaces

CNS – Central Nervous System

CTU – Czech Technical University

DARPA – Defense Advanced Research Projects Agency

EMG – Electromyography

EEG – Electro Encephalography

MUP – Motor Unit Potential

PCA – Principal Component Analysis

PLP – Phantom Limb Pain

PNS – Peripheral Nervous System

SDK – Software Development Kit

# 1 Introduction

In this document, we will learn the principles of *Electromyography* (EMG) for identifying a set of hand gestures, using various feature extraction and pattern recognition techniques. The relevancy of such project lies in its possible application as the input flow of information for a prosthetic device using EMG and machine learning, for *decoding* the patient's movement intention into a dictionary of hand poses. A special emphasis has to be done in the word *decoding*, since using the same nervous pathways used by a limb prior the amputation for the control of a prosthesis, reduce the potentially negative effects of aberrant neuroplasticity and, in consequence a treatment for *Phantom Limb Pain* (PLP).

Neuroplasticity is the capacity of the brain, *Central Nervous System* (CNS) and *Peripheral Nervous System* (PNS) to reconfigure after a traumatic event in order to compensate for the amputation, by improving the somatosensory representation and motor system reaction. However, the physical reconfiguration of nervous tissues, can sometimes lead to the aberrant growing of neuromas (Tumor growth of a nerve), which is one of the main causes of PLP.

This text is organized in the following way. In chapter 2 we will introduce the reader to related works involving the use of EMG, machine learning, cybernetic prosthesis and the Myo armband itself, in order to provide some state of the art and proof of the validity of this work.

In chapter 3, we will explain some physiological principles for understanding the problem this work attempts to treat, first explaining some neurobiological principles behind bidirectional interfaces of cybernetic hand prostheses, allowing the reader to understand the context and purpose of this thesis work. Here we will explain, how the nerve tissues transform through time after an amputation, its implications, and how the use of certain types of prosthetics can reduce the problematic PLP. Later, we will explain the principles of EMG relevant for the development of our solution. Where we will see how a thought is translated into a mechanical movement by our body, the mechanisms and processes implied. And a description of the methods used to measure, record and process the action potentials generated by the contraction of muscles. Also, some mathematical manipulation to extract additional features from the raw signals acquired.

In chapter 4, we will show an overview of the final layout of the system developed for satisfying our objectives, followed by a detailed description of each phase. In this section we will show in detail the hardware and software used and developed for the purpose of the thesis. With the use of block diagrams, schematics and references to the codes developed, we will illustrate how the Myo armband was used, the interface to Matlab, the data preprocessing and acquisition protocols, also the principles and functions of the different types of classifiers used for the machine learning part of this work. Also, we will provide a description of the methodology followed for each experiment performed, organized by relevancy and context.

In chapter 5, we will find the experimental results of all the experiments done, using different acquisition frequencies for the EMG signals, different methods, approaches and classifiers. Also showing problems and solutions that grew during the experimentation and tuning of the different classifiers, and part of the empirical knowledge gained during this process.

In chapter 6, we will see the conclusions harvested from the whole experience and, we will propose possible future works using the results of this thesis as a starting point, towards cybernetic prosthesis capable of smooth adaptation to the patients, with reduction of aberrant neuroplasticity and PLP. And finally, in the appendixes, the reader will find useful information of the devices engineered for solving the issues risen in the development of this work.

# 2 Problem description and related work

Prosthetics have evolved drastically since the early wooden limbs of XV century, to the amazing pieces of Bio Mechatronic engineering we see today, capable of almost human like movements, replicating their precision and dexterity.

However, aside from the cybernetic and mechatronic development, the bottleneck of prosthetics in the scientific and industrial community has been the link between man and machine. Progressively seeking the ultimate cyborg interface, a bidirectional link allowing both control of the prosthesis and feedback from it.

In order to develop cybernetic prostheses with interfaces to the nervous system, there needs to be a classifier estimating the meaning of the biological information coming from the patient. However, each case of amputation and patient represent unique physiognomies, thus for increasing the proportion of correct classification, the designers and developers of prosthesis must have a broad interdisciplinary knowledge, including a deep understanding in neurobiology: from post-amputation plastic reorganization characteristics and mechanisms, to different nervous system levels [1].

In addition, prosthetics designers should aim for functional restoration of neural pathways, supporting the reorganization of the brain, by properly guiding neuroplasticity in a convenient manner to bring back to life the same neural pathways used by the organism prior the amputation, and convert them to targets for direct neural interfaces. This can be achieved by artificial sensory feedback and creating prosthesis that decode the motor intention of the user, using machine learning. The latter is the exact problem we want to solve with this work. However, this thesis knows its limitations and won’t go as far as creating a prosthesis, so we will focus in decoding the motor intention of the user, using EMG and machine learning. Thus preparing the road for a low cost prosthesis capable to deal with this problem, hopefully a future master or PhD thesis of the CTU.

Along this section we will describe relevant technological developments for cybernetic prosthesis, organized in different areas, including low cost approaches and the state of the art project developed by the Defense Advanced Research Projects Agency (DARPA) of US, which currently is revolutionizing prosthetics by integrating all the right components together.

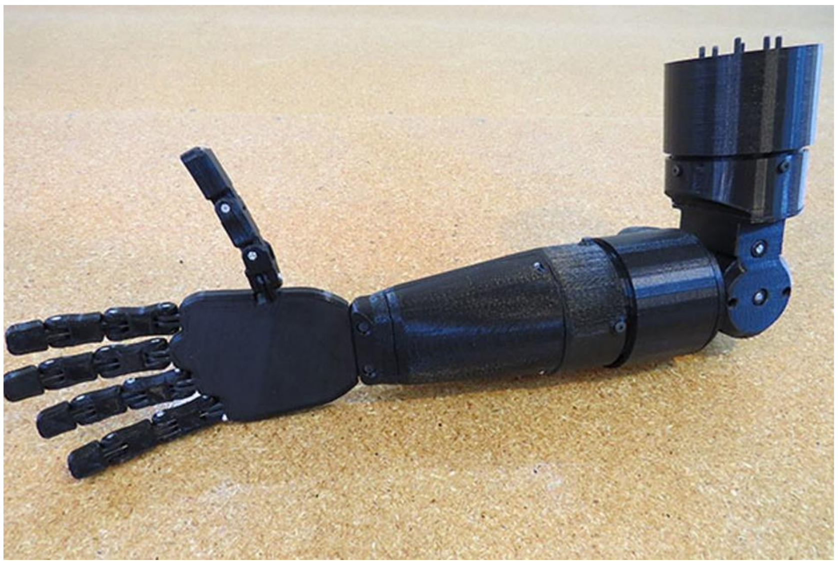
## 2.1 Economic prosthesis, enhanced evolution.

### 2.1.1 Mechatronic Achievements

Until not long ago, the price for acquiring a prosthetic device grew exponentially according its sophistication. Allowing the existence of a multi-billionaire industry, feeding on the sorrow and limitations of many by maintaining ridiculously high prices, focusing on very wealthy clients and organizations.

Recently, the fast evolution of 3D printing and its decreasing prices, summed with visionary and altruist efforts of scientists like *Easton LaChappelle* that at the young age of 17, gave a hard strike to the old prosthetic industry, by open sourcing the model of a prosthetic 3D printable hand (figure 1). Turning a product usually found in the market around the price of $80.000 US dollars, into a device able to be home produced around $350 US dollars.

Figure 1 – 3D printable hand



**Source:** http://theroboarm.com/

Whether this is not the most sophisticated prosthesis of this time, it must be highlighted how this kind of initiatives had encouraged the involvement of many new institutions, scientists, companies and hobbyist, resulting in a fast evolving industry.

### 2.1.2 Different interfaces

There are 5 main approaches to classify prosthesis according to their type of interaction with the patient [1]:

- Mechanical Interfaces: Powered by the body or the stump, which are usually limited to one or two degrees of freedom.
- Myoelectric interfaces using non-homologous muscles: Which are based in EMG, from both residual muscles in the amputated area, and non-homologous muscles, where the user is forced to perform unnatural expressions of movement in order to move the prosthesis.
- Myoelectric interfaces using homologous muscles: Which is only possible when the muscles used to move the amputated area are still available, which is perceived by the user in a more natural and user-friendly way.
- Non-invasive neural interfaces: Such as superficial EEG controlled devices, which deal with the major problem of noise.
- Invasive neural interfaces: Mostly known as BCI. They require the insertion of electrodes directly in the cortex or the nerves, which solves the noise problem from the previous kind of prosthesis but requires a surgical intervention prior to use.

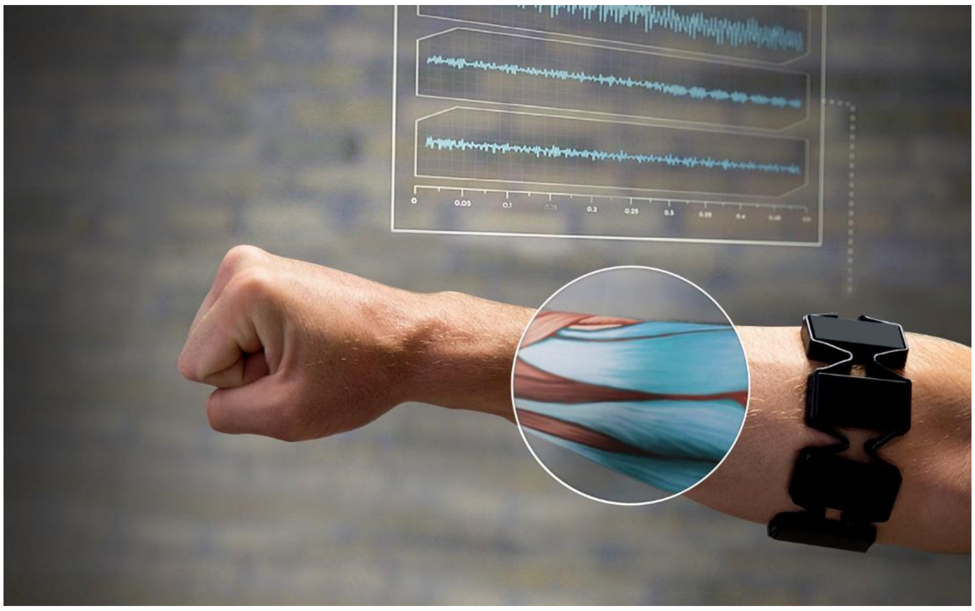
In addition, recently an invasive approach called targeted reinnervation was developed, which transfers sensor and motor fibers of residual truncated nerves from the stump into another area where the EMG is easier to record. Inducing surgically a myoelectric interface using homologous muscles, even if it's not anatomically possible.

For the sake of space and relevancy, we will only deepen the myoelectric interfaces using homologous muscles, since it is the approach chosen for this work. Now, with a cheap and accurate effector just as the one shown in the previous section, a robust and equally cheap signal acquisition device and control input are required to exploit the full potential of this economic efforts.

### 2.1.3 Cheaper interfaces

In a similar way to the prosthetic development, until not long ago a reliable EMG acquisition device was both costly and bulky, and given the stochastic nature of this method, several filtering techniques were needed to be able to use EMG in any practical application. However, in 2013 Thalmic Labs, a Canadian company, born from a crowd funded startup released to the market the Myo armband (figure 2), a dry EMG acquisition device originally intended to be used as a computer gadget. Allowing the user to scroll, control presentations, etc. Nevertheless, the company was aware of the great potential behind a cheap and robust EMG acquisition device, thus providing full access to the device software.

**Figure 2 – Myo armband, graphical concept**



**Source:** developerblog.myo.com/raw-uncut-drops-today/

From its release to the date, hundreds of different developments based on this device have been reported and open sourced for the growing community of developers and users. Details of this device will be exhaustively explained in the fourth part of this thesis work.

## 2.2 DARPA's Revolutionizing Prosthetics challenge

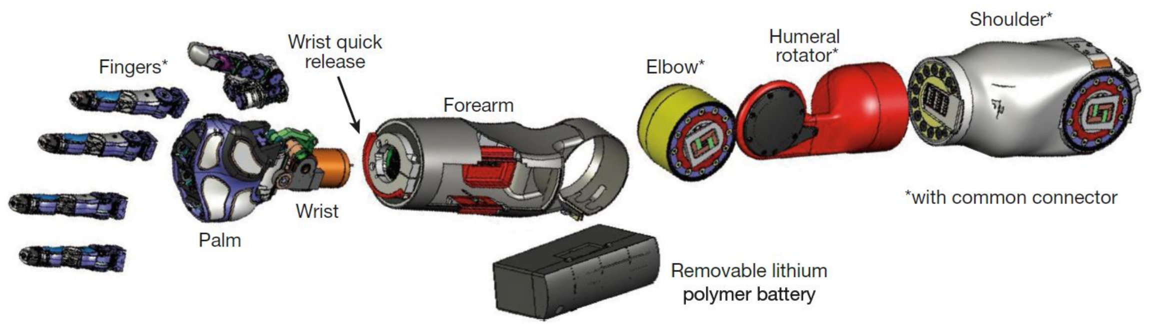
### 2.2.1 Accepting the challenge

From 2005 to 2009 DARPA took the challenge to develop the world's most advanced prosthetic limb; having strength, sensation, weight, comfort and appearance of a native human limb, and in addition it had to be controlled using the patient's mind. They succeeded by having an interdisciplinary team of more than 100 engineers, scientists, researchers and clinicians from academic, government and industrial institutions [2].

Each of the contributors was subdivided in several areas of expertise. Engineering specialties included systems, electrical, mechanical, wireless communications, power-sensitive applications, human factors, cosmesis, reliability, manufacturability, and project and program management. Scientific specialties were neuroscience, sensory feedback and haptics, neural motor decoding, neural stimulation, and research studies. Clinical specialties were surgery, clinical/research prosthetics, physical therapy, occupational therapy and human subject research.

The final product was a modular prosthesis (figure 3), able to adapt to numerous different kinds of amputations, and allowing the easy replacement of small parts in case of malfunction, or upgrade. Rather than depending in the prosthesis as a whole.

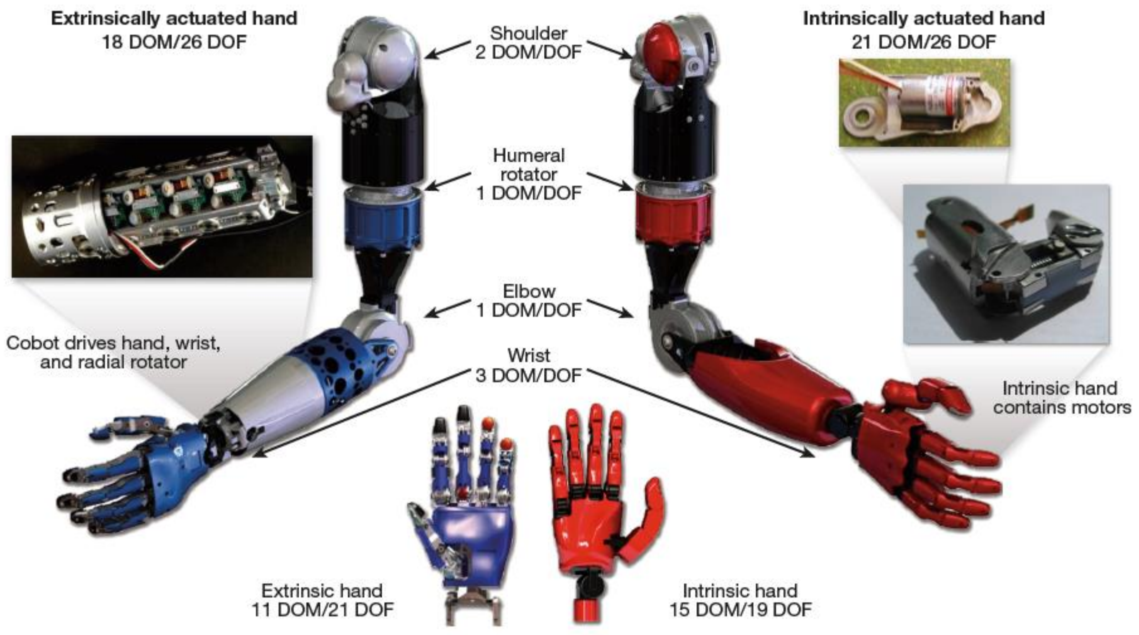
**Figure 3 – DARPA modular prosthesis in pieces**



**Source:** [2]

Additionally, since the motors for the finger flexion were located in the forearm, DARPA developed a variation of the prosthesis to fit transradial amputees, which by definition still have their forearm. This prototype would have the motors built in the palm of the hand (figure 4).

Figure 4 – DARPA prosthetic arm, extrinsic hand vs. intrinsic hand



Source: [2]

For controlling this devices, DARPA originally explored different kinds of custom-designed neural interfaces, ranging from electrode arrays surgically implanted both in the brain and in muscles and nervous tissue to micro coils for wireless power and data transmission.

