Master's Final Thesis

Automatic Control and Robotics

EEG Based Volitional Interaction with a Robot to Dynamically Replan Trajectories

MEMOIR

Autor: Sergio Galve Ceamanos
Director: Alicia Casals Gelpí
Codirector: Raimon Jané Campos

Convocatòria: September 2018

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Abstract

Everyday robots are more involved in our daily basis and they are expected to be part of the domestic environment eventually. Assertive robots deal with this scenario providing help to people with certain disabilities, a task that requires an intuitive communication between human and robot.

One of the control methods that have become popular is the Brain Computer Interfaces (BCI), using electroencephalography (EEG) signals to read the user's intention. Commonly applied to let the user choose among several options, without further interaction once the robot starts acting.

This project explain a method to interpret the EEG signal online and use it to manipulate online the movement of a robot arm. The signal that is received comes from Motion Imagery to allow the user to communicate constantly his intention. Using Dynamic Movement Primitives (DMP) and Virtual Force Systems the stored trajectories of the robot can be modified online trying to adapt to the user's will. With this elements the trajectories are applied in a real robot arm, chequing online if all the requested goals are feasible positions for the robot.

This method intend to make more natural the collaboration with a robot in domestic tasks, where slight modifications of an action can lead into a more satisfactory interaction.

Acknowledgments

This project wouldn't have been possible without the guidance of Alicia Casals and Raimon Jané as directors of the project. Also the workmates in the laboratories where the project was developed: group CREB from ESAII and group BIOSPIN from Institut de Bioenginyeria de Catalunya (IBEC). Also to IBEC itself for granting me with the Master Programme scholarship and letting me work in this project. Finally, I would like to thank Bernhard Füerfanger for his contribution to the redaction of the project and the search of new ideas.

Contents

List of Figures

List of Tables

ETSEIB

Chapter 1

Introduction

The robotics field is booming and day by day it seems more clear that it is going to play a key role in the near future. Up to this point robots have become a common tool in industry but the techniques used are not appropriate for us on our daily basis. To reach that point, many obstacles need to be overcome before the robots can take part in more complex activities.

One of these barriers is the interaction with humans, the robots can't provide an intuitive and effective communication yet. If the way users control actions are not natural, the inclusion of these machines in domestic environments doesn't seem likely.

One of the fields that tries to include robots in a domestic environment is the assistance robotics. Robots can be seen in activities where the user interacts closely with a robot to obtain a better performance. There are many examples where this collaboration provides the user with new capabilities (for example assistive robots working with people that suffer from a certain disability).

Even if the robot actions can be useful or necessary for the users, they should adapt to them to provide a satisfactory experience. Especially in small domestic tasks, the same activity can be done in different ways and each user feels more comfortable with their own. If the robots could adapt to these small changes in each user, it would create a better experience while collaborating with an assistive robot. In this and in other fields, the need of adaption can be derived from the context, obstacles, safety, ...

This project aims to address this adaptation problem and propose a method that could obtain different variations of an action (in a domestic environment), according to the user's will. In order to obtain this natural relationship with the robot, the communication must be constant and the robot needs to adapt to the desired changes.

The communication method chosen for this project is a Brain-Computer Interface (BCI) system, a method that can link the electroencephalography (EEG) signal with the intentions of the user. This relation has been studied before [2] [3] in order to let the user, through this brain signal, choose among a few options. These proposals usually work in a "two steps" way, where the person selects the action and then the robots execute it.

Those approaches doesn't address the problem of the lack of adaptation towards the user that the robot actions usually have. This project proposes that the users can perform an online manipulation of the tasks to provide a new degree of controllability. This means that while a certain pre-programed action is being executed, it is also being manipulated to fit the person's desire.

The manipulation of a pre-loaded motion (robot's action) happens because it has been considered that certain tasks are always performed in a similar way, indistinctively to who performs it. If a generic way of doing a task could be obtained, only a small fix is required to have a satisfactory execution for every user. To meet this requirement the project aims to elaborate a control capable of an online correction to adapt the robot original trajectories.

In order to take into practice this proposal it is necessary to establish a background where it can be developed. In addition, a set of goals need to be defined in order to stablish the structure that will be followed in the report.

1.1 Background

In this subsection the aim is to illustrate the practical context, in order to set the appropriate objectives. The environment and the methods to be used bring some limitations that can be used to narrow the scope of the project and obtain a clear view of the challenges.

First of all, to bound the project the scenario that will be discussed here, is a domestic one that belongs to the Research Centre for Biomedical Engineering (CREB). The users pictured for the project would have certain physical limitations and the assistant robots are intended to help them in daily tasks.

Taking this into account, the use of an input that doesn't require motion or active interaction (the EEG signal) can be a significant improvement in the control of the robot. However, it also brings some limitations seen in the literature (ref) about what can really be interpreted in the signal.

Another boundary that will be faced is the limited reachability, which means that not all the positions are accessible for the robot. The device used in the project has a fixed base and

therefore the attached arms can only reach close points.

This is a brief introduction of the considerations taken in the beginning of the project and that lead to the solution presented in this report. The background defined can be complemented with the definition of what is considered as a domestic space and which situation is expected to take place there.

1.1.1 Proposed Scenario

As a special bound to the project and to create a common context for the future sections, here we aim to describe the scene where the developed applications would be working. In future references to the problem, a shared image will help to clarify the explanation.

The experimental scenario presents a person seated in front of a kitchen (see Figure 1.1) table where a robot arm is able to reach and manipulate the objects on it. The user is wearing a fabric helmet where a set of electrodes record the electric signal from the brain. These elements (the helmet and the robot) are both connected to the computer that controls the robot with the user's input. In addition, the kitchen contains several objects that the robot can manipulate, placed in positions known beforehand.

The robot is asked to develop a task such as a simple pick and place exercise. During the movement, the patient should try to modify the robot motion (through the EEG signal). To achieve so, there must be a shared code between the user and the control, where thinking about moving a certain limb pushes the robot arm trajectory in one direction.

This means that in an ideal case it could be easily relatable to move the right arm when the intention is to displace the trajectory to the right. This requires that the disposition of this environment provides a clear view to the user about the actions that the robot is carrying out.

1.2 Objectives

After setting the project scope and the background where it will be developed, this subsection will define a set of objectives that should be met. The objectives are extracted from the presented scene and the requirements that are necessary to provide a useful tool for the user.

The main goal is to provide a control system that can manipulate online trajectories with the EEG input from the user. This is a very wide objective that include all the activities, but for a better guidance it can be divided in several subobjectives:

- EEG online interpretation: The EEG signal represents in a way the user's brain state

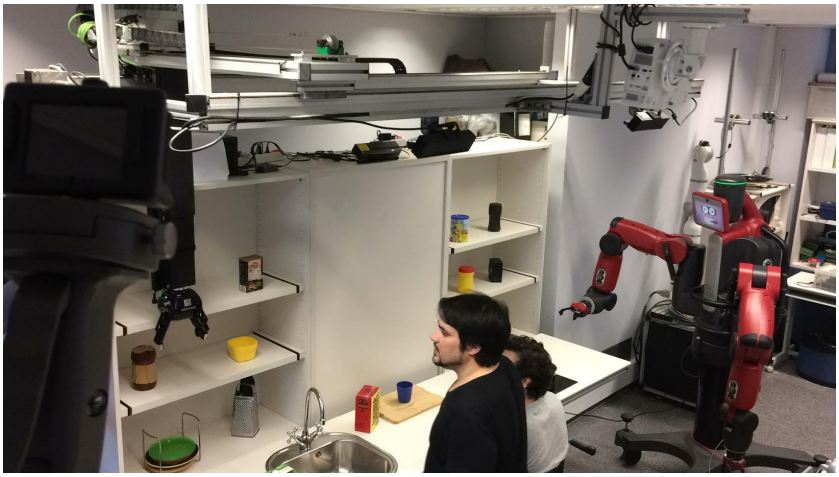


Figure 1.1: Robotic Kitchen in CREB's laboratory

and it changes through time according to his will. That's why during the collaboration with the robot, certain states can be interpreted to modify the movement of the robot. In order to be a useful tool for communication with the robot, the interpretation of the EEG signal has to be fast and accurate.

- Fast trajectory manipulation: If the movement of the robot is defined by trajectories, these paths must be updated constantly according to the user's will. To handle these changes based on an original trajectory a method that can generate fast new paths is necessary. In addition, these trajectories generated must be feasible, the robot must be able to reach all the points in the requested trajectory.

- Robot trajectory execution: Apart from defining the path, the execution of the trajectory by the robot has to be defined. This implies managing the robot hardware and integrate the previous subobjectives in order to produce a fluid motion.

The subobjectives discretize the work to be done and help to visualize the challenges in each part. In the last instance, these objectives will be used to evaluate the result and the performance, being easier to evaluate the sub-goals. In the same way in the following sections the subobjectives can be divided again to obtain requisites that show what must be evaluated.

1.3 Structure

The problem discussed can be understood from different points of view, the technical parts are the objectives of the project. Therefore, the most relevant perspectives can be identified with the core elements that must be handled:

- EEG signals representing the will of the user
- Trajectories defining the tasks
- Orders to the robot actuators

According to the modular nature of the project, the work was divided in two phases that tackle the interpretation of the input and the action. To keep a coherent structure the report is divided in two parts: one for the processing and interpretation of the EEG signals and one for the robot control.

In the second part it is also possible to distinguish between two different sub-tasks that are closely related. The first would be the manipulation of the trajectories that guide the robot arm, to fit the user's intentions. The second would be the execution of the motion, in other words, to integrate the previous results with the hardware control. These two tasks interact between them and complement each other, as a result they will be studied in the same section.

Also, between the two main blocks (the EEG signal interpretation and the robot manipulation) there must be a connection and a certain feedback. Since they act together it is also necessary that they take each other into account to be able to perform a final implementation.

To illustrate the process followed and the distribution of the work through the duration of the project, a Gant diagram is shown in Figure 1.2. In it, the duration of every element is estimated and presents discarded work paths that weren't integrated in the final proposal. Also the chart shows clearly the links between tasks and between different blocks which help to understand the structure chosen.

The Figure 1.2 shows the proposed calendar starting on the 16^{th} of February until the end of July. We coded the chart in five colours:

- Blue: technical activities.
- Purple: study of the state of the art.
- Orange: demos and works towards a practical application.

- Yellow: tasks to improve or add interesting features.
- Green: documents.

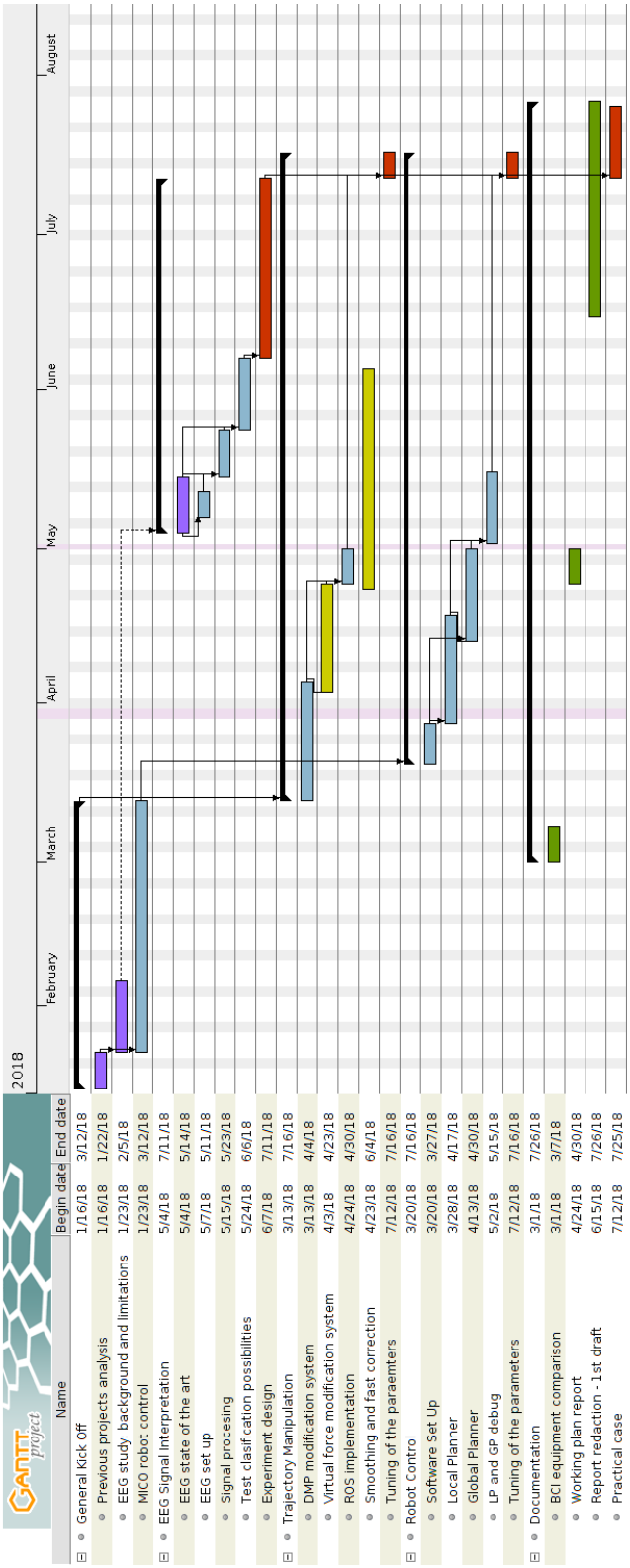


Figure 1.2: Gantt chart with the proposed working plan, obtained with the program GantTProject

Chapter 2

EEG Signal Interpretation

2.1 Introduction

This block aims to explain how the communication between the user and the robot is done. Since the tasks being carried out by the robot intend to adapt to each case from the user perspective, his criteria must determine how to adapt the movements.