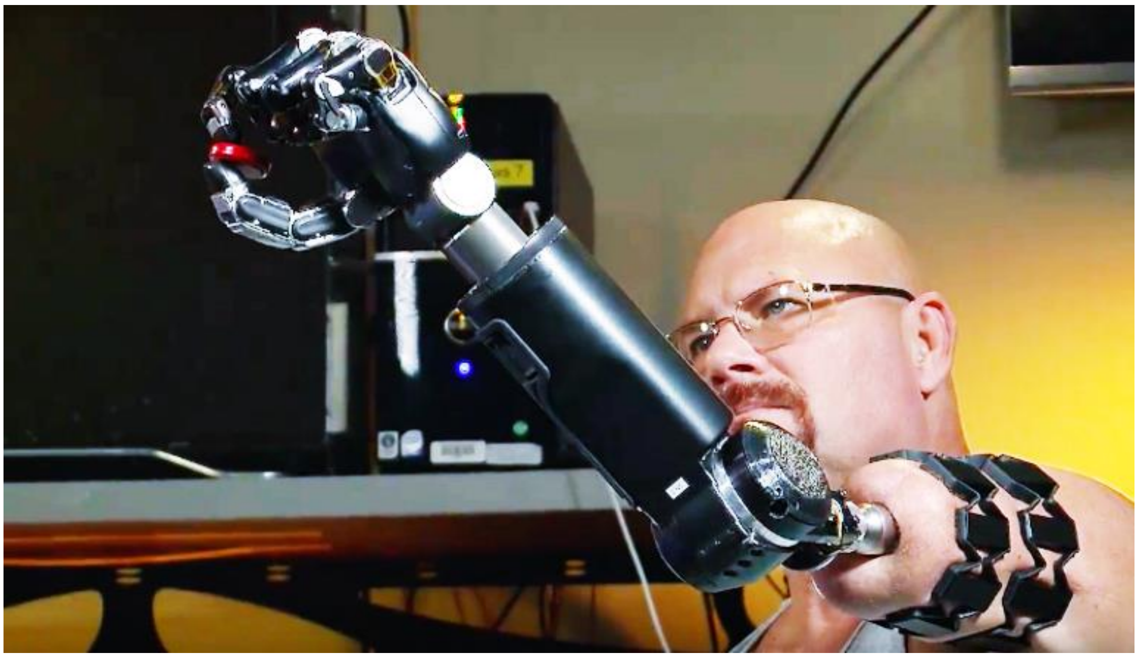
### 2.2.2 Taking the challenge far beyond, feedback and enhanced interfaces

After the unanimous success of the prototypes developed by the Revolutionizing Prosthetics challenge, DARPA continued investing in refining the prosthesis for 3 years until 2012, when trials on human patients started. 35 volunteer amputees participated for the testing and provided design feedback for the development of the Gen-3 Arm System by DEKA Integrated Solutions Corporation, one of two primary performers on the Revolutionizing Prosthetics program [3].

In 2015 DARPA claimed to have closed the circuit for bidirectional communication by providing sensory feedback to a man who has been paralyzed for more than a decade as a result of a spinal cord injury. They solved this puzzle with an invasive solution, placing electrode arrays in the sensory cortex of the patient (*brain region responsible for identifying tactile sensations such as pressure*). Then they linked to a hand prosthesis developed by the Applied Physics Laboratory (APL) at Johns Hopkins University, which contains torque sensors that can detect when pressure is being applied to any of its fingers, and can convert them into electrical signals that would be interpreted as sensations by the sensory cortex [4].

At the end of the same year and beginning 2016, the APL released footage and scarce information about their latest achievement, which for the surprise of the author was using 2 Myo armbands. Their patient was subject to various surgeries, procedures and extensive training sessions to allow the intuitive control of the latest DARPA prosthesis with more than 22 DOF, having the amputation above the elbow. As shown in the following figure.

**Figure 5 – DARPA prosthetic arm, powered with 2 Myo armbands**



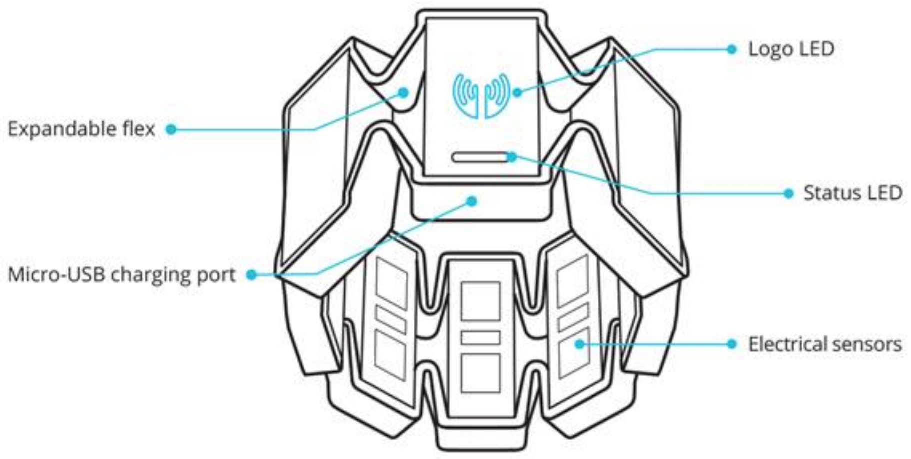
**Source:** http://www.jhuapl.edu/newscenter/pressreleases/2016/160112.asp

Among the surgeries done to the patient were: osseointegration, which consisted in attaching a mount for the prosthesis device directly to his bone, which solves one of the main issues of wearing a prosthesis; comfort. The second surgery was the re innervation procedure mentioned earlier in this section, deploying several nerve and muscular tissues in his residual upper arm, making the amount of EMG data acquired by the Myo armbands to be of substantial importance, since they were carrying more information than a normal arm [5].

## 2.3 Myo armband

Myo is a device designed for detecting a set of poses and events, based on EMG readings and machine learning algorithms for classifying the muscular activity into a set of predefined poses. **It acquires the raw EMG signals in real time**, using dry platinum electrodes. Myo has 8 pods, each acquiring independent signals along a bracelet (Figure 6) and streaming the acquired data through the central pod (the one with the logo) using Bluetooth communication, to a computer. Then, the software from Thalmic Labs will process the signals to detect and determine the arm wearing the device, the orientation of the Myo and a set of fixed poses, also allowing intuitive linking to other apps or functions within the pc, whether using their application, or writing simple codes in *lua* programming language.

**Figure 6 – Myo Armband, hardware overview**



**Source:** https://developer.thalmic.com/docs/api_reference/platform/getting-started.html

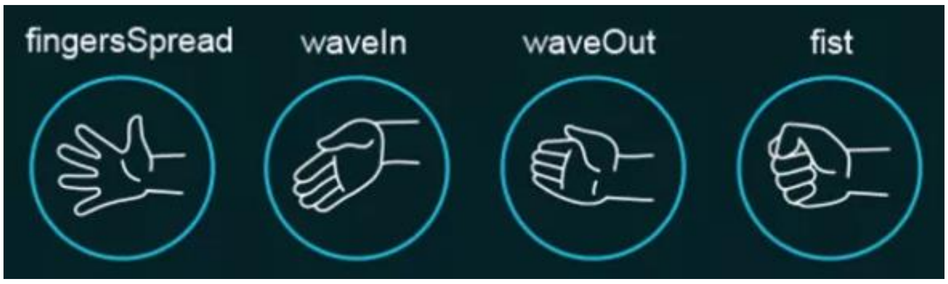
Myo armband was created out of a crowd funding back in 2013, and it was originally intended to be used as an external controller for computers. Allowing the user to control the mouse, scroll a page, control presentations, etc. However, the founders of the company were aware of the great potential behind a cheap and robust EMG acquisition device, but they were also aware of their limitations as a newborn company. Therefore, they released an SDK allowing full access to the hardware of the Myo, leading to a deep interaction with the device for any kind of application.

Since the release of the SDK, hundreds of developers have used their creativity to control and interface with a broad kind of devices, software and platforms. Revolutionizing the EMG application possibilities forever.

### 2.3.1 Myo Armband Manager

This application is the bridge between the device and the computer, since the Myo itself is only a data acquisition device. The hand gesture classification is done by the manager, using the computer's resources, and the identified hand pose is displayed in the right lower corner of the screen with the symbols seen in the following figure.

Figure 7 – Myo Armband's, hand pose dictionary



**Source:** http://theburningmonk.com/2014/10/myo-first-day-of-happy-hacking/

Additionally, the Myo can be manually calibrated from this program, in order to enhance the sensitivity of the classifier. Any application developed using Myo must use the Armband Manager as an intermediate step for gathering the data coming from the device.

## 2.4 The magic behind the robot, Machine Learning

Machine learning is one of the few common techniques seen among all the published approaches for cybernetic prosthesis, given its power to classify and learn different behaviors or clusters of information. However, given the broad amount of methods available for machine learning, the following table was extracted from a review article summarizing the results of many researches aimed to the analysis of EMG classification in the domain of neuromuscular pathology [6], the table was further improved by the addition of results involving ***machine learning approaches for movement and pose pattern recognition***, easily identifiable by the ***bold italic font***. In the table, Ns is number of subjects and Nm is number of measurements.

| *Year* | *Technique* | *Ns(Nm)* | *Accuracy (%)* |
|---|---|---|---|
| | *Artificial neural Networks* | | |
| *1992* | *Self-organizing feature map (SOFM)* | *114* | *76-83* |
| *1995* | *Integration of parametric pattern recognition algorithm (PPR) and artificial neural network (ANN)* | *44* | *80-90* |
| *1996* | *SOFM and learning vector quantization (LVQ)* | *50* | *60-80* |
| *1998* | *Modular ANN* | *40(800)* | *79.6* |
| *1999* | *SOFM, LVQ and statistical methods based on Euclidean distance* | *(1213)* | *90* |
| *2004* | *Continuous wavelet transform (CWT) and a multi-channel ANN* | *13(260)* | *-* |
| *2005* | *Multi-layer perceptron (MLP)* | *12* | *91.6* |
| *2006* | *Wavelet-based neural network (WNN)* | *(1200)* | *90.7* |
| *2007* | *Radial basis networks (RBN) and decision trees* | *62(365)* | *89* |

| | | | |
|---|---|---|---|
| 2012 | *SOFM and LVQ* | 11 | 97.6 |
| 2013 | *Principal component analysis (PCA) and probabilistic neural network (PNN)* | 12 | 68-94.3 |
| | *Fuzzy* | | |
| 2001 | *Fuzzy logic* | (29) | 88.4 |
| 2006 | *Adaptive fuzzy k-NN classifier (AFNNC)* | (11) | 96.6 |
| 2012 | *Fuzzy logic* | 97 | 97 |
| | *ANN hybrids* | | |
| 1996 | *Combined ANN and genetics-based machine learning (GBML) models* | 34 | 80 |
| 2004 | *Fuzzy integral of multiple ANN* | 80 | 88.58 |
| 2010 | *Neuro-fuzzy system (NFS)* | 177 | 90 |
| | *Support Vector Machines* | | |
| 2002 | Support vector machine (SVM) with one against one training algorithm | (231) | 89 |
| 2005 | *SVM* | 59 | 92.3 |
| 2009 | Multiclass SVM | 12 | 100 |
| 2010 | Binary SVM | 12 | 100 |
| 2010 | Fuzzy support vector machine (FSVM) | 12 | 99.6 |
| 2012 | SVM with wavelet technique for feature extraction | 300 | 99.4 |
| 2012 | *FSVM classifier combined with statistical features extracted From discrete wave transform (DWT)* | 27 | 97.67 |
| 2013 | Hybridization of the particle swarm optimization (PSO) and SVM | 27 | 96.75 |
| 2014 | ***SVM (1vs1-1vsAll) with wavelet packet transform (WPT) [27]*** | - | 98.21-98.39 |
| | *Others* | | |
| 1995 | Principle component analysis and multivariate discriminant algorithm | 302(6828) | 70.4-76.5 |
| 2008 | Bayesian aggregation | 57 | 88.7 |
| 2012 | Decision tree | 27 | 96.33-96.50 |
| 2014 | ***Linear Discriminant Analysis (LDA) and fourteen state-space model [7]*** | 7(56) | 93.82 |
| 2014 | ***Principal Component Analysis (PCA) vs. Common Spatial Pattern (CSP) [8]*** | 21(168) | 88.1vs89.35 |
| 2015 | ***Spectral Collaborative Representation [9]*** | - | 98.34 |

Whether neuromuscular pathology is not directly related to the field of prosthetics, it deals directly with the analysis of abnormal EMG readings, just as what you could expect from an amputee's residual muscles.

# 3 Physiological theoretical frame

## 3.1 Neurobiological principles

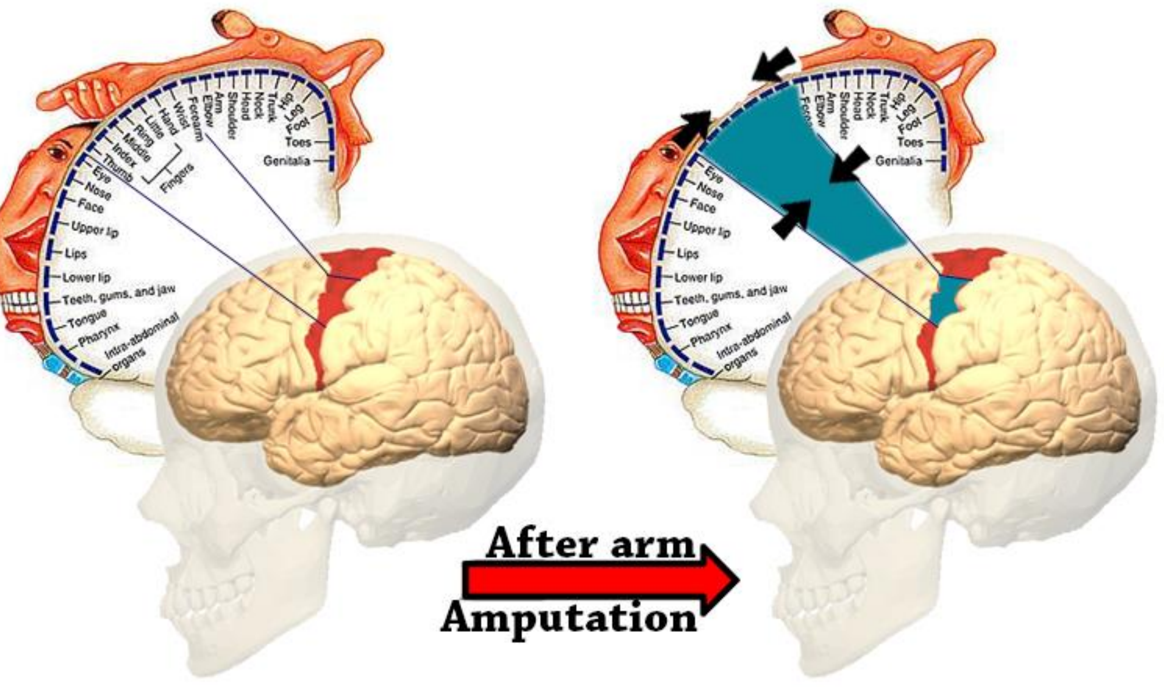
We will introduce the reader to some neurobiological principles behind bidirectional interfaces of cybernetic hand prostheses, in order to give context and purpose to this thesis work. Showing the transformations in the body after an amputation occurs and how prosthetics can influence such transformation.

After an amputation, the nervous system experience changes derived from the sensory deprivation of the missing limb, due to the well-studied plasticity of the brain and the nervous system. Neural rearrangement following a limb amputation occurs as a form of compensation from the sensory deprivation, making the surrounding areas in the brain more active and lowering their electrical threshold response. In general, neuroplastic changes lead to improvements of the somatosensory representation and motor system reaction, in an attempt to compensate for the amputation.

The cortical reorganization following the amputation occurs in 3 consecutive stages. The first one, happens short after the amputation, unmasking existing neural connections that were silent, in a form of contingency plan against the abrupt body transformation. The second stage happens weeks to months after the amputation, in which cortical areas deprived from peripheral input reorganize in order to express new receptive fields from the adjacent or residual cortical areas, through synaptogenesis and dendritic arborization of neurons. Finally, the third stage is the stabilization of the new body configuration in the cortex that produce more stable and sharpened receptive fields.

However, it has been proven the physical reconfiguration of the brain, CNS and PNS, can sometimes be good and sometimes can lead to potentially negative effects of aberrant neuroplasticity. With the growing of neuromas in the stump, bombarding the nervous system with erratic stimulations, and the occupation of the sensory deprived areas in the brain by adjacent body part images. As seen in figure 8, the Penfield Homunculus illustrates a map of the somatosensory projections in the brain for each part of the body; as the figure illustrates, after an arm amputation, brain tissue from the face and the arm remains will occupy the missing hand’s cortex.

Figure 8 – Brain somatosensory cortex, before and after amputation



**Source:** Author

This phenomena, in addition to neuroma formation are the main reasons many amputees develop PLP in response to the deeply traumatic experience of the amputation, as first shown by Ramachandran [10] in his fascinating neuroscience research of the brain, following his career and achievements around twenty years ago.

PLP consists in the sensory impression of activity or pain in the missing limb, after the amputation! Ranging from tickles to unbearable pain; if treated wrongly, this sensation can persist for life in some patients. However, the use of prosthesis is directly correlated to PLP diminution, depending of the frequency of use and type of the device [11]. This could be further improved by ***associating the bidirectional interface of a prosthesis with the same nervous pathways used by the original limb prior the amputation***. The most promising approaches aim at establishing a direct connection with either the central or peripheral human nervous system by means of invasive or noninvasive neural interfaces. Thus a prosthetic could be used as neurorehabilitation tool and the level of CNS reorganization can be used as parameter of effectiveness achieved by the prosthetic device and its interfaces, in restoring the hand physiological functionality.

As mentioned before, prosthetics should aim functional restoration of neural pathways by supporting the reorganization of the brain by properly guiding neuroplasticity in a convenient manner to bring back to life the same neural pathways used by the organism prior the amputation and convert them to targets for direct neural interfaces. For doing so, a stable and consistent sensory feedback must come from the prosthesis, and the device should decode the motor intention of the user, using machine learning algorithms.

For achieving the aforementioned purpose, recording of EMG, EEG or electrocorticography (ECoG) should be guided by a video-driven or mentally stimulated imagination of different hand-wrist-fingers movements, in order to define the intuitive patterns of the signals coming from motor fibers. Then, an intelligent interface should enable the interpretation of the motor commands to be executed by the prosthetic device. On the other hand, sensory feedback from the prosthesis must be introduced in an invasive way directly into the nerves, because using alternative methods of stimulation – as vibration for representing intensity of grasp – can lead to the activation of different somatosensory areas of the brain, thus leading to neural reorganization.

## 3.2 Electromyography (EMG) principles

### 3.2.1 Definition of EMG

EMG studies neuromuscular activity and muscular morphology, by measuring the action potentials generated by the contraction of muscles, which can be recorded using surface electrodes or by inserting electrodes directly into the muscle. The former is known as Surface EMG (sEMG) and the latter Intramuscular EMG. In both cases, the data is recorded using one electrode as reference, and 2 more as differential measurements.

In order to generate muscle movements, electrical signals are passed from the brain to one or more alpha-motor neurons, each one attached to hundreds or even thousands of muscle fibers, causing their contraction while the neuron(s) remains active. [1]

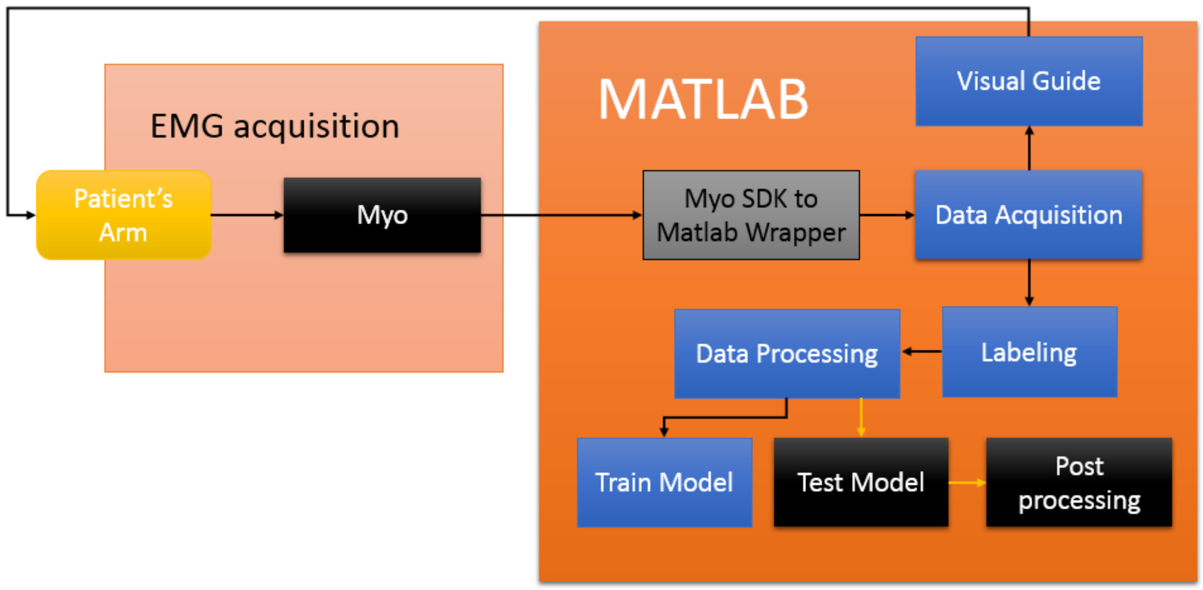
The summation of potentials produced by the alpha-motor neuron and the contraction of all the affected muscular fibers, is known as *Motor Unit Potential* (MUP), and a single muscle can contain many of them, all firing multiple times per second, giving a stochastic nature to EMG. Therefore, only acquiring the MUP is not enough for extracting useful information about the overall muscle activity, and we should define several features based on the amplitude and frequency variations of the MUP in limited time windows.

# 4 Experimental setup

As stated in previous sections of this work, our main purpose is to decode the motor intention of the user, using EMG and machine learning. For doing so, we will use a Myo armband as an EMG input device, further processing its raw information into Matlab, and then train various classifiers for allowing the recognition of a defined set of hand poses.

In the following diagram, we can appreciate an overall view of the entire data flow, mechanisms and connections involved in the development of this project.

**Figure 9 – Block diagram for the experimental setup**



**Source:** Author

## 4.1 Data acquisition

### 4.1.1 Hardware: Myo Armband

As mentioned before, Myo is a device capable of acquiring raw EMG signals from 8 different locations across a bracelet in real time. The range of the acquired signals is ±127mV and corresponds to the MUP of the measured area. In general, the Myo armband provides two kinds of data, spatial data and gestural data.

Spatial data means the orientation and movement of the user's arm, in terms of a quaternion that can be converted to other representations, like a rotation matrix or Euler angles; and an acceleration vector representing the acceleration that the Myo armband is undergoing at any given time.

Gestural data tells the application what the user is doing with their hands in the form of one of several poses, and the EMG raw data used to determine the mentioned poses and define which arm it is being worn on and which way it is oriented. Also, an application can provide feedback to the user by issuing a vibration command.

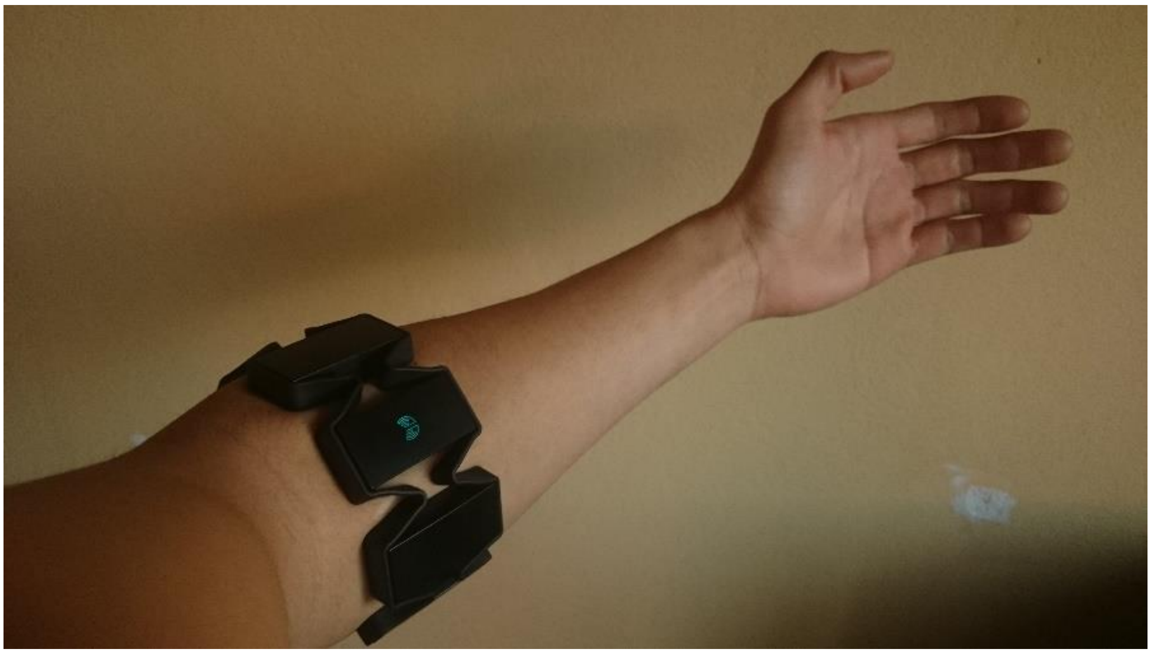
Data is sent to a computer through Bluetooth communication to an application in form of events, which identify the Myo armband sending the data and provide a timestamp of the moment the event was received. In order to access the events and data generated by the Myo, we will use the tools and codes provided by the SDK.

### 4.1.2 Myo armband placement

First we will describe some methods repeated every time we intended to acquire new data, whether it is training or testing data the acquisition protocol will be the same. It's important to mention that, the only patient to be tested was the author, and therefore discretion is advised to compare these results with other researches. Also, it should be noted that all the previous codes were modified versions of the ones provided by their respective creators. However, subsequent codes from this point forward were completely made by the author of this thesis work.

In every data acquisition session, the Myo armband was placed in a healthy left arm, trying to pay special care to place all the pods in the exact same position every time, as shown in figure 10.

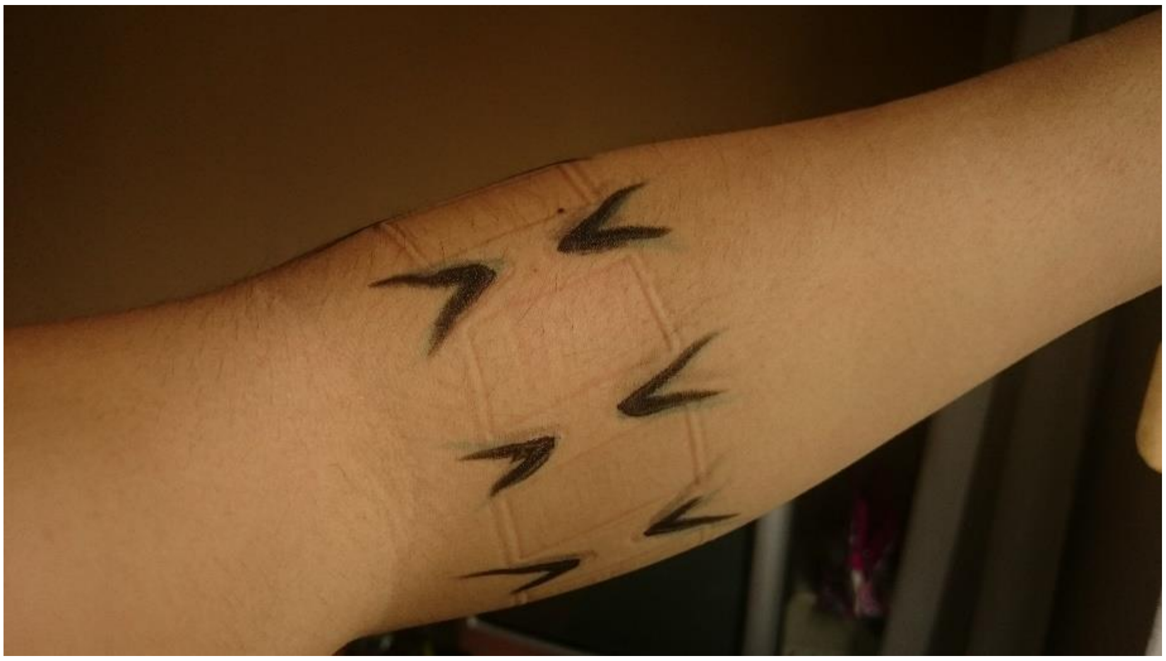
**Figure 10 – Myo armband placement**



**Source:** Author

However, despite many efforts and strategies like the one shown in the figure 11, the acquired EMG signals from each pod would change significantly every time the Myo armband was removed and worn again. If the pods were to be moved even few millimeters, this leaded to corrupted and misleading data for the classifiers. Therefore, data recorded in different sessions is unusable together, thus defining an important limitation of our work; training and testing data should be acquired in the same session.

**Figure 11 – Markers used for attempting placing the Myo in the same place**



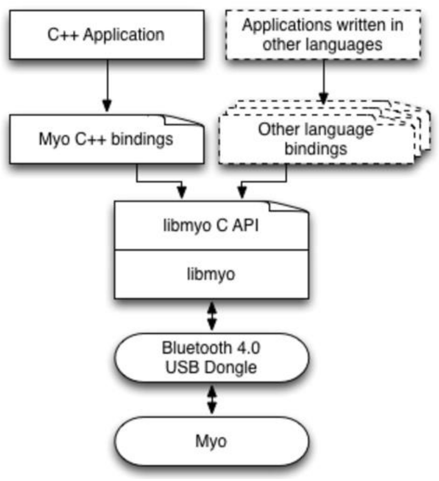
**Source:** Author

### 4.1.3 SDK

This is a compilation of binary executable, libraries, drivers, headers and documentation necessary to interact with the Myo armband; along with sample codes demonstrating the implementation of the former files, which can be used as templates, for developing a wide variety of apps.

To enable communication with the Myo in a physical level, the SDK contains a library called **libmyo** which parses the data from the Bluetooth of the device into a C API, allowing applications in various programming languages to access, both raw data, and the built in classifier of the device. As illustrated in the following diagram:

Figure 12 – SDK data flow from physical device to an application



**Source:** https://developer.thalmic.com/docs/api_reference/platform/the-sdk.html

The SDK, also provides bindings for C++. Allowing the access of the features seen in the following table, by the definition of classes [13].

| Class name | Description |
|---|---|
| myo::Hub | Used by the app to use the SDK |
| myo::Myo | Represents the Myo armband |
| myo::Pose | Poses identified by the Myo |
| myo::Quaternion | Object representing orientation |
| myo::Vector3 | Object representing acceleration vector |
| myo::DeviceListener | Receives new events |

### 4.1.4 SDK to Matlab wrapper

Using the C++ bindings, and the required libraries and headers included in the Myo SDK, we can parse the data from the spatial and gestural data from the Myo directly to Matlab for further processing. Myo Armband Manager, the software from Thalmic Labs, must be running always during the execution of subsequent codes.

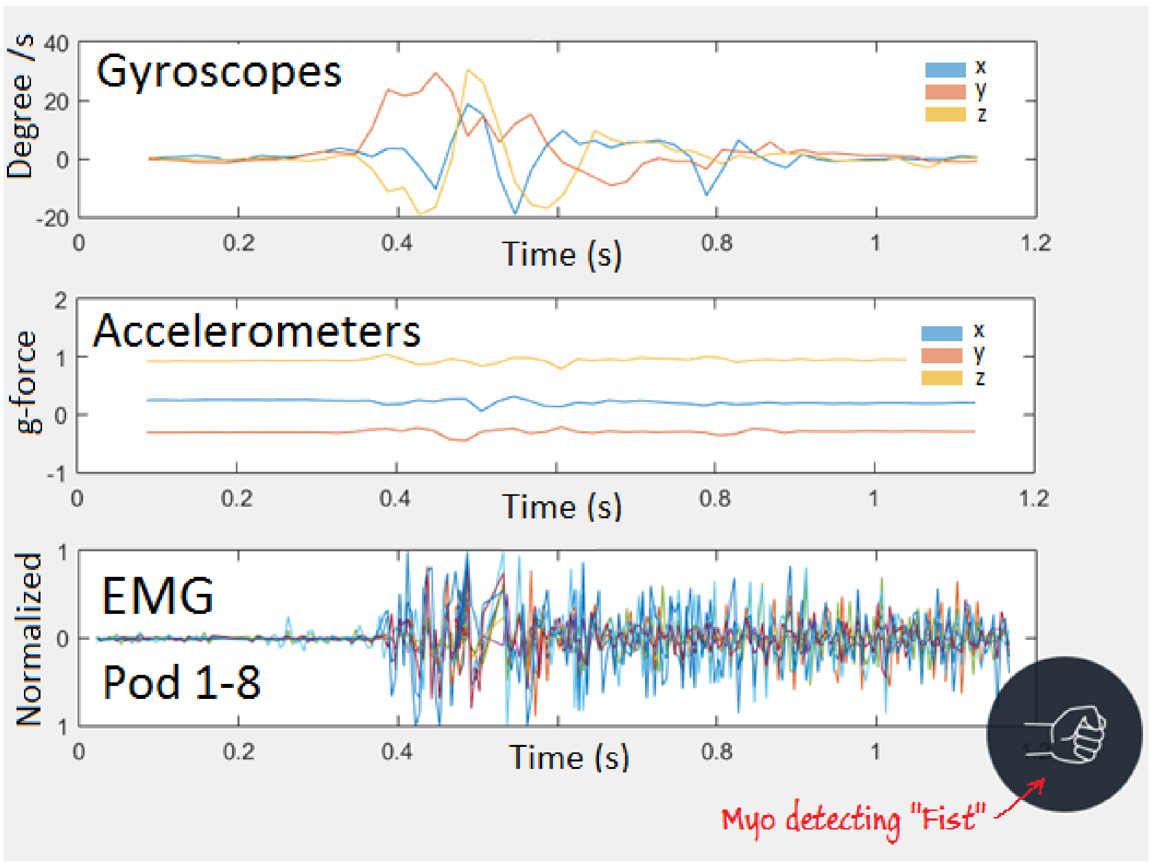
There were 2 different Matlab wrappers used during the development of this work. The first one allowed the acquisition of EMG raw data for a fixed time of acquisition of 10 seconds at a frequency of refreshing of 10Hz [14], which was used for the acquisition of the first set of experiments. The second wrapper, developed by Mark Tomaszewski [15] can stream data from the Myo at a tunable time at a frequency of 200Hz, which is the actual acquisition frequency of the device. This is a compilation of codes that wraps the C++ bindings of the SDK into a Matlab object, capable of streaming data in real time in the Matlab environment. Now, we will explain briefly the function of each step in this data conversion process.