The EEG signals have been used to interpret many different responses in the brain [4] [3] and when it comes to control, the motion intention has been recently one of the most successful cases [5].

Since our scenario presents a clear view to the user, a movement from the right or the left hand would be an example of intuitive control command (to move the robot to the ight or the left). Other movements can also be useful or easier to interpret such as upper versus lower limbs.

2.1.1 Objective

The objective of this part is to be able to distinguish online a set of two or four easy movement intentions through the corresponding EEG signals. Also, a resting state must be identified when the user doesn't want to modify the action.

Going deeper in what will be carried out in the following lines, we can divide this goal into smaller requirements. We can define them by considering the different perspectives we evaluate in this part:

- Intuitive Control Protocol: One of the requisites to make this project interesting is that the user can eventually manipulate the robot movement. This can only happen if the

interaction with the robot is natural and simple.

- Time Series Analysis: It's always a challenge to create a mechanism able to identify the state of a given time series data. Especially in this case, where the signal is really noisy and the states can change fast, the system should be robust enough.

- Minimum Time Processing: Another requirement given from the context of a real activity being carried out by the robot, is time. The word "online" from the project's title has a big weight in this point because it means to react as fast as it is possible to the user's intention. Therefore, the first step to achieve this condition is to have a fast EEG signal processing.

2.2 Theoretical Background

Before introducing the experimentation carried out, it is necessary to contextualize the EEG signal processing field and the nature of these signals. There are several concepts related to the nature of these brain signals and the strategies to use them, that are key to justify the selected approach.

2.2.1 Brain-Computer Interface Using EEG Signals

A Brain-Computer Interface (BCI) [6] is a platform for communication between a human being and a machine that is based on brain signals, bypassing the need of neuromuscular involvement. There are both invasive and non-invasive methods of monitoring the brain's neural activity, whereby this work focuses on non-invasive electroencephalography (EEG)-based interfaces. A BCI typically aims to estimate its user's mental state or intent from the monitored signals and translate it into a physical action, like controlling a robot arm.

BCIs can be categorized into two types based on the monitored EEG features, event-related or oscillatory brain activity. Event-related potential (ERP) based BCIs, also called reactive BCIs, rely on external time-locked stimuli. The stimuli provoke temporally correlated waveforms in the EEG signal that are well-characterized. ERP-based BCIs in general are relatively robust across subjects when compared to BCIs based on oscillatory processes [7].

BCIs based on oscillatory processes are also called active BCIs, because they do not rely on external stimuli, are not time-locked, and are able to translate voluntary, self-paced mental tasks into physical control.

In this project, the user can translate voluntary between different mind states and use his own EEG to have a physical control, however he is also influenced by external stimuli. This external influence is given by the robot performance and the user perception of it. In other words, the user might want to modify the movement if it doesn't please him.

2.2.1.1 Sensory Motor Rhythms and Motor Imagery

The brain generates constantly rhythmic or repetitive patterns of neural activity, signals that can be identified with EEG measures. Different areas of the brain manage different tasks, in this project the interest is focused on the brain part in charge of the motion, the motor cortex (shown in Figure 2.1). In this region takes place the planning of the motion that a person intends to do, takes place.

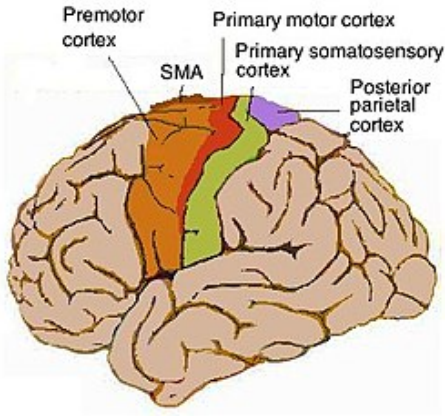
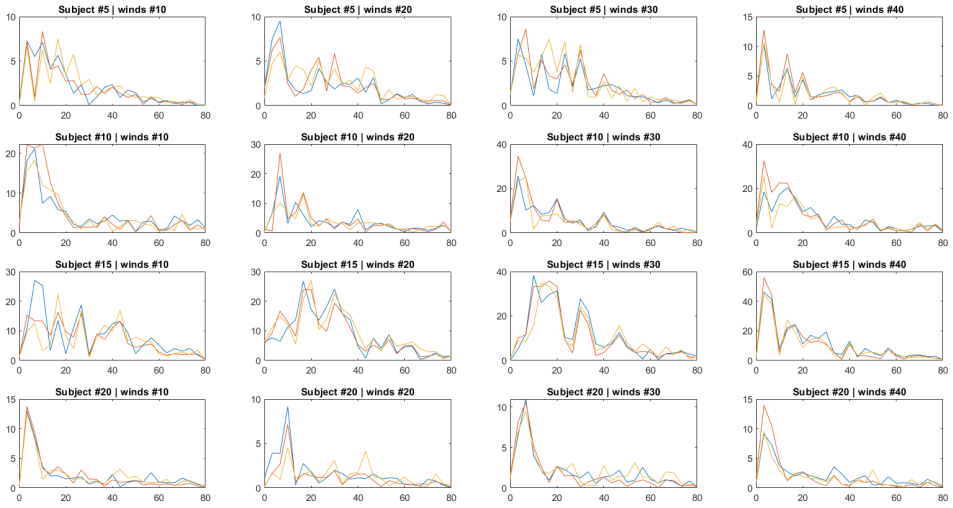


Figure 2.1: Representation of the brain and the motor cortex obtained from Wikipedia entry for Motor Cortex.

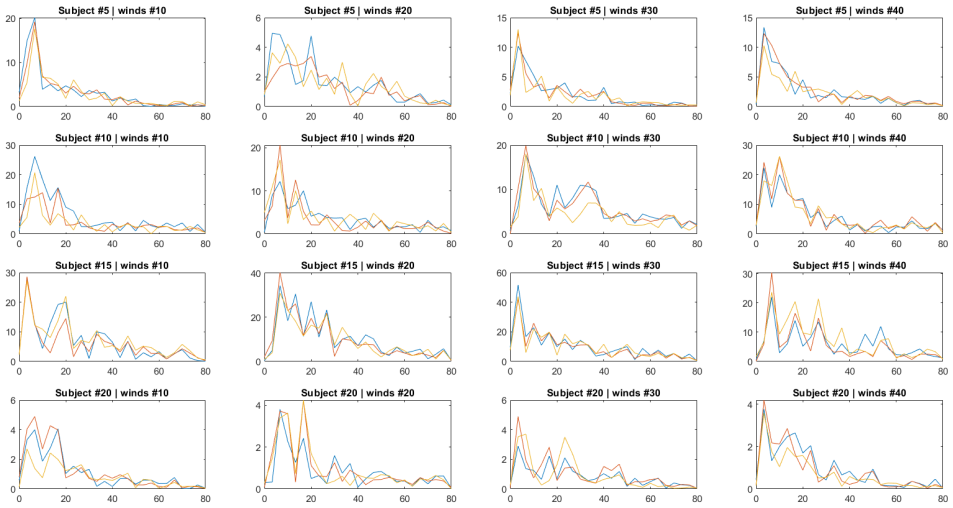
As a consequence of this process the brain generates in this region a pattern that are identified as sensory motor rhythms (SMR) [4]. SMR is a common control signal for active BCIs because of their stable patterns in the noisy EEG recordings. The brain is constantly active and it generates a series of signals, commonly categorized according to their frequency range (alpha waves, beta waves, mu waves, etc.). Characteristics of SMR can be described by the desynchronization of these oscillatory activities over the sensorimotor cortex, specially in alpha- (8 Hz - 13 Hz) and betha- (12 Hz - 25 Hz) frequency bands during a physical or imagined movement of limbs.

SMR can be induced by the mental imagination of a physical task, even if the movement is not carried out. This mental strategy is also referred to as motor imagery (MI) [2]. On a

neurophysiological level, similar brain regions are activated during motor execution and motor imagery, however, the performance is blocked. Studies based on unctional magnetic resonance imaging showed similar activation patterns during motor imagery and actual movement execution. In the same way, the MI is also noticed with the EEG equipment (as shown in Figure 2.2) and used in BCI as a communication mechanism.



(a)



(b)

Figure 2.2: Both images show four different subjects' EEG signal divided in frequency components. The patients are closing their right fist in different time instants (winds). In the image (a) they perform the real movement meanwhile in (b) is a MI exercise. The X axis represents frequency (Hz) and the Y axis voltage (μV). The data from the different subjects was created and contributed to PhysioNet by the developers of the BCI2000 instrumentation system

2.2.2 EEG Data Recording

The brain activity generates an electric signal that can be recorded with electrodes in the process of EEG. This data can be used for identifying which process are taking place in the brain at each moment.

Before providing any control over the robot to the user, the EEG signal require a certain study to find which classifying options fit to the signals. In order to interpret them correctly, the collection of data is a crucial step in order to develop a good classifier. This gathering of recorded data has been standardized in the research community and there are several procedures for study of MI.

2.2.2.1 Recording Sequence

In order to label properly all the samples obtained the procedure consist of a video sequence that guides the user through "rest and action" states. This sequence is repeated numerous times to increment the dataset as much as it is possible or necessary.

Obviously, the time available for measuring is usually a limiting factor since each sequence takes in our case (and in most of the literature explored [6]) 10 seconds. In addition, this is an exhausting process and it is not possible to make long recording sessions with the same user.

While making the recordings it is necessary to separate the action states from the rest states, obtaining a clear sample of each case is very important towards the performance of the future classification of these signals. In the explored cases the recordings were structured in several phases:

- Rest Phase: The first sight presents a clear screen with no indication of the future action. For around 3 seconds the subject is expected to wait, allowing to label the rest state in this time.

- Set Phase: To provide a transition a first signal (visual or auditive) is presented, followed by a second of inactivity before the cue is presented. This cue was explained beforehand and associated to a motion that need to be recorded. A beep sound is usually played to indicate the beginning of the action Phase, with a random wait time after the second that is given to get ready.

- Action Phase: The sound commands the user to initiate the action, during a period between 3 and 5 seconds (the variability is added in some cases, to avoid that the user

acts due to habit after several tests). The action ends with another beep sound and during this process the eyes can be closed to avoid distractions. In this project's context, the eyes closed can make a clear signal for the database but doesn't fit the real scenario where the user must pay attention to the robot.

- End Phase: Finally, another transition phase, with the same screen display as the rest phase. This phase allows to prepare several exercises in a row, after this phase the whole cycle can start over again.

These exercises are recorded without pause during around two to five minutes and pauses between them. Even though, the process is very exhausting for the user and after several tries the obtained measures can be unrealistic. This sequence has been adapted in several ways according to the case, preserving all of them the structure presented here. The structure used in this project also follows this sequence and the video was presented to guide the recording session. To show an example of this videos the following link presents the video used in some of the recording carried out in the project:

`https://youtu.be/M57xFImcZIE`

2.2.2.2 Electrode Placement

For the positioning of the EEG electrodes on the scalp an internationally recognized topology called the 10-20 EEG system is widely used. It was developed to ensure reproducibility and comparability of results by standardizing the electrode positions.

The system is based on the relationship between the location of an electrode and the underlying area of the cerebral cortex. The "10" and "20" refer to the fact that the actual distances between adjacent electrodes are either 10% or 20% of the total front-back or right-left distance of the skull (ref).

Previous studies show that electrode positions C3, Cz and C4 (shown in Figure 2.3) are suitable for recording characteristic motor imagery signals as they are directly covering part of the sensorimotor cortex. In this project the measures were taken only in these three positions, each of the colours used for the graph lines in Figure 2.2 belong to a different electrode. These sensors are placed in a fabric helmet in order to have reproducible measures (a more detailed procedure for managing the electrode placement is found in the 10/20 System Positioning Manual [8]).

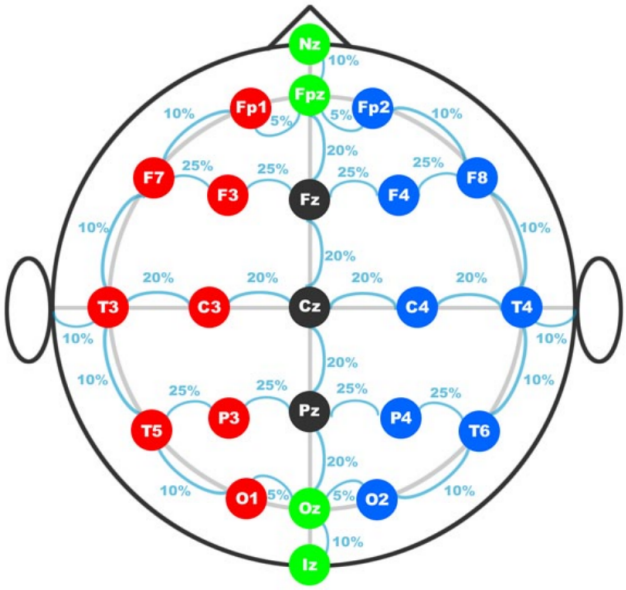


Figure 2.3: Representation of the skull division according the 10-20 EEG system

2.2.2.3 Preprocessing of the EEG Signal

This is one of the most standardized procedures, since this is a very noisy signal this filtering process is the first step before any type of interpretation. It also includes a segmentation so we can talk about samples in future sections.