## Matlab initialization

Every time a session for data acquisition is to be started, the file ***MyoMex_Quickstart.m*** should be executed, in order to call and compile the previous files and define them in the current Matlab workspace. Once the MyoMex object has been created, the streaming can be toggled on and off using a sub function, when its streaming any Matlab function can run without affecting the flow of data, including the *pause()* function. Also relevant properties can be accessed, such as Quaternion, gyroscopes, accelerometers and EMG data in real time. Also, logs for each property is created when the streaming start, which concatenates the received data until we use a sub function to clear the logs. Logs can be accessed at any time, and will show the current accumulated data.

After this code has allowed the creation of the temporary MEX file, it will run a short data acquisition test for a few seconds, to verify that everything is working correctly. Plotting the acquired values of the 3DOF gyroscopes, the 3DOF accelerometers and the raw EMG signals of the 8 pods overlapped, as shown in the following figure.

Figure 13 – Gyroscopes, Accelerometers and EMG signals. Myo detecting "Fist" pose



**Source:** Author

Overlapping the figure generated, we can see the result of the Myo's built in classifier. That figure will print in real time any pose detected by the device, as long as the program *Myo armband manager* is running.

After the placement of the Myo armband, and the Matlab initialization. The acquisition and labeling script ***MyoMex_mexbuilt.m*** will create a MyoMex object, and require a movie file, which is a *.mat* file, containing a sequence of images that execute as a visual guide for the pose we wish to record. The movie will repeat in a loop in cycles in an attempt to synchronize the patient with a guideline, by using a sound cue that indicates when the pose should be performed; this will repeat itself for 5 cycles. After the 2nd cycle, the data stream from the Myo will be activated and the recording will start, this means at least 3 poses will be recorded per recording session.

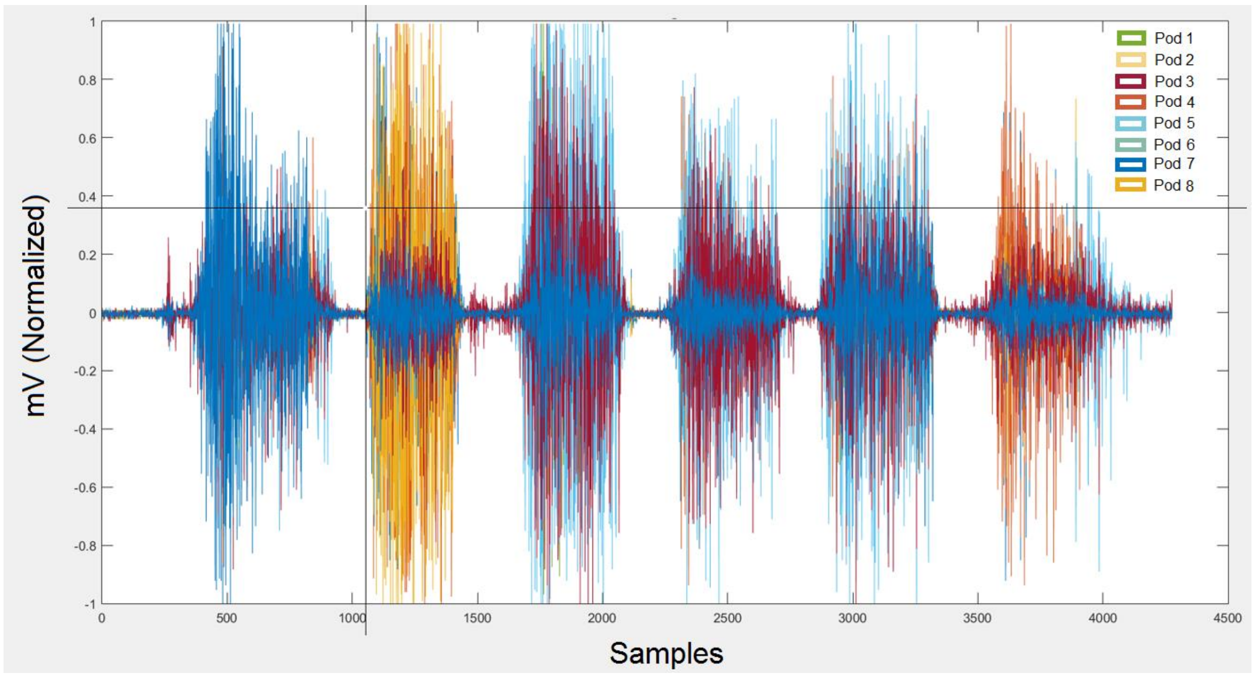
For the first experiments, while using the first Matlab wrapper, the data was acquired at a rate of 10Hz for a fixed time of 10 seconds, leading to fixed matrices of size 100x8 where the columns represent each channel and the rows were 100 observations.

For the second Matlab wrapper used, the frequency was 200Hz, meaning the amount of samples generated will be 200 times more than the time spent recording. For the sake of speed, no real time graphs are generated during the data acquisition, and the data logs will only be checked once the streaming is turned off. The data is stored in a matrix of variable size, with 8 fixed columns containing the EMG value of each POD at a given moment of time, and the number of rows will represent the amount of observations or samples.

### 4.1.5 Labeling

After the data acquisition is finished, a plot will appear with the overlapped EMG signals of the recorded session, and an interactive labeling system will execute automatically, where the user will see a black cross following the mouse cursor and can click in the graph where the pose transitions happened (figure 14). After clicking all transitions, pressing ENTER in the keyboard will pop messages in Matlab's command window displaying the sections defined by the transitions previously clicked, and asking to manually assign a label for each one of them.

Figure 14 – Interactive labeler, and signal samples for several poses.

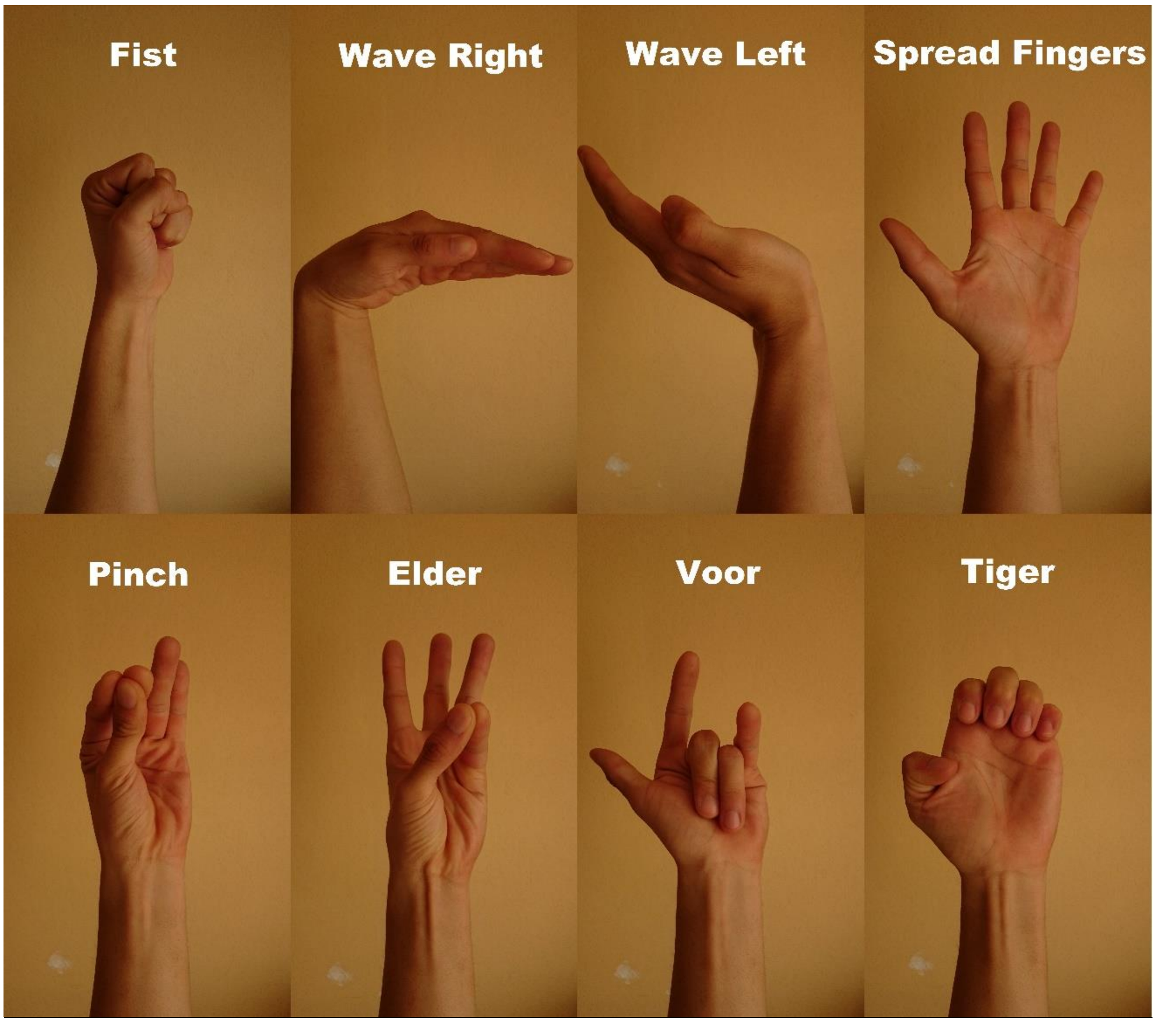


Source: Author

From left to right, the signal sections that present activity correspond to “Fist”, “Wave Right”, “Wave Left”, “Spread Fingers”, “Elder” and “Voor” poses. As we can see in the image, it’s easily differentiable when a pose is being executed from the general resting position.

The poses selected for the tests and the chosen labeling code is expressed in the following figure and table, where we can see a number and a color assigned to each pose. The numbers are used by the subsequent codes to identify the poses, while the color codes are meant as a visual guide for the user when the scores of the PCA are plotted. The latter will be deeply explained in **section 4.2.4.**

Figure 15 – Hand poses used in the experiments.

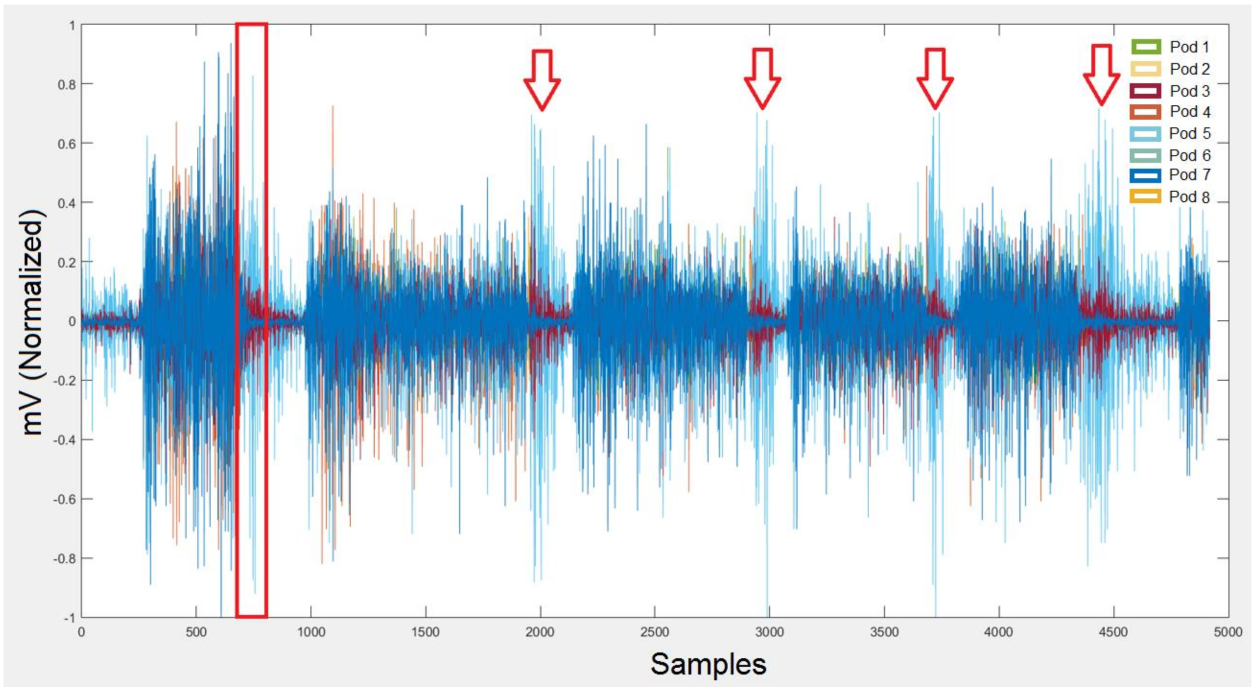


Source: Author

| Pose | Color code for PCA | Label |
|---|---|---|
| Rest | Cyan | 1 |
| Fist | Blue | 2 |
| Wave Left | Green | 3 |
| Wave Right | Red | 4 |
| Spread Fingers | Magenta | 5 |
| Pinch | Black | 6 |
| Elder | Red Cross | 7 |
| Voor | Blue cross | 8 |
| Tiger | Black cross | 9 |

However, when we release especially stressful positions like *Fist* or *Wave Out*, there is a second set of EMG readings (Figure 16) we must be careful not to label as the originally intended pose.

Figure 16 – “Fist” sample signal, releasing pose signals pointed with red markers



Source: Author

It is noticeable that this releasing EMG are not considered part of the pose by the built in classifier of the Myo, since whenever you release the pose it is immediately detected. It was learned empirically during this work how the accuracy of the classifiers would decrease if we fail to label as *Rest* (or some *Transition*) pose this second set of EMG signals.

After the data is labeled, a vector with the same length than the EMG current readings will be stored in a Matlab structure containing independent slots for both the raw EMG of data and the label. Then, the user should provide a name in the command window, and the Matlab structure will be saved as a *.mat* file in the *data/train* subfolder.

The previous process is the protocol for acquiring new data samples, and it was repeated several times for each pose. Whether it's training or testing data, the acquisition process will be basically the same. However, the labels will play different roles in the code if they are meant to be training or testing data; if the data is meant for training a new classifier, the labels will define a class for each observation. But if the new data acquired is meant for testing, the labels are used for defining the ground truth of the classifiers and calculating the accuracy as it will be **defined in section 4.3.5.**

## 4.2 Data preprocessing

All recorded *.mat* files should be saved by default in the same folder, in order to be compiled by the code *gatherProcessNTrain.m* which is responsible for gathering all available data, preprocessing, model training and storage. Now we will describe in general terms what is inside the code and their purpose.

### 4.2.1 Gather and concatenate all available data

Once the code is executed, the user will be asked to select a folder containing the training data, it should be the *data/train* subfolder. The program will load and concatenate all data files, creating a matrix with the same 8 columns for the EMG readings and a row number equal to the summatory of the recorded samples of all the loaded files. In the same way, a single vector will concatenate the corresponding labels for each observation.

In some classifiers like KNN, the testing of new data will require to compute each new sample against the whole training set. Thus becoming computationally expensive when the training data set is too big. Having this in mind, most of the experiments were done with rather non abundant data, in order to reduce the testing phase computational time.

### 4.2.2 Windowing

Given the stochastic nature of the EMG readings, we can't expect to extract too much information from the samples alone. Therefore, some windowing or smoothing technique is required for calculating additional features from a small set of data [17].

In general, the window size is related to the data length and the number of training samples (windows), as shown in the following equation:

$$\text{No. of training samples} = \frac{\text{data length} - \text{window size}}{\text{window increment}} + 1.$$

Selecting smaller window increments than the window size will lead to overlapping windows, which has been demonstrated to have better classification performance, than a disjoint window scheme [18], where the window size is equal to the window increment. For this work we chose to use the disjoint window scheme, since is computationally cheaper than the overlapping one.

In the literature, the requirements for real-time applications, like control of prosthetics or virtual avatars, define the window length to be smaller than 300ms [17]. We chose a window size of 5 samples at a frequency of acquisition of 200Hz, the window length in time is of 25ms, satisfying the proposed condition. On the other hand, since there is a label associated to each observation of the raw EMG signal, we will determine the label of an entire window by a majority vote of the labeled samples inside a window. Therefore, if the window size was higher, windows allocated in the transitions between poses would create misleading information for the classifier.

### 4.2.3 Features of EMG

After the window size is set, the concatenated data will be separated in windows, and we will calculate various features that can be extracted from the EMG raw data, each giving additional information of the muscle behavior. Some of them extracted from the magnitude variations of the signals, and some calculated from the frequency of certain changes. There are many different features which are very useful for training a classifier, but we will only mention those used in our study [12]. The calculated features for each window will be stored in a single vector, concatenating the features of all the pods.

#### *4.2.3.1 Integrated EMG (IEMG)*

Describes the EMG signal sequence firing point, as onset detection index of muscle activity. It's defined as a summation of absolute values of the EMG signal amplitude, which can be expressed as:

$$IEMG = \sum_{i=1}^{N} |Xi|$$

Where *Xi* represents the EMG signal in a segment *i*, and ***N*** refers to the length of EMG signal.

#### *4.2.3.2 Mean Absolute Value (MAV)*

Can be used to detect muscle contraction levels. It's defined as an average of absolute value of the EMG signal amplitude in a segment, expressed by the equation:

$$MAV = \frac{1}{N}\sum_{i=1}^{N} |Xi|$$

#### *4.2.3.3 Simple Square Integral (SSI)*

Uses the energy of the Motor Unit Potentials as a feature, it can be defined as the summation of the squared absolute value, as follows:

$$SSI = \sum_{i=1}^{N} Xi^2$$

### *4.2.3.4 Variance of EMG (V)*

Represents how signal varies from its average value with a statistical measure, also expressing the power of a single EMG reading, it's defined as follows:

$$V = \frac{1}{N-1}\sum_{i=1}^{N}(Xi - M)^2$$

Where M is the mean value of the signal, defined in a similar way as the MAV, but without the absolute value

### *4.2.3.5 Root Mean Square (RMS)*

This feature represents the mean power of the EMG signal and it's related to the constant force and non-fatiguing contraction of the muscle, RMS is defined as follows:

$$RMS = \sqrt[2]{\frac{1}{N}\sum_{i=1}^{N} Xi^2}$$

### *4.2.3.6 Waveform Length (WL)*

This feature represents the cumulative length of the EMG signal waveform over the time segment, or in other words WL is a measure of EMG signal complexity, relating the waveform amplitude, frequency and time, it's described by the equation:

$$WL = \sum_{i=1}^{N-1} |X_{i+1} - X_i|$$

### *4.2.3.7 Zero Crossing (ZC)*

This feature represents the number of times the amplitude of the EMG signal crosses the 0 axis, giving an insight in the frequency of the analyzed signal fragment, it can be formulated as follows:

$$ZC = \sum_{i=1}^{N-1} sgn(X_i * X_{i+1})$$

Where

$$\operatorname{sgn}(x) = \begin{cases} 1, & \textit{if } x \geq 0 \\ 0, & \textit{otherwise} \end{cases}$$

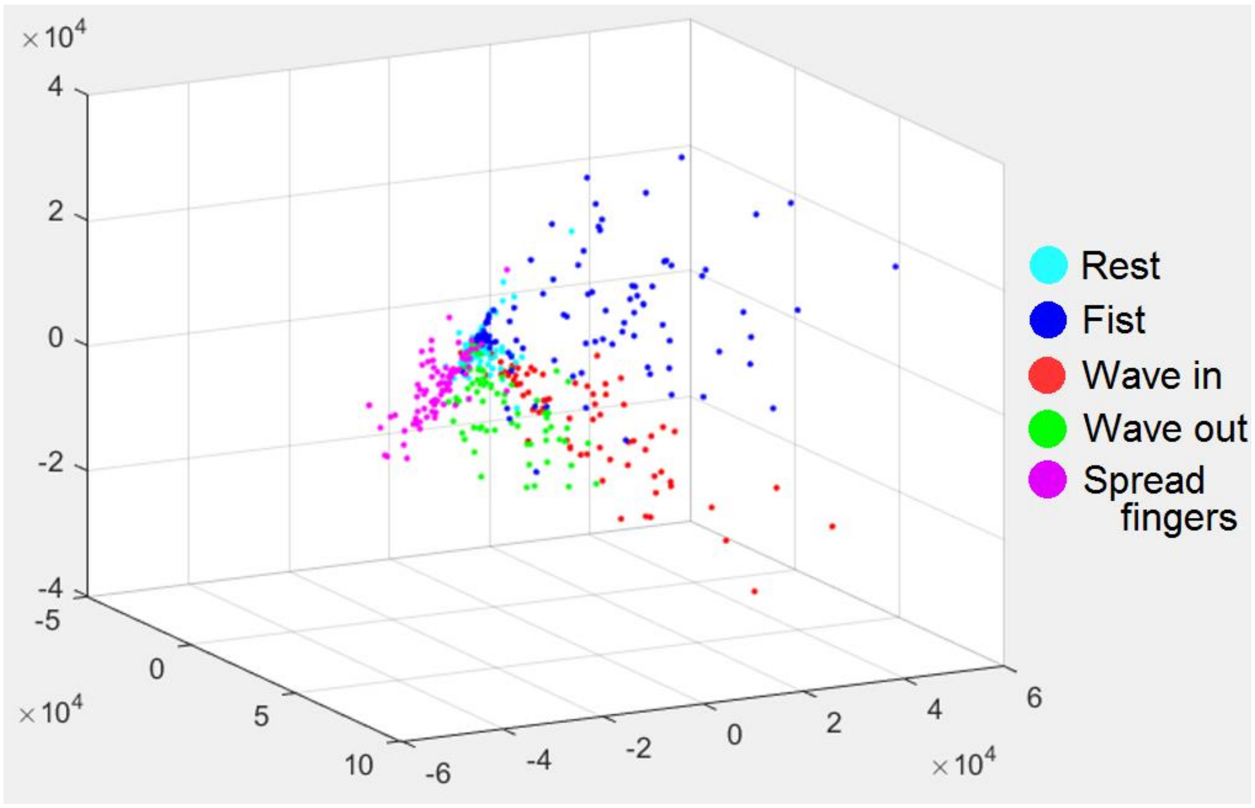
### 4.2.4 PCA

After all the features have been calculated for each window for all the pods, a matrix will be generated. Where the columns are represented by the vector containing the features of each window in each pod, and since we calculated 7 features per pod, and we have 8 pods, we will have 56 columns. And the rows will be the observations, whose number depends on the length of the data acquired.

For the generated matrix, we calculate the *Principal Component Analysis* (PCA), using the ***pca*** function in Matlab. PCA is a multivariate statistics algorithm, performing data orthogonal transformation converting a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables [17]. New variables are called *Principal Components* (PCs), and they have a dimensionality as big as the number of features provided to the PCA algorithm, this means we will have a score with 56 dimensions. We will use only 3 components for visualizing the data in 3D, and the whole 56 PCs for training different classifiers.

The following figure shows how just 3 components of the PCA are very helpful to realize the clustered projections of different poses.

Figure 17 – 3D PCA scores of basic dictionary with acquisition frequency of 10Hz



Source: Author

## 4.3 Modeling

Using the calculated scores and the labels for each window, we can create training data suitable as an input of different kinds of models and classifiers. As mentioned before, training and testing data are acquired and labeled in the same way, therefore we must clarify a distinction between them.

In general, training data is a compilation of several files containing many repetitions of all the poses we wish to train the classifier with. And testing data will be just two repetitions of each pose, recorded in different files than those used for training the classifiers.

In the following section we will describe briefly the principles behind each classifier and the definitions of accuracy created to measure the experiments made for each one of them.

### 4.3.1 Decision Trees

Decision tree learning is a method for representing data in form of if-then rules for approximating discrete-valued target functions. The decision tree representation sorts data down from the root of the tree, through branches corresponding to a discrete division of the possible values of the attribute. From the branches, nodes in the tree represents an attribute of the data, until finally leaf nodes provide classification of the data [19].

In Matlab, the function ***fitctree*** can generate a decision tree classifier, taking as input a table of values with features in the columns and observations in the rows, and a vector of labels or classes corresponding to each observation. This function returns a binary tree, where each branching node is split based on the provided features. Additionally, some other parameters can be defined in the Matlab function, such as minimum number of node observations, or maximum number of splits and the cost of misclassification, among many other customizable options to fit the classifier to specific needing [20].

### 4.3.2 Random Forest

Random forests combine the predictions of several decision trees, as the ones described in the previous section. Each of the trees is trained in isolation and combined using a weighting scheme through averaging. The construction of the tree is parameterized by the minimum number of estimation points in each leaf, indicating the minimum leaf size.

At each point in the construction of the tree, a leaf is selected at random for expansion as long as there exists a candidate that doesn’t create any children with fewer leafs than defined by the minimum leaf size. If there is no such candidate at the expanding point, the expansion stops [21].

In Matlab, we can create a Random Forest classifier using the function ***TreeBagger***, which takes as input the number of decision trees, a table of values with features in the columns and observations in the rows, and a vector of labels or classes corresponding to each observation. This function returns an ensemble of the decision trees used for the model training. The minimum leaf size can be defined as an optional function of the function, just as the cost of a misclassification and some other functions [22].

### 4.3.3 Error Correcting Output Codes (ECOC)

ECOC is a method for decomposing a multiway classification problem into several binary classification tasks, each removing some uncertainty about the correct input class. Then a voting scheme will decide by the combination of the results an estimated solution for the original problem [23].

In MATLAB, this classifier is trained using the function ***fitcecoc***, fed with a matrix of observations and attributes treated with PCA, and a vector with the unique corresponding labels [24].

### 4.3.4 K Nearest Neighbors (KNN)

Let's imagine a training data set with known labels for each observation, if we consider each characteristic or feature of the observations in some space, an observation of such features will project a coordinate in that dimension. Therefore, we can consider some kind of distance between different data points under some appropriate metric, and cluster the observations whose distance with neighboring observations is small.

A KNN classification algorithm would estimate the class or label of a new observation by performing a majority vote of the classes of the $K$ nearest neighbors, where $K$ is a positive integer and it's usually small. For our particular case, each class in the feature space is not clustered in a compact way, and the classes tend to overlap leading to the misclassification of some of the data points. There are many ways to make KNN more robust, such as assign a weight to the contributions of the neighbors, or even defining a weight for different labels, which is useful when there is an overabundance of some class data. Also, defining the weight in relation to a relative distance, meaning that the nearer neighbors contribute more to the voting than those far away [25].

In Matlab, a KNN classification model can be created with the function ***fitcknn***, which takes as an input the matrix of predictor values, meaning the result of the PCA performed to the features extracted from the windows of each EMG source. Another input to the function is the vector of classification, which is the label vector we create manually for each training sample. Additionally, some other parameters can be tuned in the Matlab function, such as *Weights* defining a vector with a specific weight for each observation, or *NumNeighbors* which is *K*, and *DistanceWeight* for defining a weight related to the distance of the neighbor for a particular sample, among many others [26].

### 4.3.5 Accuracy definition

#### *4.3.5.1 A1*

The accuracy of the classifiers was calculated based in the classifier's prediction against the label created for the testing data (ground truth), defining the accuracy from the number of correctly classified observations according the next equation.

$$A1 = \left(\frac{Number\ Of\ Correctly\ Classified\ Observations}{Total\ Number\ Of\ Observations}\right) * 100$$

#### *4.3.5.2 A2*

After performing some experiments, we realized the overabundance of the "Rest" pose in the data sets stated a problem, which is the inflated accuracy of the classifier given the correct classification of the *"Rest"* pose with respect to the other poses.

Therefore, a new accuracy was defined excluding the *"Rest"* pose from the manual label, and the incorrect classification of "*Res*t" from the classifier's results. The new accuracy A2, will give a better insight of the classifier's results and is calculated according to the following equation.

$$A2 = \left(\frac{Number\ Of\ Correctly\ Classified\ Observations - CorrectlyLabeled"Rest}{TotalNumberOfObservations - ManualLabeled"Rest - IncorrectlyLabeled"Rest}\right) * 100$$

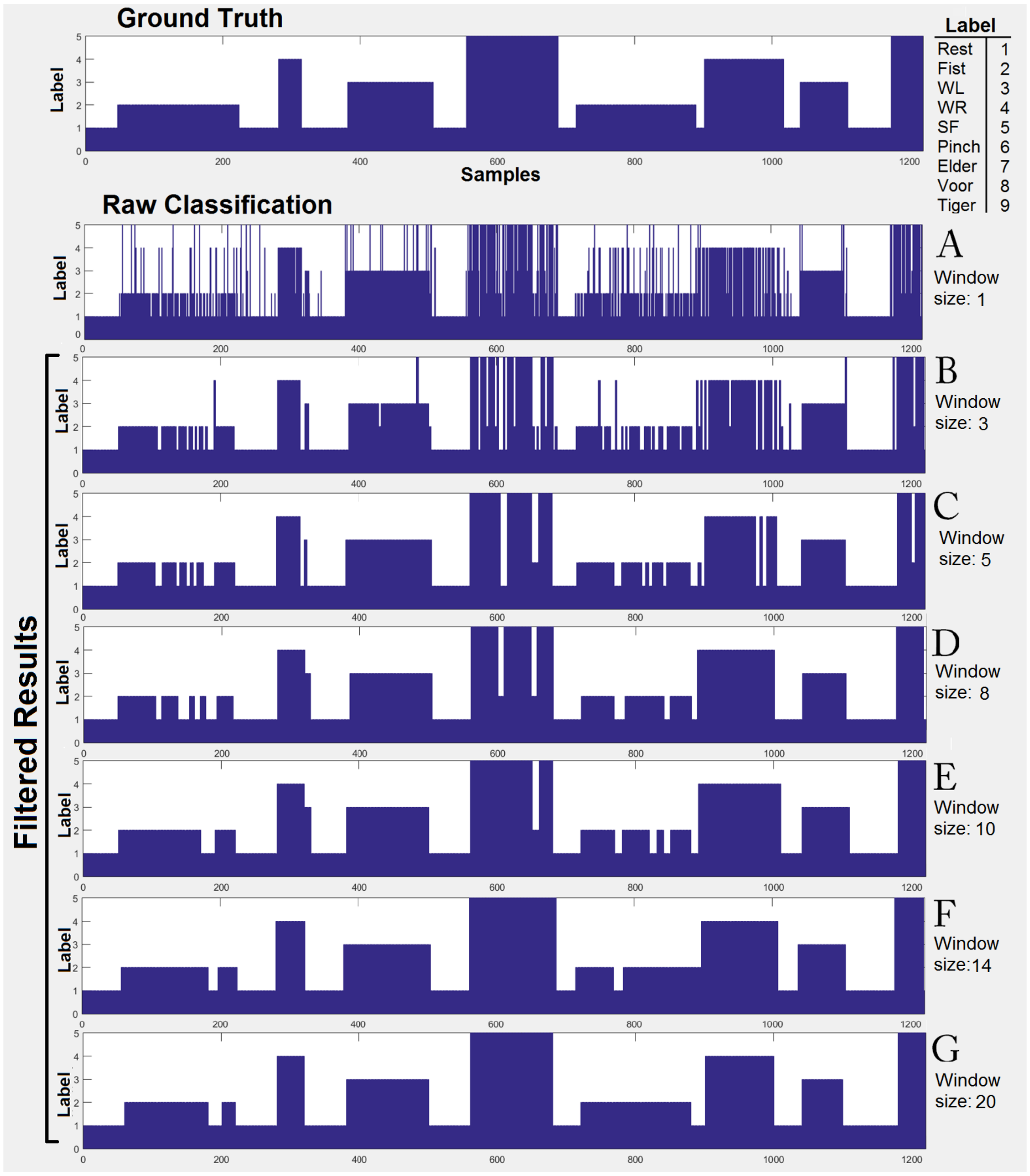
## 4.4 Post processing

During the experimentation with the 2nd Matlab wrapper, it was noticed that the nature of misclassification was anatomically impossible, since the classifier's results were *noisy*. Such results were suggesting the patient was switching from one hand gesture to another, and then jump back to the original pose in 2 samples, which is anatomically impossible for a human in 5ms (acquisition frequency of 200Hz). This scenario presented the possibility of performing some kind of low pass filtering to the classification results.

### 4.4.1 Sliding window filter

Due the nature of the misclassification, it is possible to filter the anomalies with a sliding window and a majority vote. This was developed from the codes used to separate the data in windows and a similar process as the one used previously for determining the label of a window by majority vote. In the next set of graphs we can appreciate the sliding window size influence in the results. The accuracies obtained for each window size can be seen in the subsequent table.

Figure 18 – Filter window size influence in classifier's results



Source: Author

| Filter Window size | Accuracy | | Graph Index |
|---|---|---|---|
| | A1 | A2 | |
| 1 | 74.4262 | 69.1344 | A |
| 3 | 81.6393 | 76.6515 | B |
| 5 | 86.1475 | 82.2323 | C |
| 8 | 84.6721 | 81.6629 | D |
| 10 | 88.1967 | 86.7882 | E |
| 14 | 92.2951 | 92.4829 | F |
| 20 | 90.6557 | 87.9271 | G |

Despite looking better with an increasing window size, there is not a linear relationship between the filter window size and the resulting accuracy, and after some number further increasing the window size will only decrease the accuracy. Therefore, the filtering algorithm will be improved into an iterative one, maximizing the accuracy and exhausting all possibilities in order to get the best results.