The electric signal produced by the brain is very weak compared to other noise or interference aquired in the lab. Also, we must remember that we are interested in specific frequency bands in the brain, signals that change constantly. The registration of the signal is done at 160 Hz as in other references found, being able to compare with previous results in similar experiments (Phisionet database [9])

A notch filter (used to remove noise from a signal, in a very narrow frequency band) around 50 Hz is applied to remove the noisy coming from the electric supply of the equipment. In Europe the mains frequency is 50 Hz and without taking it into account the brain signal is indistinguishable.

As mentioned before we are mostly interested in the alpha and beta signals range (their frequency range), that's why we must apply a bandpass filter between 3 and 80 Hz. The energy outside this frequency bands might come from other sources of noisy and we can discard it.

The way we will split the data is in small windows that will be labelled, using times for our

purposes between 0.2 and 0.5 seconds. Now, each sample is normalized in order to avoid the detection problems due to different intensities measured. Each sample then will have less than a second of recording, but it's enough to generate detections in this type of signals.

2.3 Proposal

This section will explain the initial proposal and the necessary adaptation to fit the requirements. Most of the practical exercises consisted on testing the following proposals and study the impact of the used elements. Another problem addressed here is the lack of a wide database. As seen previously (section 2.2.2.1), the process to obtain data is time consuming and would require many volunteers. Since recording a whole database is very expensive in time and resources, an external database was used and processed as if they were our own samples.

2.3.1 EEG Database

The first problem encountered was the lack of data and the uncertainty of which data would be more interesting to record. To maximize the efficiency of our experiments and finding global solutions for different users, PhisioNet's Imagery Dataset [10] was used

This database consists of 109 subjects that carried out 14 sequences of 10 seconds recorded at 160 samples per second. For each subject we have 28 files, 14 represent the EEG signal (in the European standard format ".edf") and 14 event files that label the recorded brain signals.

Among these 14 cases we can identify two rest examples (eyes closed or open) and four tasks repeated three times. These tasks represent motor actions, and therefore will generate a stronger signal in the motor cortex (section 2.2.1.1):

- Opening and closing left or right hand, real and imagined movement.
- Opening and closing both fists or both feet, real and imagined movement.

Each file was loaded, pre-processed and labelled in order to try different approaches that show the best way to work with these signals.

2.3.2 Classification Methods

Based on the previous works in this field [7] [4] [11], we tested two different approaches to classify the samples pre-processed.

On the one hand, we know that there must be a relevant difference in the energy associated to the frequencies between 3 and 80 Hz. Especially in the alpha and beta ranges there should be a big energy reduction when a movement is being imagined (the latent signals disappear). Even the way they are distributed, in other way the "shape" of the Fast Fourier Transform (FFT) of the EEG time signal seen in Figure 2.4.

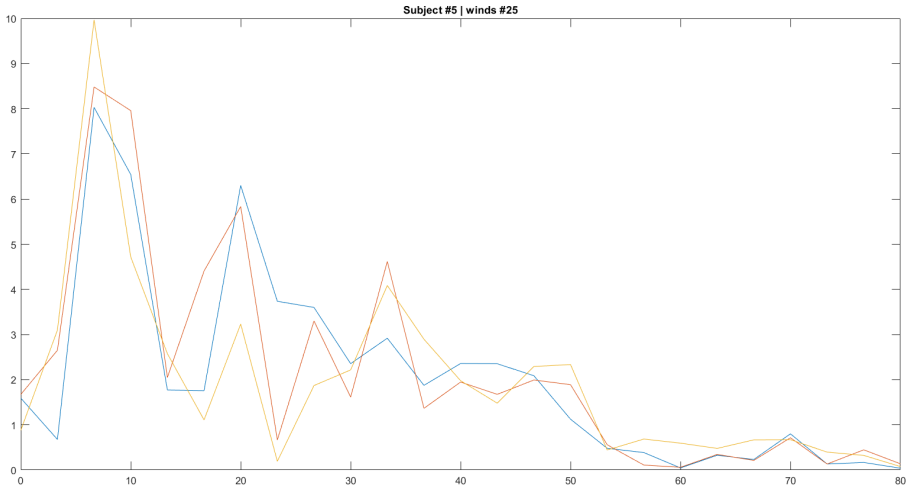


Figure 2.4: EEG signal after FFT of a person moving his right hand, using the channels C3, Cz and C4 from PhisioNet database.

On the other hand, the time series can be relevant itself. The state detection from signals that have a periodic behaviour, like audio or speech, has been very successful and in the interpretation of brain signals has been also an interesting option. The EEG signal time sequence can also be very relevant (as seen in Figure 2.5) to identify the activity peak that the motor cortex activation can generate while imagining a motion.

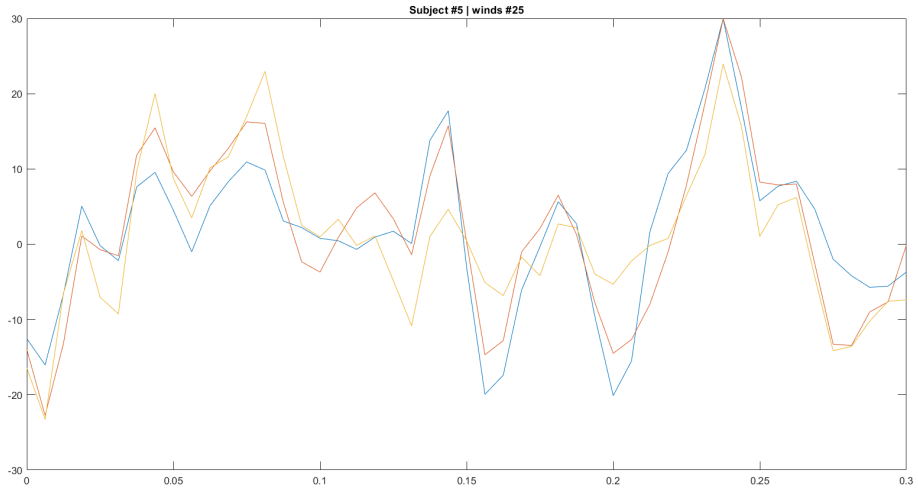


Figure 2.5: EEG signal in a 0.3 seconds window of a person moving his right hand, using the channels C3, Cz and C4 from PhisioNet Database

Taking this information into consideration while choosing the classification methods that were taken into practice, the following two approaches are presented.

2.3.2.1 Feature Extraction

The methods based on feature extraction can define a certain state as a region in a high dimensional space. If we obtain points that are labelled belonging to a certain state we can estimate which would be the hypothetical region.

The coordinates that define the positions of the points are just the features that were selected to distinguish between classes or states. If more features are used to define the difference between classes is because these categories are more complex to be defined.

The features of a small time window are less clear in the time domain than in the frequency one (the). In the frequency domain we can look at the energy associated to frequency bands (the integral of the lines in Figure 2.4), being more robust among different instants in the imagined motion periods.

The best approach seems to be looking for the biggest energy change among small frequency bands (2-6 Hz width). By comparing between all the samples, we find where the biggest standard deviations are in terms of energy (after normalizing the data).

For each set of tasks to differentiate we run this calculation again, and each time we take only

the N bigger differences in each channel, making a total of 3xN frequency bands or 3xN features. In the following table the most relevant frequencies are presented, matching consistently the expected result of being the alpha and beta frequencies (Table 2.3.2.1).

Maximum Differences	C3 Receptor		Cz Receptor		Receptor C4	
	Band	Lower Band Frequency (Hz)	Band	Lower Band Frequency (Hz)	Band	Lower Band Frequency (Hz)
1º	2	6	2	6	2	6
2º	5	15	4	12	3	9
3º	3	9	9	27	4	12
4º	9	27	10	30	9	27
5º	6	18	8	24	5	15
6º	10	30	18	55	6	18

Table 2.1: Among the frequency 25 bands studied from MI contrasted towards resting states, the bigger differences were found in the presented bands. Each electrode recordings were analyzed separately, using the PhisioNet Database

With this computation the more relevant channels were found in each receptor, but the results of each channel haven't been compared. The features extracted are the one containing clearer differences between brain states, but before taking the data for any classification the Principal Component Analysis (PCA) for cross comparison among channels, is applied. From this method we obtain a new set of values defined from the combination of the previous ones, from this set we choose those who represent more diversity in the data.

Now that we have a set of features where we know which ones represent best the diversity of classes among samples we need classification tools that use these features. The methods finally used for classification were Super Vector Machines (SVM), Neural Nets (NN) and Regressions.

2.3.2.2 Deep Learning

As one of the most popular tools lately, Deep Learning (DL) [11] techniques are able to avoid the feature selection. In these nets the multiple layers take care of the feature extraction in an optimal way due to the feedback obtained with the learning. It also means that we need a big database to keep training until there is a certain convergence.

The raw data (after the pre-processing) can be fed directly to the classifiers and with the correct labelling. With the appropriate design of the layers that compose the DL classifier and the right backpropagation algorithm (the learning method), the classifier can adapt to almost any level of complexity in the data.

Among the many options available inside the DL classifiers we found specially interesting two types:

- Convolutional Neural Nets: The idea behind convolutional layers assumes that a locally learned feature for a given input (typically two-dimensional, like images) should also be useful in other regions of that same input, e.g., edge detector that was found useful in some part of an input image should also prove useful in other parts of the image as a general feature extraction stage. The problem of applying this method here is the low number of channels that make our "graph image", composed by the 3 channels through time. Still the method can be useful to find correlations among the different channels.
- Long Short-Term Memory Networks (LSTM network) (ref): This structure is a recurrent neural net, where each time step is fed with previous predictions or values. This means that at each point the previous situation of the system is considered, which is extremely useful in this case of time series. Analysing each channel separately, the LSTM network can find out to which class the signal belongs.

As mentioned earlier, the time series can provide this raw information, but it is still interesting to test with the frequency domain data. The information in the frequency domain is still more robust through time and, most likely, easier to process.

2.4 Implementation and Measures

The implementation considers the features in the EEG signal and the challenges that it supposes. The search of original approaches aims to tackle the problem, based on the current state of the art [7]. The solution should present a classification as robust as possible to be part of the context of the daily and intuitive help mentioned before.

2.4.1 Global Models with Database

One of the reasons for working with the PhisioNet [9] [6] database [10] was to learn which combinations and elements work better before doing new measures. But, another reason was to work with different subjects and use all their samples together to try to find global classifiers that could work with different users.

Usually the differences in the EEG signals among users is big enough to prefer classifiers trained only with one user. However, if the number of samples is not enough or if we want to interact directly with new users, a general model is a good approach. This has been a challenging point in EEG signal classification, where obtaining classifiers for several users was barely tested or successful (ref).

Also, in this case obtaining a global model is rare, but since the data is such a restrictive factor testing with this external database and looking for global models can provide useful feedback. However, it is important to consider that there are guaranteed differences when comparing to new recorded data. This is a consequence of using different equipment, with different users and in different conditions (the EEG is a very sensitive measurement).

Another consideration to work with this external database, is to make sure that it is well suited for these types of classification. First of all, in any case we need to balance the amount of samples per class. In the database more than the 50% of samples belong to rest states, and that lead classifiers to avoid mistakes in this class rather than the others (to reduce the global error in the training phase). To dodge it, the number of samples in each class must be equal to the amount of samples in the smallest class (the class with less samples). The solution has been to eliminate some samples randomly in each class that need to be reduced.

From the available electrode recordings we took only those that could be reproduced with future measures in the laboratory. The equipment available can only record three channels, since the target is the activity in the motor cortex these channels will be placed in the positions C3, Cz and C4. Therefore, from the database only the recordings taken in those positions are analyzed and in a future work more channels can be including to improve the classifications.

2.4.2 Software Environment

In the other sections the work environment for the robot controllers is discussed, but the main purpose of this section is to send a certain value (indicating the state) repetitively. Communication is implemented with the standard middleware ROS, which is compatible to a wide range of platforms.

One of them is Matlab©, an environment where we can access to many toolboxes designed for the intended purposes. The Biomedical Signal Processing and Interpretation (BIOSPIN) group, where this part was developed, uses it and the measurement equipment's are prepared to work with it. These reasons make it an ideal tool to develop the required scripts. Therefore, all the computations concerning the EEG signal will be done in Matlab with special use of the Statistics and Machine Learning Toolbox [12] to work with the mentioned classifiers.

All the code developed and the results obtained are available online in the following link:

`https://drive.google.com/open?id=1xU7I-a4FvvlVcrVIWy-YDzHMYkCk5V6I`

2.4.3 Implementation Problems and Solutions

Keeping feature selection in mind, among all availablemethods of classification, the SVMs provide the best results. The SVM outstands in those cases where the complexity of the regions, which define classes, to identify is higher than the one given from the number of inputs. With this circumstances we can use kernels that help to adapt the classifier.

However, there were drawbacks in terms of lack of accuracy to distinguish between action classes. In addition, the toolbox struggled when the amount of data was too high (more than 15000 samples lead to a crash of the system in many cases). The experimental results were obtained by combining the data from 47 subjects and reducing the number of samples randomly.

The results obtained with SVM classifiers, using the features from the frequency domain, vary depending mainly in the kernel used and in the type of motion recorded. The Gaussian kernels showed a better performance every time over the polynomial ones. However, the difference between activities showed,if the motion is real instead of MI the results are better. When the motion was real the performance peaked in 65% meanwhile the best performance obtained with the MI was a 57% accuracy. The best results were using the exercises that involved legs and fists, being less determinant (the accuracy was around 3% lower).

An example of the results of these classifiers is presented with a Gaussian kernel, using data from the left and right hand exercises. Using data from MI cases of opening and closing one of the fists combined with samples of the resting state (a total of 10920 samples), the accuracy was 60.8%. From the PCA analysis the features were reduced to 13 from the initial 18 and the confusion matrix in Figure 2.6 shows the type of errors obtained.