### 4.4.2 Maximizing accuracy

As mentioned before, the accuracy and the window size of the filter will vary according the results of each classifier. Through experimentation it was determined that in general, a window bigger that 40 would only decrease the accuracy. Therefore an iterative loop from 5 to 40 will calculate the accuracies A1 and A2 and store the filtered results for the highest A2 accuracy, keeping record of the calculated filter window size

# 5 Experimental Results

## 5.1 Experiments with 1st Matlab wrapper, acquisition at 10Hz

The purpose of the first set of experiments was to determine empirically the best classifier from those selected for our particular application, and the right parameters to achieve a robust classifier. In order to achieve this, a huge data set was recorded in a single session for the analysis of these experiments. These first results were achieved with an acquisition frequency of 10Hz, considerably inferior to the 200Hz frequency of future experiments.

Each one the *Training data batches* presented in the following tables correspond to an entire file with 100 samples, where experiments were separated in sets of 2500, 2800, 3000 and 4000 samples (25, 28, 30 and 40 *Training data batches* respectively). In general, the tables will show the accuracy achieved by the classifier under different conditions or parameters.

All the experiments in this section were performed on the same testing data, acquired separately from the training data. The testing data consist of two data batches per pose, resulting in 1000 samples containing two repetitions for each pose in the base dictionary of the Myo armband's classifier. If new poses are to be tested, two additional data batches will be concatenated to this initial testing data.

### 5.1.1 Decision Tree

The following table shows the accuracy A1 of the classifier on the testing data, against the amount of training data used to train the model. All the models are applied in the same testing data with a window size of 5, in order to hold a relation.

| Train data batches | A1 |
|---|---|
| 25 | 76.25 |
| 28 | 77.5 |
| 30 | 76.25 |
| 40 | 76.25 |

### 5.1.2 Random Forest

In this table we can appreciate how an incremental number of train data batches is not directly related to the improvement of the classifier. However, since the accuracy of the random forest is the result of the average of the independent decision trees we can see how an incremental number of trees has a positive influence in the classifier's overall accuracy, since the depth of the trees is related to the minimum leaf size, which is also related to the size of the training set.

| Train data batches \ N. of trees | 2 | 4 | 6 |
|---|---|---|---|
| 25 | 51.875 | 64.375 | 69.375 |
| 30 | 70 | 71.25 | 81.875 |
| 40 | 64.375 | 79.375 | 80 |

The next table shows the relation between the minimum leaf size and the number of samples for a fixed *number of trees* of 6.

| Train data batches \ Min Leaf size | 2 | 4 | 5 |
|---|---|---|---|
| 25 | 69.375 | 75 | 69.375 |
| 30 | 78.125 | 76.25 | 81.875 |
| 40 | 71.25 | 75 | 76.25 |

It's worth mentioning that whether the accuracy in general seems inferior to the one achieved by the classifier with minimum leaf size of 1 (which is the default), the accuracy for detecting the poses increased while the correct classification of the "Rest" pose decreased.

### 5.1.3 Error Correcting Output Codes (ECOC)

The ECOC model shows increasing accuracy in relation to the amount of training data. But in comparison with the other classifiers, this is the less effective classifier and will be discarded for future experiments.

| Train data batches | Accuracy |
|---|---|
| 25 | 62.5 |
| 28 | 63.125 |
| 30 | 69.375 |
| 40 | 71.875 |

### 5.1.4 K Nearest Neighbors (KNN)

The following table shows the accuracy of the classifier, illustrating the relation between the amount of training data and the K nearest neighbors used to train the model. As well as the previous models, this ones are applied in the same testing data with a window size of 5, in order to hold a relation.

| Train data batches \ K | 1 | 3 | 5 | 10 |
|---|---|---|---|---|
| 25 | 78.75 | 80.625 | 81.875 | 81.25 |
| 28 | 84.375 | 83.75 | 86.25 | 86.875 |
| 30 | 88.125 | 85 | 85.625 | 88.125 |
| 40 | 89.375 | 89.375 | 90 | 90 |

For the 40 train samples, an additional measurement was taken with K=15, resulting in an accuracy of 88.125, this decrease is attributed to the occasional overlapping of data between “Wave Left” and “Spread Fingers” poses, since they involve similar muscle behavior.

From the previous experiment we could define the future data acquisition with a fixed K=10. However, as new poses are introduced in the dictionary of the classifier, the chances of overlapping data between the new poses are high. Thus a K=5 will be used for future model trainings.

### 5.1.5 KNN: Using new accuracy A2 and modifying the weight of “Rest” pose

There is a drawback for the majority voting strategy, since examples of the most frequent class *“Rest”* tend to dominate the prediction of the new examples, due to its overabundance in the data sets. This also states an additional problem, which is the inflated accuracy of the classifier given the correct classification of the *“Rest”* pose with respect to the other poses.

In order to solve these problems, we will take 2 approaches. First we will use the weight parameter in the Matlab function, assigning an almost neglectable weight to the *“Rest”* labeled neighbors, to reduce the impact of the overabundance of this class.

In the following table we can appreciate the comparison of the suggested approaches in the results of the 5 KNN classifier, where A1 is the overall accuracy of the classifier, and A2 is the accuracy calculated without taking into account the "Rest" pose influence.

| Weight of Rest | 1 | | 0.8 | | 0.5 | |
|---|---|---|---|---|---|---|
| Train Samples | A1 | A2 | A1 | A2 | A1 | A2 |
| 25 | 81.875 | 68.254 | 82.5 | 69.841 | 81.25 | 71.428 |
| 28 | 86.25 | 73.015 | 86.25 | 74.603 | 86.25 | 76.19 |
| 30 | 85.625 | 71.428 | 85.625 | 74.603 | 83.75 | 74.603 |
| 40 | 90 | 84.127 | 90 | 85.714 | 88.125 | 87.301 |

### 5.1.6 KNN: Expanding the dictionary

A new pose *"Pinch"* was introduced as training data for the classifier with 5 nearest neighbors, using the previous 40 batches of train data plus 5 more with information about the new pose. As expected, the accuracy A1 of the new model decreased from 90 to 83.75. However, our Matlab algorithm provides detailed information about the misclassification, allowing us to know which poses are misclassified and what was their wrong result.

Then, we created additional training data for the poses creating the misclassification, resulting in the accuracy A1 improving from 83.75 to 85.625 with 3 new data batches, thus validating the strategy. However, we need to be careful since introducing a mislabeled data batch can catastrophically affect the accuracy of the classifier.

## 5.2 Experiments with 2nd Matlab wrapper

As stated previously the frequency of acquisition for the new wrapper will be of 200Hz, meaning the amount of samples per data file will increase significantly and won't have a fixed size, therefore the tables will state the number of total samples, rather than the data batches presented before for training and testing the classifiers.

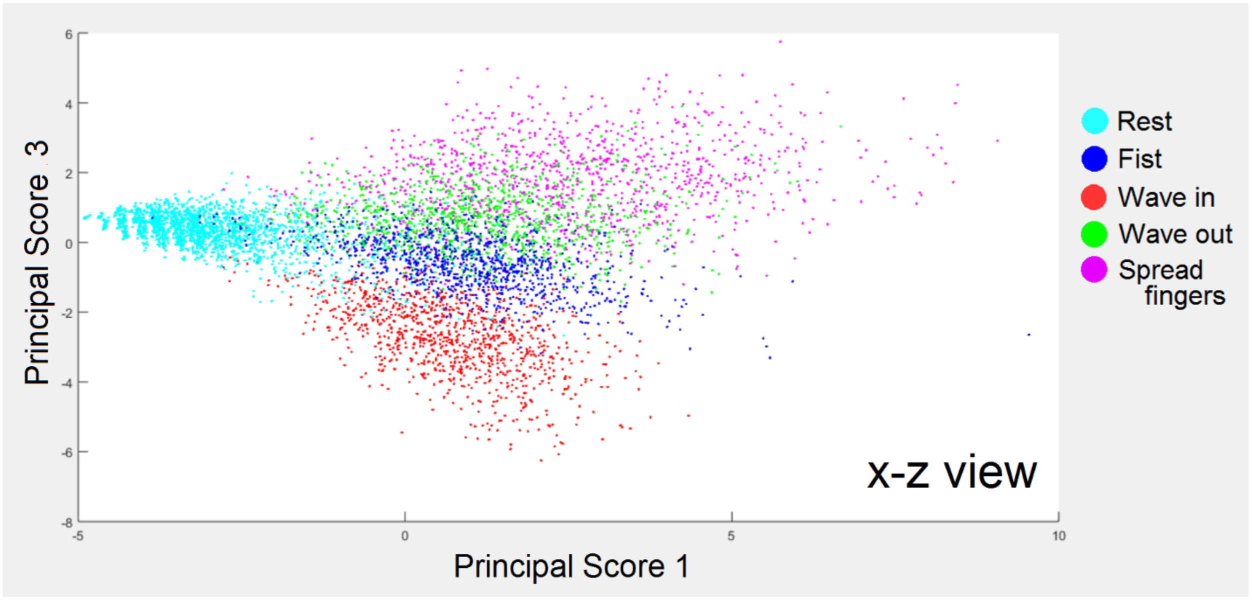
In a similar way as before, training and testing data was created separately. For all the experiments using the 2nd Matlab wrapper we will have a single file for the testing data, containing 2 repetitions of each pose of the base dictionary. When new poses are to be tested, new files containing two repetitions of each additional pose are concatenated to the original base dictionary.

In this second set of experiments, we will separate the experiments per acquisition sessions. Since as it was stated before, training and testing data should be acquired in the same session without removing the Myo armband.

### 5.2.1 1st Session, facing 200Hz and post processing

The first issue to be noticed after switching the acquisition frequency from 10Hz to 200Hz was visible after the PCA was done. The new data was significantly more abundant and scattered along the PCA 3D graph than for previous experiments, overlapping more often between classes than before. Nevertheless, it's still visible some form of cluster for each pose, thus making the 3D PCA graph still valuable for considering new poses.

Figure 19 – 3D PCA scores of basic dictionary with acquisition frequency of 200Hz



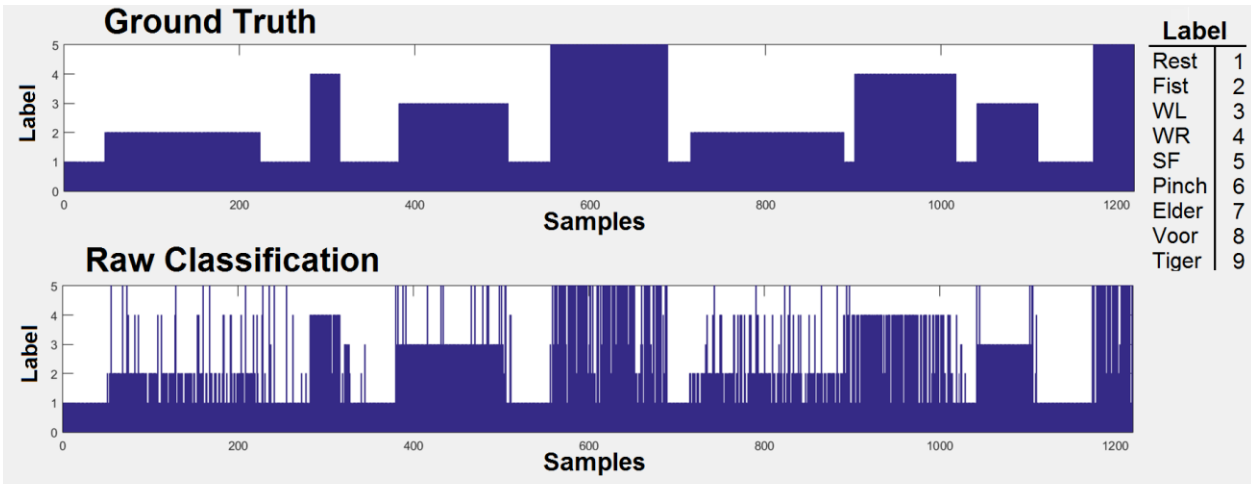
Source: Author

In the following table, we will evaluate a KNN classifier with 5 nearest neighbors, in order to state a comparison with the results obtained with the 1st Matlab wrapper. As we can see, it has a significantly lower accuracy than previously achieved, even with a huge abundance of observations available.

| Weight of Rest | 0.8 | | 0.5 | |
|---|---|---|---|---|
| Total Samples | A1 | A2 | A1 | A2 |
| 20052 | 58.8525 | 45.7859 | 61.254 | 49.8644 |
| 39786 | 71.9672 | 64.4647 | 73.9344 | 67.9954 |
| 64486 | 72.9508 | 66.2870 | 74.4262 | 69.1344 |

In the following picture we can see in the upper graph, the ground truth defined by the manual label created for the testing data, the lower graph shows the result of the classifier trained with 64486 samples and "Rest" weight of 0.5. The X axis represents the observations, while the Y axis represents the labels.

Figure 20 – KNN: Ground truth VS raw classification



Source: Author

When analyzing the data, it was noticed that the nature of misclassification was anatomically impossible. Meaning it's not anatomically possible to switch from one hand gesture to another, and then jump back to the original pose in a single sample. Which is graphically described as the spikes in the classifier's results.

It was during this experimental session we realized the suitability of post processing the data with a low pass windowing filter; this process was presented carefully in section 4.4.1.

### 5.2.2 2nd Session, evaluating previous models for new poses

In this session, there were two main objectives; the first one was to determine which new poses could fit the previous dictionary without affecting too drastically the accuracy, and the second objective was to determine which model was better for expanding the dictionary.

Independent models were created for training and testing each of the new poses and similar parameters will be tuned for each classifier as those shown for the first Matlab Wrapper, however emphasis will not be made in the number of samples, but with the interaction between the Base set of poses (Fist, Wave Left, Wave Right and Spread Fingers), and the additional 4 poses we wish to evaluate. In addition, the results shown in tables will already have the sliding window filter applied. Only a graph of the best case for each classifier will be shown for the sake of space, to give the reader a graphical insight in the classifier's and post processing results.

#### *5.2.2.1 Random Forest*

Let us remember that A1 stands for the accuracy of the classifier, counting the "Rest" position as a correct classification, and A2, the same measurement without taking that pose into account. Having this in mind we can realize that the best classification results are those which A1 and A2 are both high.

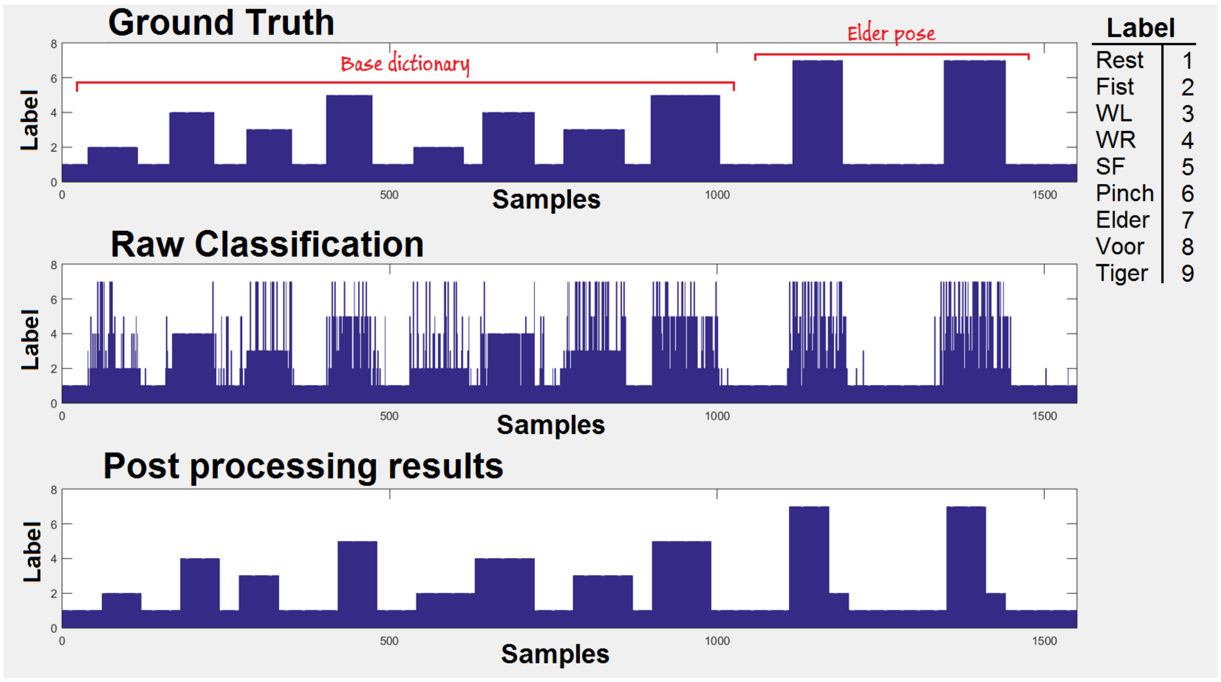
| Number of trees | 2 | | 4 | | 6 | |
|---|---|---|---|---|---|---|
| Poses | A1 | A2 | A1 | A2 | A1 | A2 |
| Base | 79.6460 | 70.0627 | 90.0688 | 85.5799 | 90.0688 | 87.3041 |
| Base+Pinch | 67.6774 | 42.002 | 73.6129 | 53.602 | 74.3871 | 55.3114 |
| Base+Tiger | 70.2128 | 50.4695 | 82.7853 | 73.1221 | **84.0748** | **75.8216** |
| Base+ Elder | 73.5313 | 52.7228 | 78.5023 | 66.2129 | 80.1162 | 65.7178 |
| Base+Voor | 65.4616 | 42.6802 | 80.6972 | 69.2568 | **81.8593** | **73.7613** |

So far the only promising poses appear to be Tiger and Voor, the other 2 introduce too much confusion into the classification of the Base poses, since they involve similar muscular and mechanical behaviors into the arm. Now we will modify the minimum leaf size, which defines the depth of the trees, for a fixed number of trees of 6.

| Min Leaf size | 1 | | 4 | | 7 | |
|---|---|---|---|---|---|---|
| Poses | A1 | A2 | A1 | A2 | A1 | A2 |
| Base | 90.0688 | 87.3041 | 93.4120 | 91.6928 | 92.7237 | 91.3793 |
| Base+Pinch | 74.3871 | 55.3114 | 75.2903 | 56.4103 | **79.0968** | **64.2247** |
| Base+Tiger | 84.0748 | 75.8216 | 86.9117 | 81.9249 | 87.6209 | 81.3380 |
| Base+Elder | 80.1162 | 65.7178 | 89.0252 | 84.7772 | 83.9897 | 73.2673 |
| Base+Voor | 81.8593 | 73.7613 | 86.3138 | 87.1622 | 87.6049 | 84.9099 |
| Base+Tiger+Voor | 69.8512 | 46.0073 | 72.6356 | 54.9909 | 76.6683 | 62.2505 |
| Base+ Elder +Voor | 84.1903 | 77.5992 | 89.9087 | 85.5388 | 90.8698 | 87.3346 |

It's remarkable how the pose "Elder" was only detectable under the minimum leaf size of 4, in any other conditions it would affect negatively the classification of base poses and it couldn't be detected by the classifier. In the following figure we can appreciate how the "Elder" pose (label 7) affects the base dictionary, and how the post processing improves the noisy results.