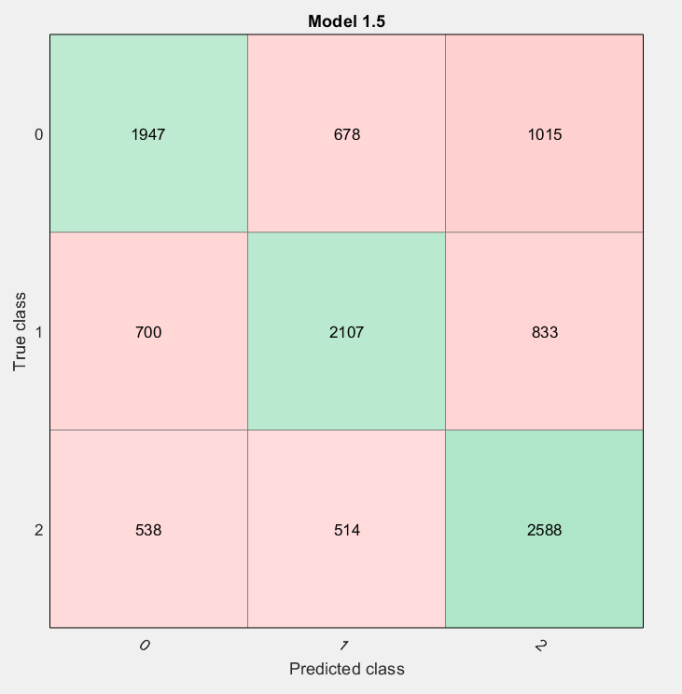


Figure 2.6: Confusion Matrix obtained in Matlab's Classification Learner tool. The model used 10920 samples, classifying them in rest (0), left hand (1) and right hand (2).

On the other type of classifiers, the LSTM network outperform the Convolutional networks with the appropriate tuning, using a total of more than 45000 samples. The selection of the right parameters was crucial to overcome problems of local minima that this classifiers present.

The parameters used to define the network determine its complexity and the way it learns the difference between classes. This represents a trade-off that:

- Complexity: As the net gets bigger, with more layers, it can deal with more complex classes. With this improvement also comes a much harder training and more complicated obtaining of valid results. This is highly correlated to the input data, in those cases where the input is much bigger it might provide more information that define the differences among classes.

- Learning: The net begins from random initialization values (weights and bias in the neural net are the parameters, of each neuron, to be trained) that step by step are transformed into the optimal values that lead to the correct classification. There are different algorithms to update it according to the reward (in this case Matlab provide three variations of the stochastic gradient descent), but in general the trade off is related

to how "aggressive" is this update. If the changes in the values are higher then the net evolves faster, but it is also harder to converge.

This networks are a managed as a data format in Matlab, making the tuning to be limit to a few values that determine the size and the shape of the different layers. The LSTM layer pass its output through a fully connected layer, a softmax layer and finally a classification layer [12] [11] that returns the label of the predicted class for each sample.

The training of these layers is monitored in the Matlab's interface, showing the evolution of the performance as it takes data in. This graphs are shown in the Figure 2.7, based on the best performance obtained.

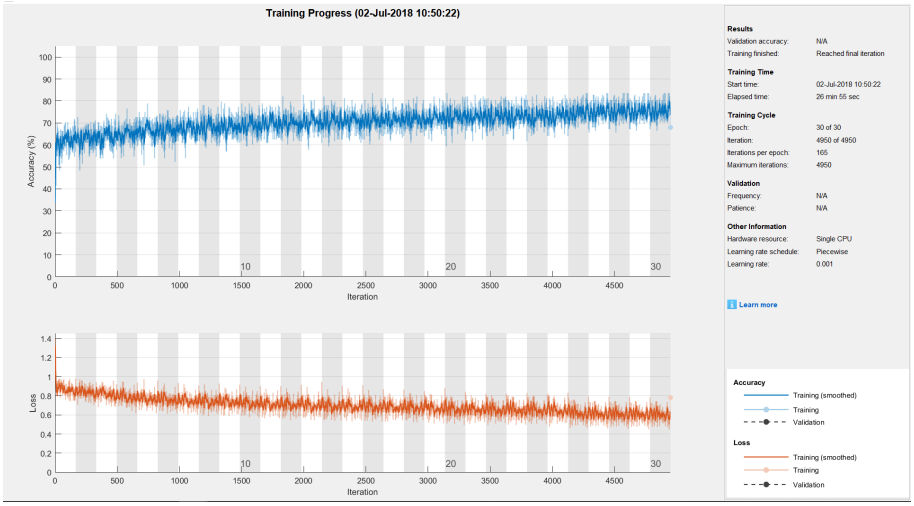


Figure 2.7: TrainNetwork Matlab's interface showing the complete training of a LSTM network. The data used consist on near 35000 samples from a legs-fists real motion.

One of the reasons of the local minima problem can be in the parameters, but some extra treatment in the data before the training part could be beneficial. One of the options that work particularly well was to change to the frequency domain as an input in this DL methods. The other approach was to augment the database.

Considering the reduced number of channels, a way to augment this data is to generate "artificial" channels. The combination of the already existing data at each timestep can help the network to perceive some new features. Mostly the augmented channels were polynomial combinations, obtaining that the simple cases in Table (ref) worked well and help to improve the performance.

In this case also the type of action used was relevant in the final performance of the classifiers.

The conclusions obtained from this parameter were the same as the mentioned before with the SVM classifiers. However, the options in this analysis was limited to the options in the database, the record of new data should consider new types of actions.

Finally we include the Table 2.4.3 to show the different test carried out that were registered. The different options vary the main parameters of sample's window size, depth of the NNs, number of samples and the internal parameters of the nets (described in the Matlab's website [12]).

Test	Name	Description	Training	Validation	Window (s)	Dataset	Tests
1	LSTM_1	layers_1 (200 hidden) and options_2 (0,01 ILR | 0,8 GT | adam)	65,00%			25 subjects	3 states: both feet or fists
2	LSTM_2	layers_1 (300 hidden) and options_2 (0,01 ILR | 0,8 GT | adam)	60,00%	54,50%	0,35	18 subjects	3 states: both feet or fists
3	LSTM_3	layers_1 (300 hidden) and options_2 (0,01 ILR | 0,8 GT |adam)	65,00%	57,30%	0,20	18 subjects	3 states: left or right fist
4	LSTM_4	layers_1 (300 hidden) and options_2 (0,01 ILR| 085 GT | adam)	60-65%	64,40%	0,30	18 subjects	3 states: left or right fist
5	LSTM_5	layers_1 (300 hidden) and options_2 (0,01 ILR| 0,9 GT | adam)	70,00%	65,00%	0,30	25 subjects	3 states: both feet or fists
6	LSTM_6	layers_1 (400 hidden) and options_2 (0,01 ILR| 0,9 GT |adam)	77,00%	74,28%	0,35	25 subjects	3 states: both feet or fists
7	LSTM_7	bi-lstm layers_1 (500 hidden) and options_2 (0,01 ILR| 0,9 GT| rmsprop)	65,00%	61,00%	0,30	35 subjects	3 states: both feet or fists
8	LSTM_8	layers_1 (500 hidden) and options_2 (0,01 ILR| 0,9 GT | adam)	72,00%	68,19%	0,30	35 subjects	3 states: both feet or fists
9	CNN_1	**using frequency** -- 2 convolutional layers layers	70,00%	64,85%	0,30	15 subjects	3 states: both feet or fists
10	CNN_2	**using time instead frequency** -- 2 convolutional layers layers	65,00%	61,90%	0,30	35 subjects	3 states: both feet or fists
11	CNN_3	**using frequency** -- 2 convolutional layers layers, intermediate NN	65,00%	63,40%	0,30	35 subjects	3 states: both feet or fists
12	CNN_4	**using frequency** -- 1 convolutional layers layers, intermediate NN	65,00%	64,00%	0,30	35 subjects	3 states: both feet or fists
13	CNN_5	**using frequency** -- 1 convolutional layers layers, intermediate NN	67,00%	66,00%	0,40	35 subjects	3 states: both feet or fists
17	LSTM_9	layers_1 (250 hidden + dropout 0,05) and options_2 (0,03 ILR| 0,95 GT|sgdm)	75,00%	71,77%	0,35	47 subjects	3 states: both feet or fists
20	LSTM_12	**Augmented Data** -- layers_1 (250 hidden + dropout 0,05) and options_2 (0,03 ILR| 0,95 GT|sgdm)	75,00%	73,75%	0,35	47 subjects	3 states: both feet or fists

Table 2.2: Record of the different tests run with the DL classifiers, identifying the parameters tuned among trials. For the validation a 5% (it represents near 2000 samples) of the data was taken away from the training part and used to test the resulting classifier.

2.4.4 New Data Gathering

Apart from the database, it is necessary to obtain new samples with the EEG equipment in the BIOSPIN's laboratory. The new data will be used to test the classifiers obtained from the database, as an exercise to test the viability of a global classifier.

The recording process asks the users to do several movements according to a video with visual and sound signals to trigger the action phases. The problem has been mainly the difficulty to fix the helmet properly, it requires gel for a proper connectivity and it should be as close as possible to the skin. In this project the data recorded is all from one user (Sergio Galve as the author of this report) in 14 sessions of 2 minutes (the recording set can be seen in Figure 2.8).

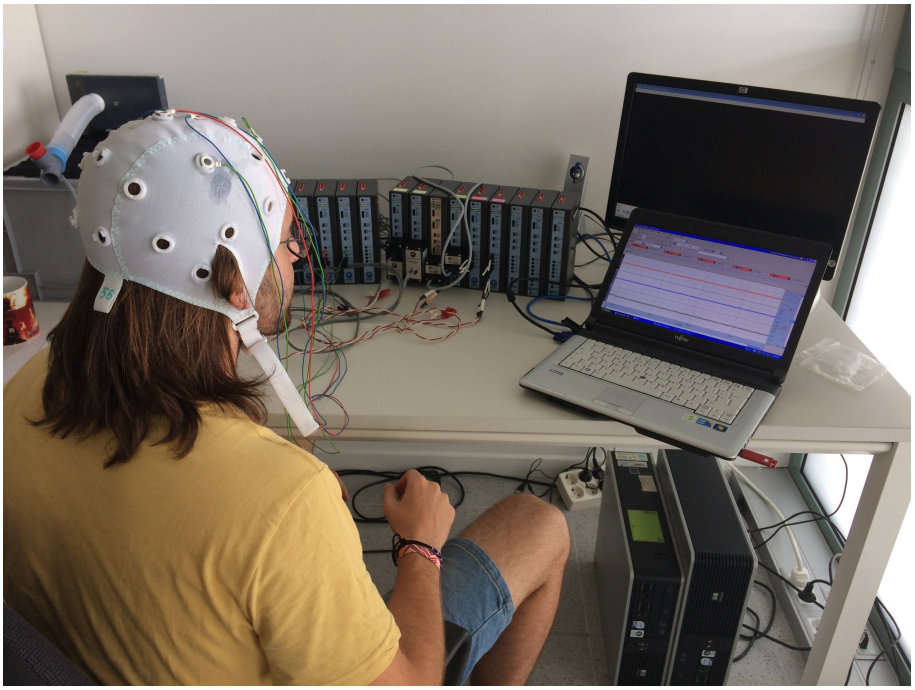


Figure 2.8: BIOSPIN's laboratory with the eeg equipment from the company G.tec and amplifiers from BIOPAC Systems.

These 14 sessions were divided between 4 types of motion:

- Circles and Squares: since the MI is based on the motion planning that takes place in the head, using shapes that require the user to focus might make easier to visualize the MI.
- Left and Right fist closing: as in the database we record the motion of closing the fists, using both hands alternatively as the video indicates.
- Left and Right hands waving: similar to the previous one but less exhausting and easier to visualize for a longer period.
- Close Fists and Raise Legs: this exercise is again as the ones seen in the database, using both legs or both fists at the same time.

For each of one of those groups we can use a total of 639 samples where almost the 50% correspond to rest states (leaving an effective database of 447 samples per group). The problem in the recorded data is the accuracy obtained from one of the electrodes, that due to a bad contact or malfunction, couldn't provide a reliable signal in this case (as it can be seen Figure 2.9, where the signal have a very reduced intensity and uneven among channels). At different

time steps the intensity of the electrodes was varying and it represents a handicap for the proposal of the global classifier, based on a database that doesn't suffer from those problem.

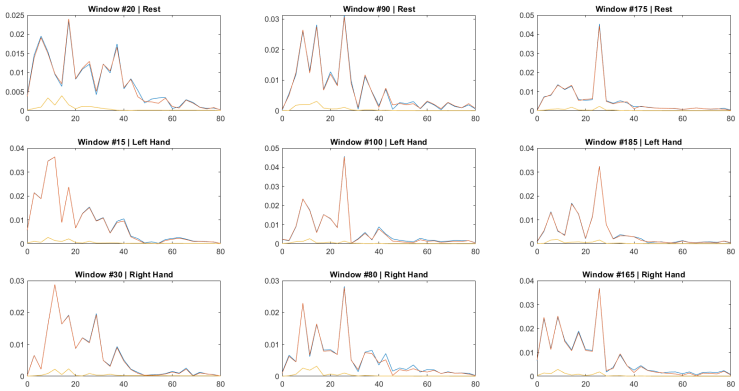


Figure 2.9: EEG signal recorded in the BIOSPIN's laboratory with the eeg equipment from the company G.tec. The data is obtained from an experiment where the user needed to close the right and left fist as the video indicated. The scale in the Y axis represents μ V and the X axis is frequency in Hz.

The problem with the measurement is mostly due to a lack of experience working with the equipment, in future works an objective would be to find the suitable way to carry out this experiments. To illustrate the recording process followed the link ahead is provided to show the video of one of the recorded sessions.

`https://youtu.be/zBsqScHJORo`

2.5 Results

Up to this point the test and hypothesis were postulated from the experience in the work with the database. The same structures that provide better results in the previous tests were used with the new data.

The first option is to test the generic models created from the database directly with the new data. Also, to retrain the net with this data could be beneficial, because we would be just adjusting the models. A second approach would be to train a model just the new data, which is a small number of samples but might be enough in some simplified classifiers.

Both of these options will be studied from each of the classification approaches that have been studied, applying the techniques that worked best in the previous experimentation with

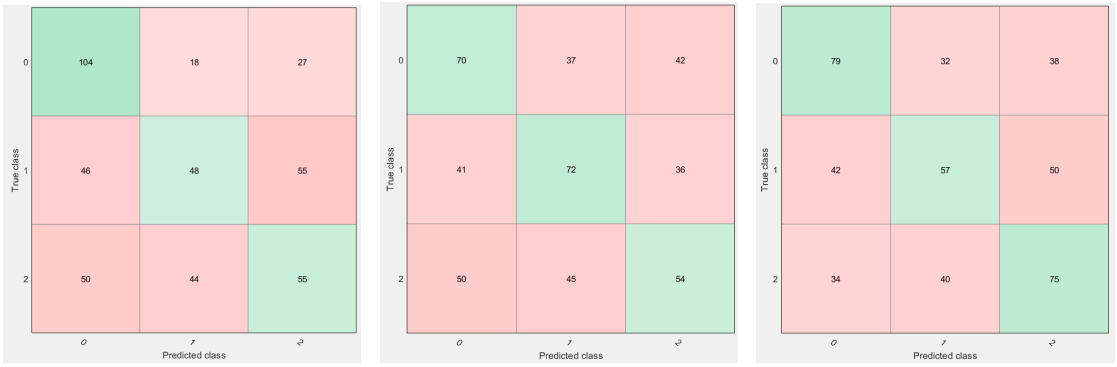
the database. However, the recording of new data couldn't provide data as reliable as in the database and in first instance the global classifiers obtained won't be able to handle the new recordings properly.

2.5.1 Feature Extraction Case

The features that define each class are extracted from the most relevant frequency bands of the signal. They were found after analyzing the database composed by many different subjects, making them valid for new data even when recorded on new subjects or conditions.

Using the global features obtained before, the trained SVM classifiers can also be used to test the compatibility of the new data with the generalist models created. Using as an input the data recorded from the experiments using the left and right fists, the old model (with an accuracy of 60.8% and the confusion matrix from Figure 2.6) is used to classify it. Once the data has been preprocessed and the features are obtained, the samples are passed to this model to adjust the classification to the new samples. In different tryouts the best accuracy obtained from this model with the new data was a 42.4% in the validation phase. The results are quite bad compared to the ones obtained with the database samples, the reproductibility of the recording experiments wasn't reliable enough.

The other approach has been to test the new data in order to train new classifiers. The main problem is the lack of data, but in exchange it has been possible to test new types of movement (like drawing circles/squares or waving right/left hand) in order to compare each for the best possible solution. In each of the cases the database consist of 447 samples divided evenly in the three classes: rest, action 1 and action 2. In the Figure 2.10 can be seen the confusion matrix of the new cases, compared to the one where both fists or legs are used, present in the database as well.



(a) (b) (c)

Figure 2.10: Each of the following images represents the confusion matrix of a SVM classifier. In the image (a) the classifier uses data from the exercise of drawing circles (class 1) or squares (class 2), with an accuracy of the 46.3% (the correctly classified values, in green, over the 447 samples used) with a Gaussian kernel. In the image (b) the classifier uses data from the exercise of waving the left (class 1) or the right (class 2) hand, with an accuracy of the 43.8% with a Gaussian kernel. In the image (b) the classifier uses data from the exercise of closing both fists (class 1) or moving both feets (class 2), with an accuracy of the 47.2% with a linear kernel.

The results are still worse than the ones obtained with the database, but the results of the drawing exercise show an interesting new option towards the MI experiments. In addition, the results are significantly better than the random option (where the accuracy would have been 33%) and they could be use in the robot control, even though the performance would be low.

2.5.2 Deep Learning Case

The DL option was based mainly on being able to use the generalist models in order to interpret the new data. Due to the handicaps in the measured data the model learned from the database is not applicable to the new EEG signals. The results of validating the models with the new data gave poor results under 40% accuracy with the LSTM best cases.

Neither it was possible to train new models from the gathered data, the amount of data necessary to generate a new model is much higher than the one available. The new exercises tested in the recording sessions were promising and for future works generate a larger dataset for the LSTM models is still the most interesting option.

Chapter 3

Robot Arm Control

3.1 Introduction

The control of the robot arm is a wide concept that goes from the reception of the user's input, up to the end of the motion. Through this process the robot is intended to help the user on a selected task in a known environment.

The problem is to define the best possible way to help in a scenario where the user is able to tell us (with his brain signals in a passive way) how he would change what he is seeing. By modifying the motion while the robot is moving, the degree of complexity increase significantly.

First of all, the user's intention must lead to a new valid path that the robot can follow, new trajectories that should fit better the user's request. Secondly there must be a control strategy that allow the robot to follow the path that is being changed by the EEG signal constantly.

This block will present the reached solution for these problems and in order to do so in a clear manner, the proposal and implementation has been divided into two parts. The first one takes care of the changes in the trajectories and the second one work with the robot motion planning.

3.2 Theoretical Background

This block is divided into two parts that work over the same robot, sharing together the same work environment and conditions to develop the required control. This section will introduce these elements and the methods necessary to reach the solution.

Mainly the background is given because of the use of a collaborative robot (Baxter), bringing

a set of theoretical concepts and technical capabilities that are key in the proposal of a solution. However, it is also necessary to introduce methods enabling the manipulation of trajectories and the strategies typically used for the management of robot's motion.

3.2.1 Dynamic Movement Primitives

One of the requirements of the block is to be able to generate new trajectories from an original one, preserving the shape of it. In 2002 Dynamic Movement Primitives (DMPs) [13] were presented as a method of trajectory control, their approach fits perfectly the problem described here.

The case of discrete DMPs is what will be implemented as one of the methods of modifying trajectories. This method is based on the idea of attractor dynamics [1] :

$$\ddot{y} = \alpha_y(\beta_y(g - y) - \dot{y}) + f$$

Where y is our system state, g is the goal, and α and β are gain terms. Therefore, the state will be driven towards the selected goal. If a forcing term (f) is added to allow modifications over the trajectory.

To achieve the desired behaviour another nonlinear system is necessary to describe the forcing term over time. The introduced system is called the canonical dynamical system, denoted by x, and has very simple dynamics:

$$\dot{x} = -\alpha_x x$$

Where f depends on this system:

$$f(x, g) = \frac{\Sigma_{i=1}^{N} \psi_i \omega_i}{\Sigma_{i=1}^{N} \psi_i} x(g - y_0)$$

There, the term y_0 is given as the initial position of the system:

$$\psi_i = exp(-h_i(x - c_i)^2)$$

Here ω_i is a weighting for a given basis function ψ_i. The ψ_i equation above defines a Gaussian centred at c_i, where h_i is the variance. So the forcing function is a set of Gaussians that are distributed along the convergence path of x to its target. Their weighted summation is normalized, and then multiplied by the $x(g - y_0)$ term, which is both a 'diminishing' and

spatial scaling term.

An image can show how the proper selection of these weights leads to a particular shape of their weighted summation. In Figure 3.1 this effect can be seen in 1D, allowing allows DMPs to adapt to almost any shape.

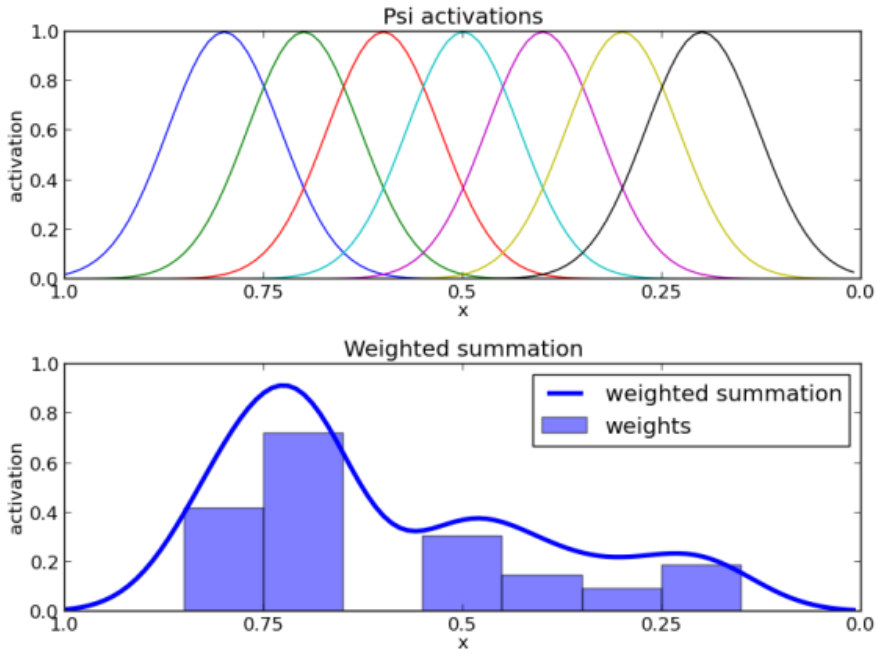


Figure 3.1: Example of the DMP shape obtained (bottom graph) with the combination of weighted ψ activations. Obtained from Studywolf website [1].

The term x always leads to 0 over time, which means that eventually the main system will lead to the attractor dynamics defined in first place. But at this point, there is still undefined the mechanism that allows to manipulate the trajectory freely.

To modify the trajectory in space we can modify the term $(g - y0)$ that defines the goal that the system tends to reach. By changing the goal, the new system will stabilize at a new equilibrium point (it tend to reach the new goal). This element that is also present in the forcing term will scale the Gaussian addition to follow the same shape in a scaled version.

A time scaling of a trajectory is also possible, but for the purposes of the project this can be done by modifying the robot joint speeds. Only spatial scaling will be used in the implementation of this method.

To learn the weights that will allow to set these shapes a given trajectory is needed. If the term y is known we can substitute in each point y, $\dot{y}$ and $\ddot{y}$. Then only the forcing term is

unknown. By using an optimization technique like locally weighted regression to choose the weights over the basis functions such that the forcing function matches the desired trajectory.

3.2.2 Inverse Kinematics

This is the mathematical process that allows us to obtain the necessary joint parameters, which lead to the desired end-effector's position. In this project this method will be a fundamental part of the motion planning [14].

The desired position of the end-effector is defined by six parameters (three position and three orientation parameters) and each joint has one parameter to define its position with respect to its arms. Given that the robot has seven joints, we will have seven unknown parameters to define the end-effector position.

Each task is defined by the sequence of movements that the end-effector would do, in the same way we think of how to move our hand to carry out a task. Based on that idea, the user will interact with the robot when he/she intends to modify the movement of this end effector.

To respond to this changes the robot must modify the end effector trajectory as the user commands, which means new values must be defined for each joint. This IK process will therefore be repeated every time new goals are sent to the robot.

To find the values of the joint parameters which will provide the desired value of the position of the end-effector, an already provided function can be casted. Rethink Robotics include in the used SDK (reference to next block) a function that finds, if feasible, the set of angles in each joint.

The fact that not every position is achievable is one of the most relevant problems to face in this block. The arm has a limited workspace and without a mobile base, the set of points that can be reached is constrained by arms length. This has a very important weight when we recall that each position might be susceptible of being modified by the user, but not all the requested modifications will lead to feasible options. This problem is tackled individually in section (ref to feasible matrix).

3.2.3 Motion Planning

Motion planning is a term used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement.

A motion planning algorithm would take a description of these tasks as input, and produce the speed and turning commands sent to the robot's actuators.

Commonly showing two parts within: one related to the optimization of the following positions to be taken and the other one taking care of the physical actions that will lead to those positions. The problem of motion planning is then divided into two different parts that in many cases can be almost independent.

Usually the part in charge of selecting the route or the following milestones is known as Global Planner. It plans ahead and the objective is to minimize a certain cost associated to certain rules evaluating the movement. In some cases it will only run the calculation once to find the best path, but usually there are external inputs (obstacle sensors, internal sensors or user's input) making the calculation more complex.

The part that receive the goals (selected by the Global Planner) and execute them by translating its milestones into commands to the actuators, is the Local Planner. Since this is a robot arm with its base "fixed", the Local Planner won't need manoeuvring capabilities.

3.2.3.1 Configuration Space

A configuration describes the pose of the robot, and the configuration space [15] C is the set of all possible configurations. If the robot is a fixed-base manipulator with N revolute joints (and no closed-loops), C is N-dimensional.

To work within a more comfortable space (as referred in section 3.2.2) the point of view can be changed to consider the end-effector as a solid 3D shape that can translate and rotate. The workspace is 3-dimensional, but C is the Special Euclidean (SE) group $SE(3) = R^3 \times SO(3)$ (being SO the space defining the orientation), and a configuration requires 6 parameters: (x, y, z) for translation, and Euler angles (α, β, γ).