Figure 21 – Random Forest: testing base dictionary plus “Elder” pose

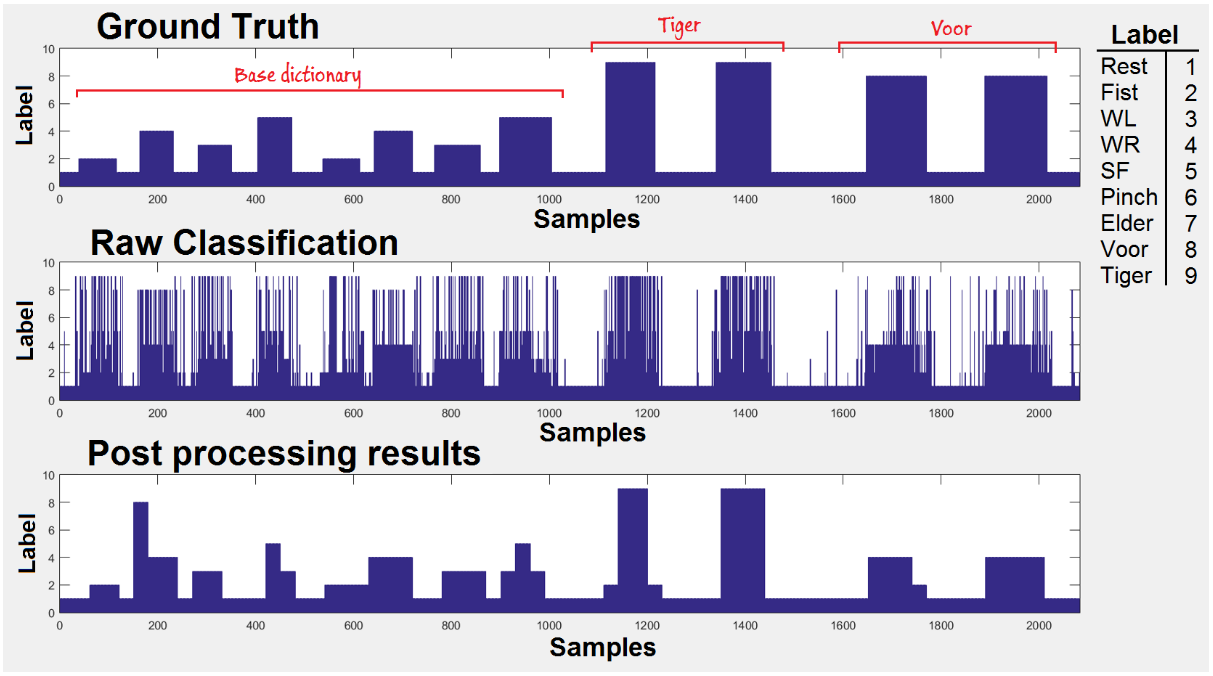


| Label | |
|---|---|
| Rest | 1 |
| Fist | 2 |
| WL | 3 |
| WR | 4 |
| SF | 5 |
| Pinch | 6 |
| Elder | 7 |
| Voor | 8 |
| Tiger | 9 |

Source: Author

Additionally, we tried to add 2 new poses into our Random Forest, relating those poses which got the best results for expanding the base dictionary. In the following figure, we can see the results for the classification of the base dictionary in addition to the poses “Voor” and “Tiger”.

Figure 22 – Random Forest: testing base dictionary plus “Voor” and “Tiger” poses



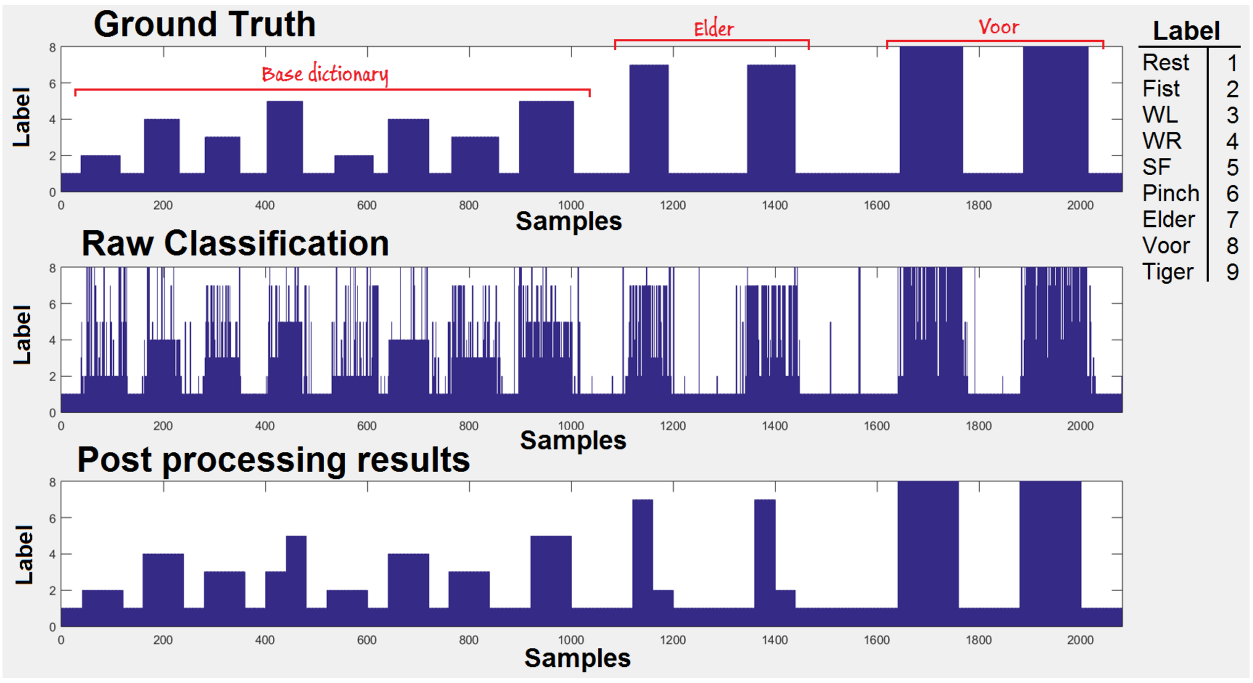
| Label | |
|---|---|
| Rest | 1 |
| Fist | 2 |
| WL | 3 |
| WR | 4 |
| SF | 5 |
| Pinch | 6 |
| Elder | 7 |
| Voor | 8 |
| Tiger | 9 |

Source: Author

We can see how the introduction of the “Tiger” pose introduce a drastic increase in the misclassification of the base dictionary, and the pose “Tiger” is unrecognizable. Suggesting these two poses are not especially compatible in the dictionary expansion.

In the following figure we can see how the base dictionary was expanded with “Voor” and “Elder” poses, without compromising too much the classification of the original poses, and allowing the classification of “Elder” in some grade.

Figure 23 – Random Forest: testing base dictionary plus “Voor” and “Elder” poses



**Source:** Author

### *5.2.2.2 Error Correcting Output Codes (ECOC)*

For the correct classification of the ECOC using the Matlab function, the class names must be provided inside the function, aside from the vector containing the labels of the training data. The following table summarizes the influence of each pose into the base dictionary.

| Poses | A1 | A2 |
|---|---|---|
| Base | 90.4621 | 91.2226 |
| Base+Pinch | 82.6452 | 72.8938 |
| Base+Tiger | 86.0735 | 85.3286 |
| Base+ Elder | 78.825 | 74.7525 |
| Base+Voor | 82.1175 | 83.6712 |
| Base+Tiger+Voor | 74.94 | 58.9837 |

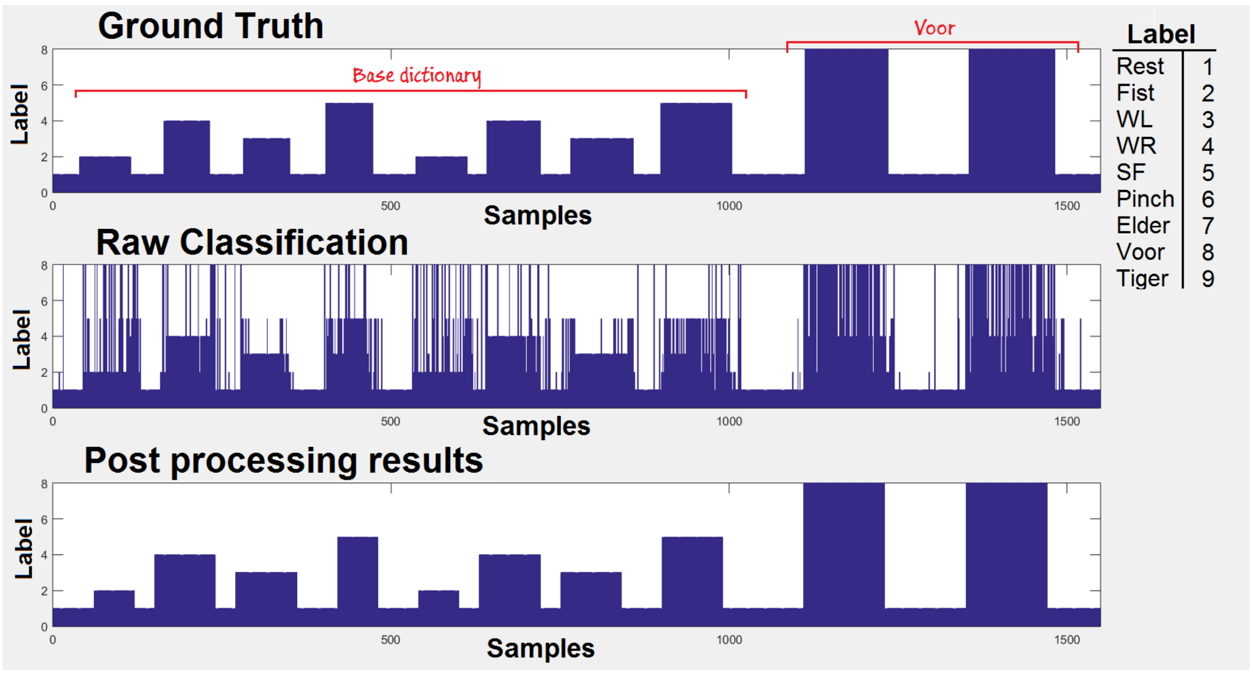
#### *5.2.2.3 K Nearest Neighbours (KNN)*

Based on our previous experiences with overabundance and diversity of the "Rest" pose samples, we will modify K for using fixed weight of 0.5 for every sample labeled as Rest.

| Number of neighbors | 3 | | 5 | | 10 | |
|---|---|---|---|---|---|---|
| Poses | A1 | A2 | A1 | A2 | A1 | A2 |
| Base | 88.9872 | 89.4984 | 86.2340 | 86.0502 | 89.8722 | 90.7524 |
| Base+Pinch | 75.5484 | 64.9573 | 78.7097 | 65.812 | 76.1935 | 62.5153 |
| Base+Tiger | 87.7498 | 90.7277 | 82.334 | 73.5915 | 81.109 | 73.7089 |
| Base+ Elder | 83.6023 | 73.2673 | 81.3428 | 79.5792 | 83.5378 | 80.4455 |
| Base+Voor | 89.7353 | 92.3423 | 92.1885 | 92.9054 | 93.0278 | 93.1306 |
| Base+Tiger+Voor | 74.988 | 59.4374 | 72.8757 | 54.9909 | 76.5242 | 61.9782 |
| Base+Pinch+Voor | 83.8617 | 79.7007 | 82.2286 | 73.1525 | 84.4861 | 77.7362 |
| Base+ Elder +Voor | 90.2451 | 87.3346 | 89.7645 | 91.3989 | 88.3229 | 88.5633 |

In the following figures we can see the results marked in the table, showing an outstanding compatibility between the "Voor" pose and the Base dictionary. And the only compatibility between 2 additional poses to the base dictionary, "Voor" and "Elder".

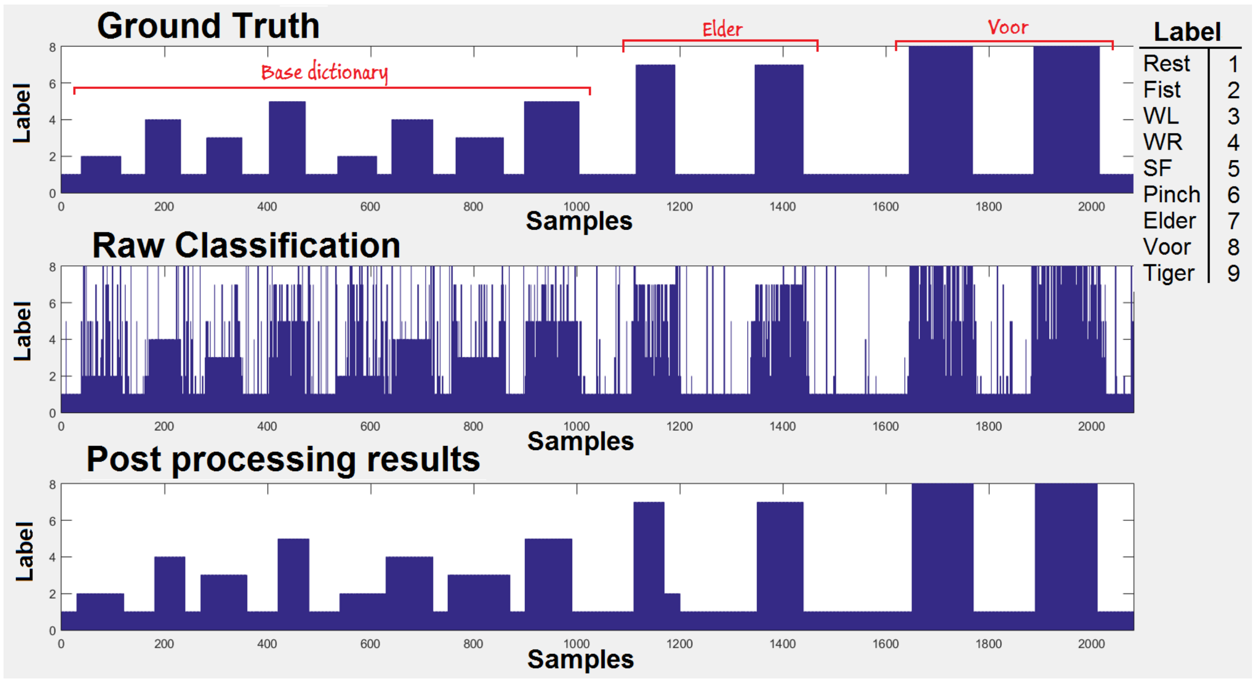
Figure 24 – 10 Nearest Neighbors: testing base dictionary plus "Voor"



| Label | |
|---|---|
| Rest | 1 |
| Fist | 2 |
| WL | 3 |
| WR | 4 |
| SF | 5 |
| Pinch | 6 |
| Elder | 7 |
| Voor | 8 |
| Tiger | 9 |

Source: Author

Figure 25 – 5 Nearest Neighbors test: base dictionary plus “Voor” and “Elder” poses



Source: Author

We can appreciate in the figure the successful classification of the base dictionary with 2 additional poses, and the significant improvement of post processing the data.