From these spaces, when the unfeasible areas are removed, the remaining is what can be called Free space. The reason to consider a region as feasible or not, in most of the motion planning cases, is a collision (with external elements or with the robot body itself). A simplified example of this configuration space can be seen in Figure 3.2, where the orientation of the two joints defines the configuration space with all the possible positions, and the unfeasible areas.

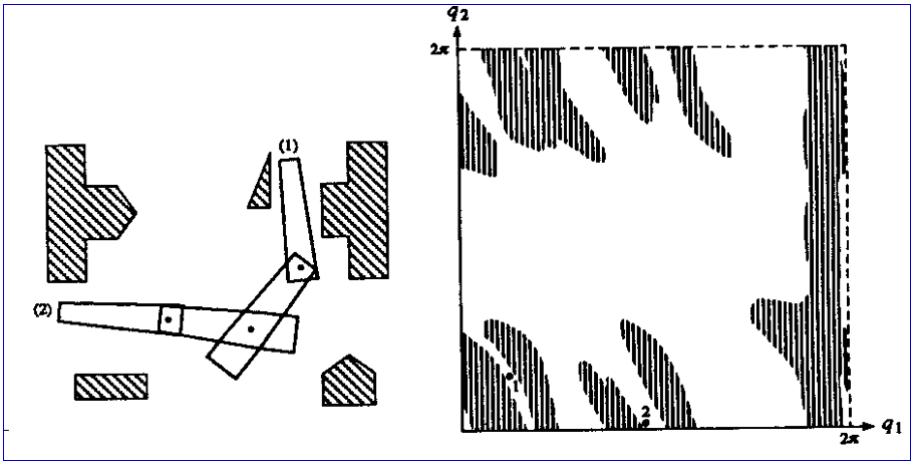


Figure 3.2: The left part of the image shows the cartesian view of a robot with two degrees of freedom, and the equivalent scenatrio in the configuration defined by the position of the two joints. The robot is shown in two positions that appear also represented as dots in the configuration space. Obtained from the Stanford Artificial Intelligence course

3.2.4 Collaborative Robots

As the word collaborative suggest, this type of robots are designed to work with people [16]. The problem of working with humans is the uncertainty that it brings, and the bigger requirements in security as a consequence.

There are various types of collaborative robots and only one type of collaborative robot can be used without any additional safety features. According the international standards ISO 10218 part 1 and part 2, there are four types of collaborative features for robots represented in Figure 3.3:

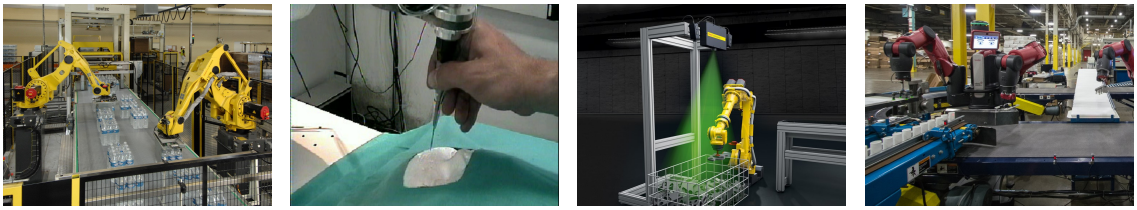
- Safety Monitored Stop: This kind of collaborative feature is used when a robot is mostly working on its own, but occasionally a human might need to enter its workspace. If the human enters the restricted area, while the robot is performing an action, the robot will stop all movements altogether. The robot is not shut down, but the brakes are on. These types of robots are most efficient when people work mostly apart from the robot, but might get closer for a small amount of time. Otherwise, a lot of time could be lost as the robot continuously stops for human interruptions.

- Hand Guiding: This type of collaborative application is used for hand guiding or path teaching. For example, this is a common feature while working with pick and place applications, where new paths can be learnt quickly. However, it must be noticed that

this type of collaboration uses regular industrial robots, but with an additional device that can measure the forces that the worker is applying on the robot tool. In many cases the sensor is placed in the end-effector, then the user can guide the tool installed in the robot arm.

- Speed and Separation Monitoring: Here the environment of the robot is monitored by lasers or a vision system that tracks the position of the workers. The robot will act within the functions of the safety zones that have been pre-designed for it. If the human is within a certain safety zone, the robot will respond with designated speeds (generally slow) and stop when the worker comes too close. So, when the workers are approaching the robot, it slows down, as the workers approach even closer, the robot slows down even more or stops. The difference with the first feature is the positioning detection that allows the progressive speed reduction, meanwhile the other group just detects when the safety limits are trespassed.

- Power and Force Limiting: The robots with this feature can work alongside humans without any additional safety devices. The robot can feel abnormal forces in its path. In fact, it is programmed to stop when it reads an overload in terms of force. These robots are also designed to dissipate forces in case of impact on a wide surface, which is one of the reasons why the robots are rounder. They also don't have exposed motors. A lot of these robots are certified by third parties who focus on industrial safety for human-robot collaboration.



(a) (b) (c) (d)

Figure 3.3: The figure portraits the four types of collaborative features: (a) shows Horan robots with Safety Monitored Stop, (b) shows an example of medical surgery in CREB (Hand Guiding), (c) shows a vision system for Speed and Separation Monitoring (from a FANUC's device) and (d) shows the robot Baxter from Rethink Robotics with Power and Force Limiting.

The technical specification ISO/TS 15066 (ref?) will specify the maximum forces (N) and energy (J) that can be applied on a human without any harm. This technical specification

will clarify the safety requirements for human-robot collaboration for both regular industrial robots and force limited collaborative robots. Complementing the norm ISO 10218 (ref?) to determine the type of collaborations that a robot can work with, associated to a set of minimum requirements.

3.2.4.1 Baxter Robot

Baxter (as seen in Figure 3.4)is a robot design and distributed by Rethink Robotics as a collaborative robot that can share the workspace with a human. Among the listed features that define the collaboration, this device has power and force limiting as its main advantage.

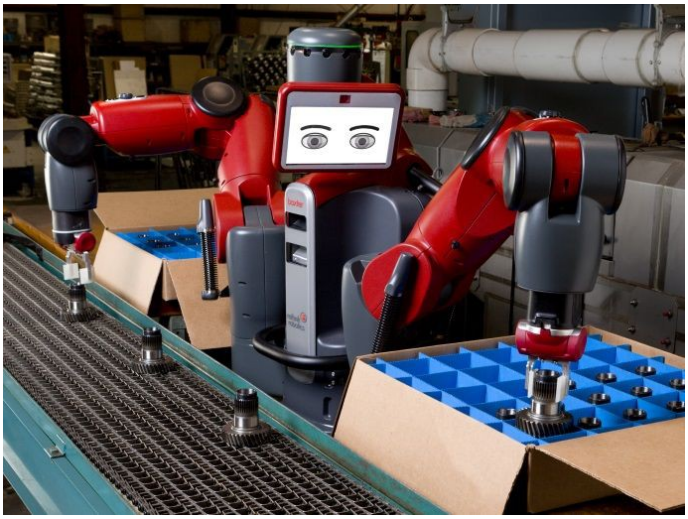


Figure 3.4: Baxter, from Rethink Robotics, in an industrial workplace.

First of all, the main feature of this robot is his ability to read forces in their joints. This allow Baxter to detect when abnormal forces are applied on it while it is working. In these situations, it can be programmed to stop or sometimes reverse positions mediating the initial contact. This means it can respond immediately if they come into contact with a human and perhaps dissipate some of the energy transferred from the impact.

As a consequence of this feature the robot can be used with the "hand guiding" feature. With prepared features that allow to easily manipulate the robot position and to record the positions required for a certain task. In this project the base trajectories are designed over a few critical points obtained this way.

The robot also comes with integrated cameras and a sonar control. The cameras are placed in the extreme of the robot arms and in the head, providing several points of view. The sonar

control is placed in the "head" and has 12 evenly spread sensors that detect objects around Baxter.

PHYSICAL SPECIFICATIONS	
Robot Height	3' 1" without pedestal 5' 10" – 6 '3" with adjustable pedestal (optional)
Reach	Maximum reach 1210 mm
Torso Mounting Plate Diameter	13.3" (for mounting on table)
Body weight	165 lbs. without pedestal 306 lbs. with pedestal (optional)
Degrees of Freedom	14 (7 per arm)
Pedestal Footprint	36" × 32"
Max Payload (Including End Effector)	5 lb / 2.2 kg
Gripping Torque (Max)	10 lb /4.4 kg

Table 3.1: Physical specifications of Baxter provided by Rethink Robotics in their website.

When it comes to the software, the robot includes an application programming interface (API) developed in Python and ROS that can be triggered to control the hardware. The robot was designed for a mimic behaviour and most of the functions are oriented towards the hand guided feature. However, the API allows to control the position of each joint and to obtain the inverse kinematics of a certain position (for each arm).

3.2.5 Robot Operating System

The Robot Operating System (ROS) [17] is a robotics middleware (i.e. collection of software frameworks for robot software development). Although ROS is not an operating system, it provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor, control, state, planning, actuator and other messages (ref definition).

The main ROS client libraries (C++, Python, and Lisp) are geared toward a Unix-like system, primarily because of their dependence on large collections of open-source software dependencies. For these client libraries, Ubuntu Linux is listed as "Supported" while other variants such as Fedora Linux, macOS, and Microsoft Windows are designated "Experimental" and are supported by the community.

ROS has gone through several versions that doesn't have complete compatibility among them. These compatibility issues limit the software environment where the project can be developed. The Baxter API is designed to work with the Indigo ROS version, mainly used with Ubuntu 14.04. There exist certain tools as dockers or virtual machines to avoid the compatibility problems, but in this case the best approach is to develop the project within these versions of ROS and Ubuntu.

3.3 Online Trajectory Manipulation

The use of a robot in the project seeks to achieve a certain movement that can fulfil a task. The movement of the robot arm during this action can be given by a trajectory (sequence of points oriented in the space). This trajectory describes the positions of the end effector, which is the element interacting with the scene.

The actions to be performed are simple domestic tasks, but in those tasks each user might have a different way of doing it. In some cases, it just means to place an object a bit closer. However, while talking about the trajectory, these small differences can change the whole path. In fact, it is proposed here that all the small variations of a certain task have a common root, for example, there is a similar shape in the way two different people poor the milk.

Even if it is a slight modification, before a robot arm can carry out the new motion, the points must be validated for feasibility. Also, there is a challenge in how to modify these trajectories when we know we want to preserve partly the shape of the curve (since it defines the common root of certain action).

3.3.1 Objective

Having a more natural adaptation of the robot can be important to close the gap between people and the assistive technologies. The objective in this part is to obtain a method capable of modifying a base trajectory and provide a new feasible path.

The manipulation of this trajectory must lead to an improvement from the user perspective.

This criterion defines a set of requirements for this block:

- Smooth Movement: The transition toward the new trajectories must be natural and without sudden changes.
- Manoeuvrability: Every segment of a task should be susceptible of changes, every part of it might be adapted to make the user feel more comfortable with the movement.
- Shape Preservation: The activities that are thought to fit this proposed approach, will be based on a unique trajectory that will be modified. In any case the modifications should respect the shape of the root movement that defines the task.
- Robustness: This project aims to improve the assistive robots, so including the online modification can't lead to the failure of a task that would have worked initially, the new trajectories must be feasible.
- Constant Adaptation: During the whole activity period the user might want to adjust the movement of the robot arm. Therefore, the trajectories must be recalculated almost constantly.

3.3.2 Proposal

This block is subdivided into two parts that correspond to the two operations that the trajectories require before being sent to the robot. First, it is necessary to adapt to the user's command, and therefore modificating the path initially designed. But, this new path has not been proved feasible and we for reachability of the final positions all points in the trajectory need to be valid. The second operation checks the feasibility of a trajectory and minimum necessary modifications to correct possible conflictive points.

3.3.2.1 Trajectory Modification

As mentioned before (section 3.2.2), we want to modify the trajectories described by the end-effector acording to the users input. Before commanding any movement to the actuators we need to find to which position.

In this section the relation between the inputs and the new trajectories will be discussed, how we can use this input to generate a change in a whole trajectory.

3.3.2.1.1 EEG Input Treatment

Due to the complexity in the control that the user has, we will limit the modifications to one direction. This direction can be any in space so that in this case where the base trajectories are known beforehand, it is possible to choose the most interesting modification in each case.

The options that can be received are two opposite commands (like right-left or up-down) and a rest state that is constantly being read. Due to the high error rate in this classification (see section 2.5) the most used approach [2] is to assign a positive, a negative and a zero value, one to each class. If now it is assumed that the person is trying to send a command by imagining moving his right hand, if this is associated with a positive value, the integral of many of these values (even when classified incorrectly) should remain positive. In fact in future references to this value the term EEG integral will be used.

It can be set a maximum absolute value for the modification that can be obtained, since the objective is lead to small adaptations it shouldn't compromise the objectives of the control strategy. Now the range of modifications is discrete (there is a minimum positive and negative value, associated to the detected classes) and finite.

Even though this represents a finite number of possible trajectories, an online adaptation would allow to easily include new actions to the robot. The constant adaptation necessary comes from the constant update of the EEG states received. The integral value that defines the modifications is constantly updated and the modifications must be applied repeatedly to adjust the movement at each phase of the movement (examples of the EEG input treatment are shown in Figures 3.5 and 3.7).

3.3.2.1.2 Dynamic Movement Primitives

The first method used to modify the trajectory is the DMPs. As explained before (in section 3.2.1) this method can store one function that defines a variable and re-adapts it. In the case of the robot end-effector there are six variables that we need to obtain (position and orientation).

To codify each variable, a new set of weights would be necessary to adapt to its evolution in time (that define ultimately the trajectory). Therefore, we have six individual sets of DMPs that can be used to define the action and obtain new trajectories from it.

If in addition, the action would be split in time to make different phases in the same task, we would need six new DMPs for each section created this way. In this way it is possible to provide more points to articulate the movement. This is a result of the characteristic of the re-scaling in space, where we can adjust the whole trajectory when the final goals are changed.

The way we include the integral value related to the EEG signal is by modifying the goals with it. In each section there would be a direction where the modification is more relevant (for example vertical to sort an obstacle or horizontal to place a cup). We add this value decomposed along the relevant direction and we add the resulting values to each goal. With the new goals we can adapt the trajectories and obtain the new points easily.

This means that the user can control the position of the closest goal, which is the starting point of the next section (apart from the case of the last section of the motion). This seeks to make the trajectory as flexible as possible at each point. An example of DMP modification is shown in Figure 3.5 where a random signalis processed and the effect of the integral value

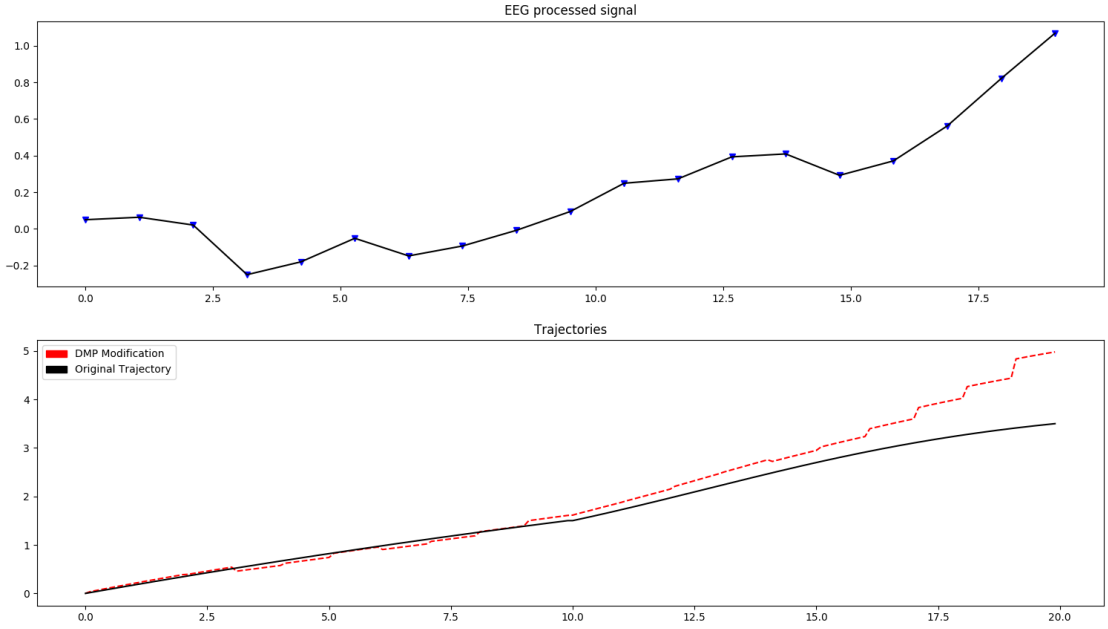


Figure 3.5: The plot above shows the EEG integral value evolving through time (X axis) with a random emulated input. Below the second plot shows how the DMP method adapt the original trajectory according to this input. Before every update of the trajectory, 10 random values are used to obtain the new value of the EEG integral.

3.3.2.1.3 Virtual Force

To fit the objectives the modification methods should be fast to compute and preserve the shape of the path to be followed. An easy option to implement is to define a set of virtual forces that determine a certain variable evolving in time.

These variables are the six degrees of freedom from the end-effector position. Each variable has a known set of values, the base trajectory. If the force model is applied over a particle, representing the current state of the robot, there must be a force acting over it pulling towards the base position at this timestep. Also, another force must represent the user' will to adapt the trajectory, pushing it out of the base trajectory.

Electric field equations are used to model these force interactions. This forces are defined by Coulomb's law where a particle with electric charge q_1 at a position x_1 generates a force over a particle with charge q_0 at a position x_0 following the following formula:

$$F = \frac{1}{4\pi\epsilon_0}\frac{q_1 q_0}{(x^1 - x^0)^2}\hat{r}_{1,0}$$

Where $\hat{r}_{1,0}$ is the unit vector from x_1 to x_0. As a result when the charges have the same sign the force is positive and the particle is pushed away from the other. Derivated from this idea appears the concept the electric field at point x_0 due to the point charge q_1; it is a vector equal to the Coulomb force per unit charge that a positive point charge would experience at the position x_0:

$$E(x_0) = \frac{F}{q_0} = \frac{1}{4\pi\epsilon_0}\frac{q_1}{(x^1 - x^0)^2}\hat{r}_{1,0}$$

In practical cases the formula of the electric force generates forces really big when the distance between particles is small. For that reason and to simplify the tuning of the parameters in the future, we add 1 in the denominator of the formula to avoid the cases where the force tend to infinite.

To model the evolution of the position in time, each of the variables $(x, y, z, \alpha, \beta, \gamma)$ is considered as positively charged particles representing the robot current state. In a 2D scenario, the particles are moving along the X direction (time) with a constant velocity (the speed of the robot reaching new positions). Also in the same scenario are necessary two forces actuating over the particle along the Y axe (the position of the robot). This forces come from the elements representing the user's will and the base trajectory:

- Ghost particle: this particle follow the function $f(t)$ that defines the evolution of the position (each variable) of the end-effector through time in the base trajectory. It is negatively charged so that it attracts the positive particle to the base positions, in the absence of other forces the point with less potential energy is the same point that the negative particle occupies. Since the only interest is in modifying the value in the Y axe,

this particle is always placed in the same X when computing the force interaction over the positive particle.

- Conducting parallel plates: two infinitely large plates in X maintaining a voltage between them. They are displayed up and down from the particles, creating a uniform electric field depending on voltage (electric potential difference $\Delta\phi$) with the formula:

$$E = -\frac{\Delta\phi}{d}$$

  In this problem the distance would be fixed and the value of the voltage is directly proportional to the EEG value integrated, either positive or negative depending on the users will.

Both forces define over the positive particle a position in the Y axis for each value in X, where there would be a static equilibrium. This positions will define the goals of the robot at every instant, as a function of the EEG input received in that moment. With this proposal is easy to introduce the users input to obtain a new trajectory without recalculating the whole trajectory, picturing a scenario like the presented in Figure 3.6.

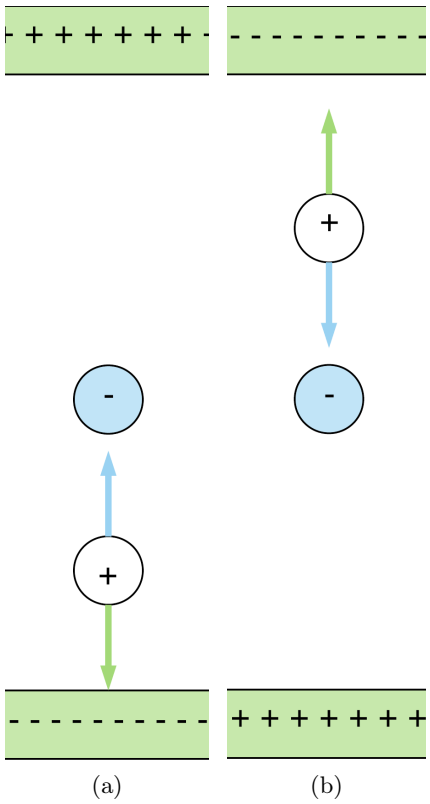


Figure 3.6: Both images display a particle with positive electric charge $q0$, a particle with negative electric charge $q1$ and a uniform electric field produced by two plates at a distance d with a potential difference $\Delta\phi$. In both scenes the blue particle remains in the same position and the voltage changes the value and the sign. This would exemplify the impact of the user modifying the trajectory through the voltage.

In order to set a limit in the practical sense, the maximum deviation from the base position is saturated to a maximum value. This limit should be set according to each movement and the feasibility of the points nearby the trajectory.

To show the effect of the Virtual Force approach the Figure 3.7 shows the modification of a trajectory according to a random EEG input:

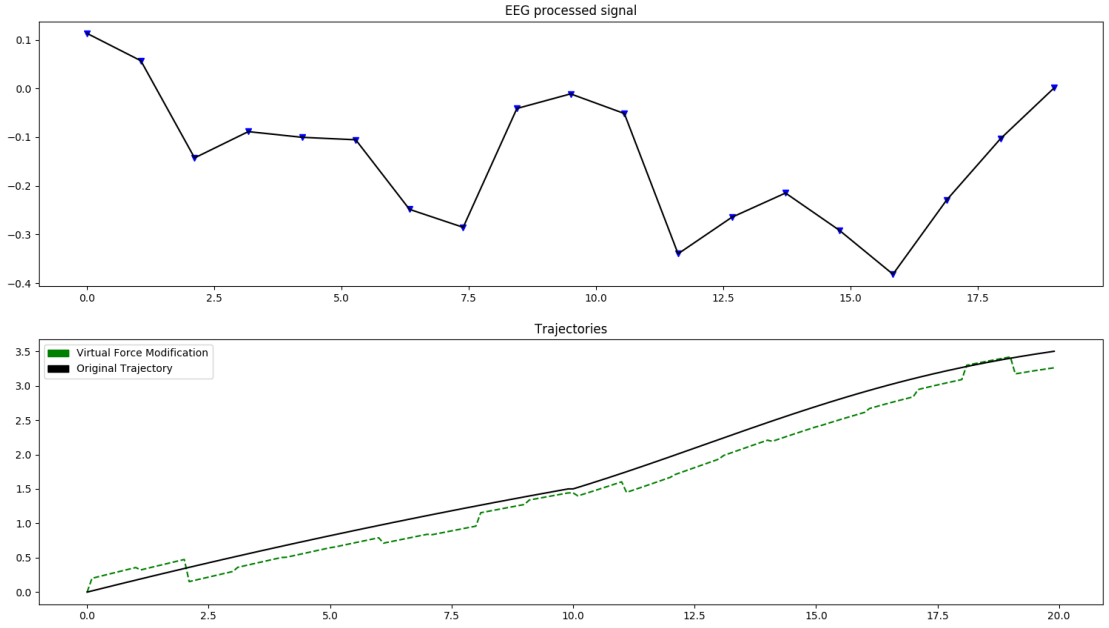


Figure 3.7: The plot above shows the EEG integral value evolving through time (X axis) with a random emulated input. Below the second plot shows how the Virtual Force method adapt the original trajectory according to this input. Before every update of the trajectory, 10 random values are used to obtain the new value of the EEG integral.

3.3.2.2 Feasibility Adaptation

Once obtained the modified points for the end-effector from the previous methods, the new positions should be feasible for the robot joints. Not every position is feasible, most of the new trajectories might include unfeasible points. Exploring all the possible combinations is not feasible and would lead to little flexibility while testing movements.

This was a problem encountered during the testing of the modified trajectories. Most of the points in the modified paths were unfeasible and lead to a sudden stop. The problem in ultimate instance is not the unfeasible points, but how to reach the feasible points could substitute that unreachable positions.

To understand the problem, a study over the distribution of the feasible points was carried out (for the Baxter robot discussed ahead in Section ref). Taking a cube that includes the furthest points that the robot can reach, the whole space can be discretized and tested each single point in a simulated way.

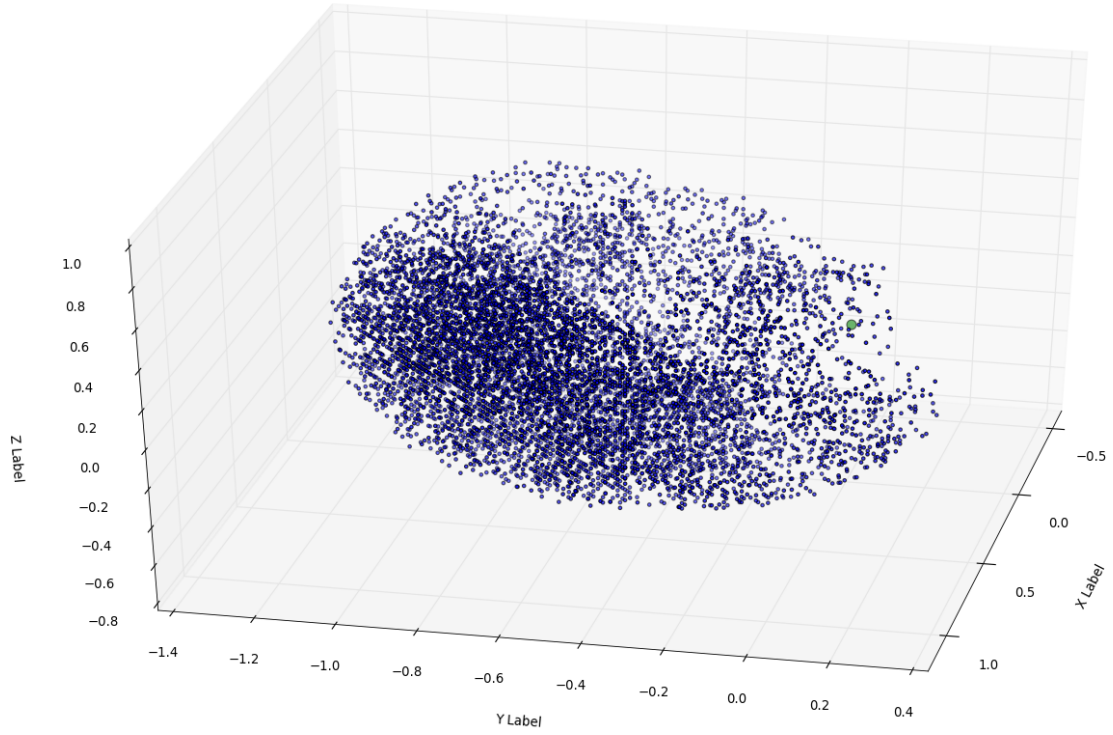


Figure 3.8: The plot shows blue points for each feasible position detected, in the cube inscribed within the axes values only the 4.42% of the 1244421 points are feasible. The orientation of the end-effector, respect to the robot reference axis in the base (being the $(0, 0, 0)$ pose in the center of the base, represented by the green dot), is $(x = -0,153, y = 0,707, z = 0.17, w = 0.667)$. The axes show distance in meters.