# 6 Conclusions and future work proposals

We successfully have learned the principles of electromyography behind the functioning of the Myo armband, as well as its SDK, whose communication protocols enabled data stream for processing in Matlab. Additionally, we learned and implemented signal processing and feature extraction techniques, needed to extract valuable information from the stochastic EMG signals.

Extensive data acquisition sessions were performed in order to record many different hand poses, and all data was manually labeled in order to train and test different classifiers. It was necessary to be extremely careful during the manual labeling of the samples, since mislabeled data would affect terribly the results of the classifier.

Several classifiers were explored through experimentation: Decision Trees, Random Forests, ECOC and KNN were tested, and their parameters modified in order to reproduce the ability of the Myo armband of classifying a set of basic hand poses, and later expanding the dictionary.

The system developed was compared to the Myo armband's software by graphical cues, since the software of the device doesn't provide directly the results of the built in classifier, but rather prints the detected pose in the screen in real time. Therefore, a special way of determining the accuracy of the classifier was proposed by making a general comparison between the results of the used classifier against the manual labeling of the testing data.

However, in our experiments the "Rest" position had an overabundance of samples in both the training and testing data sets, thus having a huge impact in the accuracy calculations, since "Rest" was defined as a pose too. Therefore, a second accuracy was defined for the pattern recognition algorithms, excluding the non-recognizable and idle patterns from the overall accuracy, to give a precise insight about the classifiers behavior.

Given the stochastic nature of the EMG signals and their subsequent features, the classifier's results were always very noisy. But, when analyzing the data, it was noticed that the nature of misclassification was anatomically impossible. Meaning it's not possible to switch from one hand gesture to another, and then jump back to the original pose in subsequent samples at 200Hz. Therefore, it was possible to filter the anomalies using processes implemented in previous tasks, such as a sliding window filter and a majority vote.

The sliding window filter enhanced the classifier's results in an outstanding way. However, each case would require a different window size to achieve the best possible results, and there was no linear relationship between the window size and the improvement of the accuracy. Therefore, enhancing the filter with a maximization algorithm allowed the automatic optimal filtering for each experiment.

However there is a drawback on the current filter approach, since the average window size of the filter throughout the classifiers was 18, and the window size of the EMG signals last 25ms; this means the fastest classification time our algorithm can perform for optimal accuracy takes at least 450ms. Let's remember that the requirements for real-time applications, like control of prosthetics or virtual avatars, define the refresh rate to be smaller than 300ms [17]. Proving that so far, our classifier is not ready for real-time applications.

For expanding the vocabulary of gestures of our classifier we needed to explore hand poses that involved different sets of muscles than those of the base dictionary, otherwise a new pose would introduce misclassification errors to the poses previously learned. This means some anatomical knowledge of the arm's mechanisms was required to define the ideal additional poses. However, there was no physical formation of the author allowing to define such poses. But, the 3D visualization of the PCA scores allowed some insight in the same principles required for defining a new pose. As shown previously, each sample from each pose would have a projection in the 3D PCA graph, allowing to determine graphically new hand poses that were essentially different from those learned already.

The procedure we just described for learning new poses grew more and more complicated as we were able to introduce new poses to our dictionary, since every new pose should be compatible with every other pose existing in the dictionary. At the end we were able to train KNN and Random Forest classifiers, capable of learning 2 additional poses to the base dictionary. Thus satisfying the main objective of this thesis work.

As it was mentioned at beginning, this work serves as the entrance to deeper and more natural cybernetic interactions. Now we will suggest to the reader, some possible future works that could use this document and algorithms, as a starting point.

The principles and procedures presented in this thesis work should be validated in amputated patients, since the physiological configuration of an amputee will surely define a unique hand pose dictionary for each of them, and present new challenges to the classification problem.

Additionally, a control system could be developed to control a hand prosthetic or a virtual reality interface, powered by the hand poses detected. DARPA has already proven that you can decode the patient's intentions to the point of controlling smoothly each finger of the hand, using Myo armbands, machine learning and some surgical interventions.

A virtual reality interface or a functional prosthetic decoding the user's intention would give sensory feedback to the brain when performing some hand pose. In a recently amputated patient, this feedback could prevent the reorganization of his PNS, CNS and brain, and subsequently prevent the development of PLP. Therefore a study could attempt to quantify the PLP reduction in different patients, at different stages of cortical reorganization.

We stated previously the importance of correct labeling for training the classifiers. If we are able to create accurate and complex labels, we could teach some equally complex gestures to the classifier. This was the main idea behind the glove for automated accurate labeling presented in the appendix, where the glove will provide detailed information of any hand current gesture and orientation, and generate a label for it. The application of this concept wouldn't be usable in amputees, but has potential in the development of VR interfaces and enhancement of the Myo armband's capabilities for detecting truly complex gestures.

# 7 REFERENCES

[1] Di Pino, G., Guglielmelli, E., & Rossini, P. M. (2009). Neuroplasticity in amputees: main implications on bidirectional interfacing of cybernetic hand prostheses. *Progress in neurobiology*, *88*(2), 114-126.

[2] Burck, J. M., Bigelow, J., & Harshbarger, S. D. (2011). Revolutionizing prosthetics: Systems engineering challenges and opportunities. *Johns Hopkins APL Technical Digest, 30*(3), 186-197.

[3] DARPA RSS - http://www.darpa.mil/program/revolutionizing-prosthetics

[4] DARPA RSS - http://www.darpa.mil/news-events/2015-09-11

[5] APL's Modular Prosthetic Limb Reaches New Levels of Operability - http://www.jhuapl.edu/newscenter/pressreleases/2016/160112.asp

[6] Yousefi, J., & Hamilton-Wright, A. (2014). Characterizing EMG data using machine-learning tools. *Computers in biology and medicine*, *51*, 1-13.

[7] Pan, L., Zhang, D., Liu, J., Sheng, X., & Zhu, X. (2014). Continuous estimation of finger joint angles under different static wrist motions from surface EMG signals. *Biomedical Signal Processing and Control*, *14*, 265-271.

[8] Riillo, F., Quitadamo, L. R., Cavrini, F., Gruppioni, E., Pinto, C. A., Pastò, N. C., ... & Saggio, G. (2014). Optimization of EMG-based hand gesture recognition: Supervised vs. unsupervised data preprocessing on healthy subjects and transradial amputees. *Biomedical Signal Processing and Control*, *14*, 117-125.

[9] Boyali, A., & Hashimoto, N. (2016). Spectral Collaborative Representation based Classification for hand gestures recognition on electromyography signals. *Biomedical Signal Processing and Control*, *24*, 11-18.

[10] Ramachandran, V. S., & Blakeslee, S. (1999). *Phantoms in the Brain: Human Nature and the Architecture of the Mind.* Fourth Estate.

[11] Dietrich, C., Walter-Walsh, K., Preißler, S., Hofmann, G. O., Witte, O. W., Miltner, W. H., & Weiss, T. (2012). Sensory feedback prosthesis reduces phantom limb pain: proof of a principle. *Neuroscience letters, 507*(2), 97-100.

[12] Phinyomark, A., Limsakul, C., & Phukpattaranont, P. (2009). A novel feature extraction for robust EMG pattern recognition. *arXiv preprint arXiv*:0912.3973.

[13] Myo SDK 0.9.0: The Myo SDK - https://developer.thalmic.com/docs/api_reference/platform/the-sdk.html

[14] GitHub – User: Boyali - https://github.com/boyali/matMYO

[15] Myo SDK MATLAB MEX Wrapper by Mark Tomaszewski: http://www.mathworks.com/matlabcentral/fileexchange/55817-myo-sdk-matlab-mex-wrapper

[16] MEX File Creation API - http://www.mathworks.com/help/matlab/call-mex-files-1.html

[17] Khushaba, R. N., Kodagoda, S., Takruri, M., & Dissanayake, G. (2012). Toward improved control of prosthetic fingers using surface electromyogram (EMG) signals. *Expert Systems with Applications, 39*(12), 10731-10738.

[18] Englehart, K., & Hudgins, B. (2003). A robust, real-time control scheme for multifunction myoelectric control. *Biomedical Engineering, IEEE Transactions on, 50*(7), 848-854.

[19] Decision Tree Learning - http://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf

[20] Fit binary classification decision tree for multiclass classification - http://www.mathworks.com/help/stats/fitctree.html

[21] Denil, M., Matheson, D., & De Freitas, N. (2013). Narrowing the gap: Random forests in theory and in practice. *arXiv preprint arXiv*:1310.1415.

[22] Create ensemble of bagged decision trees - http://www.mathworks.com/help/stats/treebagger.html

[23] Berger, A. (1999, January). Error-correcting output coding for text classification. In *IJCAI-99: Workshop on machine learning for information filtering.*

[24] Fit multiclass models for support vector machines or other classifiers - http://www.mathworks.com/help/stats/fitcecoc.html

[25] Sutton, O. (2012). Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction. *University lectures, University of Leicester.*

[26] Fit k-nearest neighbor classifier - http://www.mathworks.com/help/stats/fitcknn.html?refresh=true

[27] Xing, K., Yang, P., Huang, J., Wang, Y., & Zhu, Q. (2014). A real-time EMG pattern recognition method for virtual myoelectric hand control. *Neurocomputing*, *136*, 345-355.

[28] Clancy, E. A., Morin, E. L., & Merletti, R. (2002). Sampling, noise-reduction and amplitude estimation issues in surface electromyography. *Journal of Electromyography and Kinesiology*, *12*(1), 1-16.

# 8 APPENDIX

## 8.1 MATLAB wrapper code overview

### C++ code, myo_mex.cpp

Since this code is the glue between the SDK bindings in C++ and the Mex file required for parsing the data in Matlab, this code will be responsible of defining the structure and the thread for streaming the data in the Matlab format for the MEX file [16].

The code will access the hub, which can receive and stream data from up to 2 Myo devices at the same time, however the second Myo can only stream EMG data, no orientation data can be accessed from it.

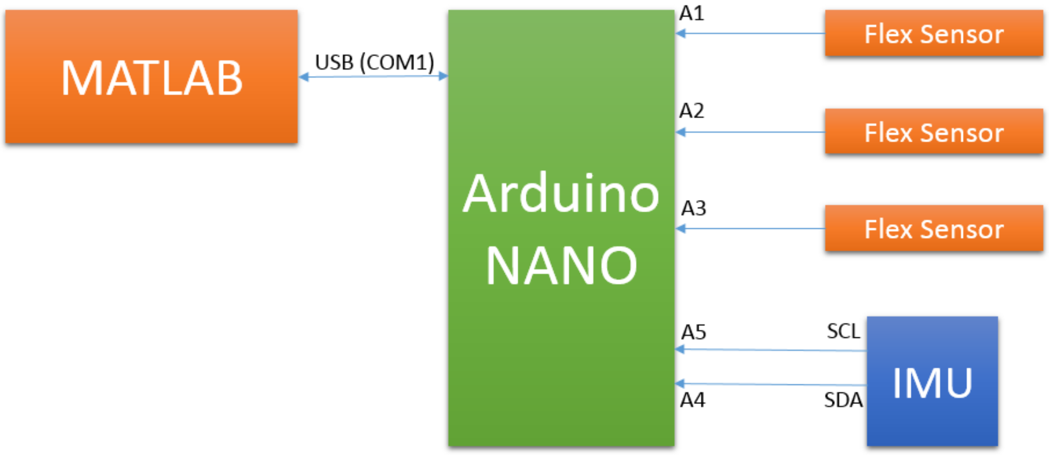
### MEX file creation and use

The creation of a *.mex* file consists in the transformation of a C++ source code into an executable file, in the code ***build_myo_mex.m*** using the **mex** Matlab function and a supported compiler for the version of the software used for the compilation, we will turn the ***myo_mex.cpp*** file into the ***myo_mex.mex64*** file.

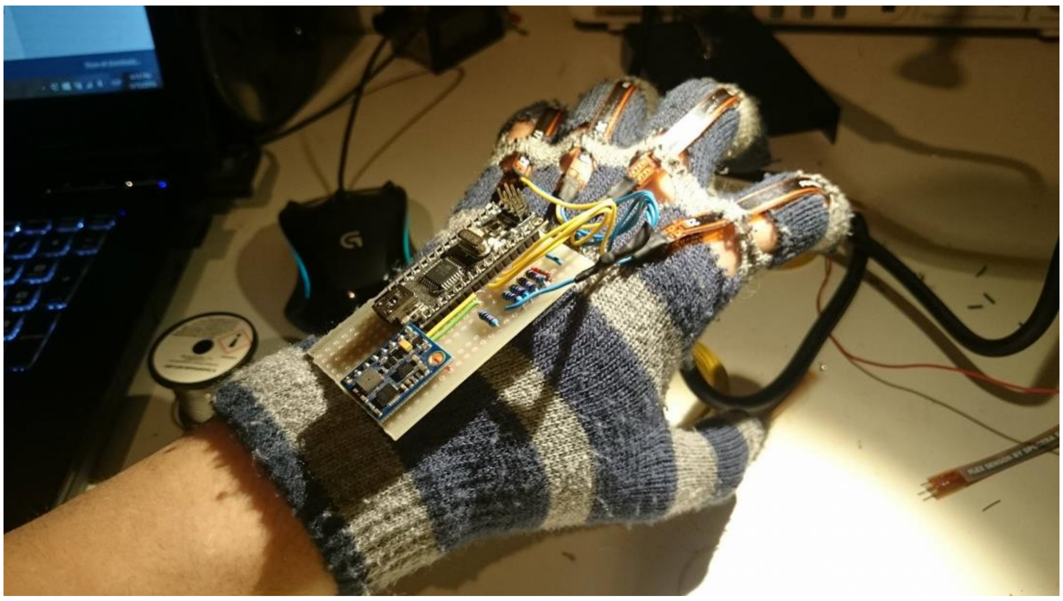
The ***MyoMex.m*** code is the bridge between the Myo SDK and the created MEX file, creating a Matlab object and calling the C++ code for initializing the data connection, data flow will start until Myo Connect is terminated. All data coming from the SDK are queued in a FIFO buffer, while this code calls and fetch new data, following the Matlab timer schedule

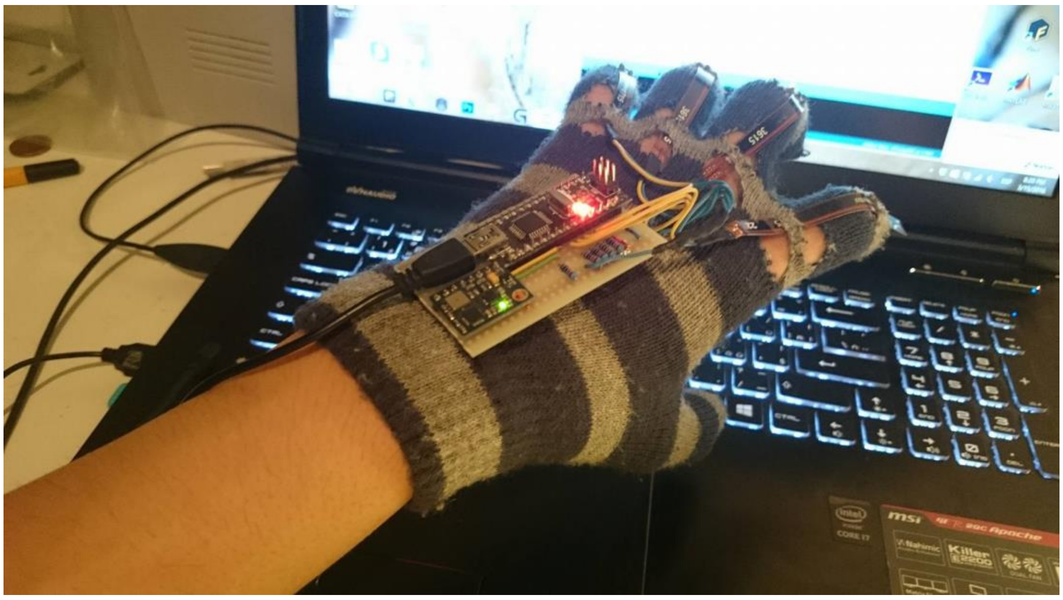
## 8.2 Glove for automated accurate labeling

Creating an external labeling system can improve drastically the accuracy of the classifier, as well as the complexity of movements that could be labeled. In order to develop this idea, a glove was designed for determining the current hand position. The system was powered by an Arduino NANO, and consisted of 5 flex sensors and a 6DOF MPU (Measured Position Unit), connected as shown in the following diagram.



Using the Arduino software, the circuit was able to determine the hand and fingers position, and stream such data through the serial port. The prototype can be appreciated in the following pictures





However, 2 major problems prevented the system to be implemented in the thesis framework. First it wasn't possible to prevent the drifting from the 6DOF MPU which increased with time, and second, the communication between Arduino and Matlab wasn't always reliable, introducing constant errors in the labeling process.

Since the glove wasn't a direct part of the thesis work, its use was discarded. Nevertheless, we present it here for the record, and hoping to encourage future readers to continue this approach.