To simplify the case, we take only three variables (x, y and z) and consider the orientation fixed. Now it can be visualized (as done in Figure 3.8) also how the feasible points are spread in space and which areas are better to work with. This simplification will also be used in real movement since for most of them the orientation was constant or switching between two options.

The results observed in Figure 3.8 are partially because the area considered contain points out of the robot's reach. However only between 4-6% of the points (depending on the discretization used in each trial) are feasible and it can make unfeasible the trajectories if they dont go through areas with "high density" of feasible points .In order to visualize the previous result with the concept of density, in Figure 3.9 the plots are in 2D (x, y) (for a fixed z) and indicate the amount of feasible neighbours (among the 27 considered closer points).

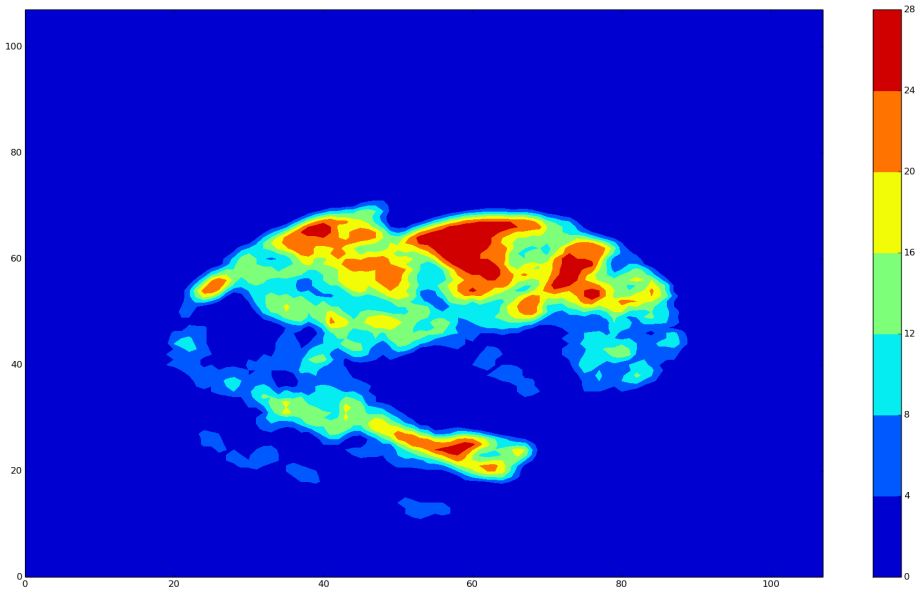


Figure 3.9: The plot represent the same analysis of the workspace at a height $z = 0m$ from the base. In each position the index of (x, y, z) are combined taking the previous and the posterior positions also, making a total of 27 points. The axes show the point position, with a distance between points of 0.042m, being the base in of the robot in $(12, 30)$.

With the simplified case, the proposal to find a way to link the unfeasible positions to the new ones fast. In first instance the solutions were developed around the discretized workspace, using different grid sizes to test the performance changes.

The search of new points in the surroundings without aiming for a direction lead to slow computations up to the point of being unfeasible. In the ideal case we would have a "hyperlink" from every wrong position, pointing to a feasible and most suitable point.

In the discretized simplification is reasonable to store each point as feasible or unfeasible (0 or 1), with this 3D matrix as a base the "hyperlinks" matrix can be built as indicated in the algorithm 1. This algorithm search through all unfeasible points and link them to the nearest feasible point. By just looking in the neighbour points (26 surrounding points in the 3D space) the algorithm finds either feasible points or hyperlinks (obtained from an unfeasible neighbour point).

Comanda 1 Hyperlink matrix algorithm

Data: N_x, N_y, N_z define the amount of index per axe. Each axe has an upper and lower bound of the maximum Cartesian values (f.e. $[x_0, x_1]$). A matrix *mat* of size (N_x, N_y, N_z) store if each position is feasible or not (marked as 0 or 1 respectively).

Result: The hyperlink matrix *Gradients* with size $(N_x, N_y, N_z, 3)$

```
Finished = False
ze = [], tw = [], n = 0, m = 0
while not Finished do
    Finished = True
    i = 0
    for x ← 1 to (N_x − 1) do
        for y ← 1 to (N_y − 1) do
            for z ← 1 to (N_z − 1) do
                i = i + 1
                if mat(x, y, z) == 1 then
                    Finished = False
                    for all neighbour points of (x, y, z) do
                        if mat(xn_i, yn_i, zn_i) == 0 then
                            n = n + 1
                            ze ← (xn_i, yn_i, zn_i)
                        end
                        else if mat(xn_i, yn_i, zn_i) == 2 then
                            m = m + 1
                            tw ← (xn_i, yn_i, zn_i)
                        end
                    end
                    if n > 0 then
                        Gradients(x, y, z, :) = Closest Point to (x, y, z) from ze)
                        mat(x, y, z) = 2
                    end
                    else if m > 0 then
                        Gradients(x, y, z, :) = Closest Point to (x, y, z) from tw)
                        mat(x, y, z) = 2
                    end
                    ze = [], tw = [], n = 0, m = 0
                end
            end
        end
    end
    return Gradients
end
```

With this *Gradients* matrix the necessary information to obtain new points is accessed fast and provide points checked as feasible. There is still a barrier to bring it to practical cases due to the continuous nature of the modifications described above.

This matrices aproach work with discretized values, but from the EEG input treatment we receive (almost) continuous values, that need to be adapted to our matrix grid. The continuous points when put under the created grid (in 3D space) are always inscribed in a cube defined by eight points that do belong to the grid. First these positions should be checked to find if they are feasible or unfeasible. Three scenarios are proposed to obtain a result from these neighbour points:

1. All of them are feasible: as seen in the Figures 3.9 and 3.8 there are important regions where all the points might be feasible. Only when all the neighbour points are valid is considered that the continuous position might be feasible as well. It is tested individually with the IK function, and in case it is unfeasible the set is treated as in the case number 2 ahead.

2. Some of them are feasible: this means that at least one point is valid and can be used. When there are more than one the closest point we choose the one that is closest to the best approximation of the continuous coordinates in the discrete space.

3. None of them is feasible: This case is repeated many times when we try to reach far positions. In this scenario the point that is closest to the continuous position, among the eight neighbours, will lead to the feasible and suitable position. With the coordinates stored in the "hyperlink" the unfeasible point can be substituted without drastically modifying the trajectory.

The limitation of this method relies on the robot feasible reach. Some of the areas are only valid for certain orientations, and in many cases (carrying a cup of water) the orientation changes are not acceptable. In other cases, the simplified 3D matrices can be upgraded with a 4^{th} dimension that represents an orientation degree of freedom where is reasonable to move, remaining the algorithm as it is now (but extended each position vector with the new value).

3.3.2.2.1 Cluster Classification for Non-Discretized Scenarios

The problem of the unfeasible points can be solved through the mentioned procedure (inscribing the continuous point in the cube of grid points), providing a robust response to the changes in

the trajectories. However, to achieve so we use a discrete solution instead of continuous values, towards future implementations the option of continuous aproaches is worked out.

In the previous case only when all the surrounding points (from the discrete scene) where feasible, the given point was considered feasible. Since we saw the existence of clusters, this is a reasonable simplification but during many tests it was seen how most of the "feasible clusters" also contain many unfeasible points and the condition of all feasible was barely ever met.

Given this problem, where it is interesting to identify the regions where the robot can move more freely, the SVM classifier can obtain a good performance using the appropriate kernel (discused this classifiers in section 2.4.3). This method is chosen in first place due to the easy definition of the features, in this case the coordinates that define a position. In second place, the complexity to define this areas and define the clusters is higher than the number of features (three dimensions (x, y, z)) and the SVM can use Gaussian kernels (among others) to deal with this problem.

In addition, it is possible to obtain the mathematical expression of those regions. This information can be used to find the gradients that lead to feasible regions from the other areas.

When tested the capability to classify this data (the matrix containing if each point is feasible or unfeasible) the amount of samples in each class must be balanced. Randomly removed the excess of non-valid points samples, the data can be used to train a classifier. The results obtained are shown in the Table 3.3.2.2.1, where the accuracy presented reach up to 85% for the feasible points and 99% for unfeasible areas. With these results, including this classifiers to deal with the "continuous input" can lead to performance improvement.

3.3.2.2.2 Extended Definition of Feasibility

The idea of labelling the whole space as feasible or unfeasible can be extended. In this case it has only been linked to the capability of the robot to reach a position. In a practical sense there are other elements that should determine which positions are feasible, for example the presence of obstacles.

The same process presented before can be adapted to consider new definitions of feasibility by changing the initial matrix of ones and zeroes. The objective remains the same since the problem is still to reach valid close positions from a modified position that turns out unfeasible.

In a future work proposal, a 3D vision camera (Figure 3.10 shows the CREB robotic kitchen through a 3D vision system) can be used to make another matrix with ones and zeroes, where one means that there is no object in that position. By multiplying this with the other matrix

SVM PERFORMANCE REPORT							
Test #	Version	Samples Ratio	Class	Precission*	Recall**	Samples(y_true)	1 out of
1	V3	4,03%	0	0,75	0,92	5614	1
			1	0,97	0,89	14680	9
2	V3	4,03%	0	0,8	0,97	4097	1
			1	0,98	0,87	7195	14
3	V3	4,03%	0	0,83	0,99	2854	1
			1	0,99	0,85	3624	18
4	V3	4,03%	0	0,83	0,99	2723	1
			1	0,99	0,84	3397	20
5	V3,5	4,33%	0	0,84	0,99	2911	1
			1	0,99	0,84	3393	20
6	V4	4,42%	0	0,84	1	5499	1
			1	1	0,83	6181	20
7	V4	4,42%	0	0,84	0,99	5466	1
			1	0,99	0,85	6881	18
8	V4_5	4,42%	0	0,85	0,99	6051	1
			1	0,99	0,84	6777	18
9	V5	4,86%	0	0,85	0,99	6074	1
			1	0,99	0,85	6824	18

Table 3.2: Results from the use of SVM classifiers to predict the feasibility of a point for a fixed orientation and given position. The version refer to the matrix used as a database (the matrix containing 0 or 1 if the points are feasible or not). The Precision column presents the result of $P = tp/(tp + fp)$ and Recall presents the results of $P = tp/(tp + fn)$ (where tp is true positive results, fn is false negative results and fp is false positive results).

only the cases where a point is feasible in both scenarios, will remain as feasible. This is a hypothetical improvement that could be implemented in future expansions of this project.

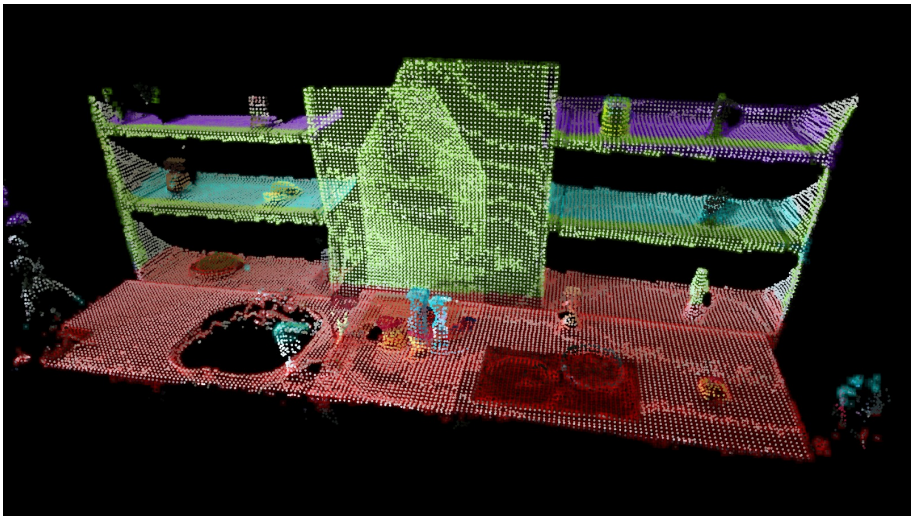


Figure 3.10: CREB robotic kitchen image generated with two Kinect cameras.

3.3.3 Implementation

The implementation of these computations was done in Python, using the ROS libraries in order to assemble these results easily. Among the scripts prepared, they are oriented in two different ways:

- Nodes: To gather the required data, test and visualize the methods that allow the trajectory modification. This work is necessary to study which approach is better and how to tune the required parameters. Each node communicates with the robot and the other nodes (from the software ROS discussed in section 3.2.5) in order to gather information or send it. The number of parameters to adjust is really high (discretization size, directions of modification, resolution of the discrete trajectories, EEG signal integral saturation, virtual physical constants, etc.) and after the implementation process there is still room for improvement, specially to adapt to the other blocks of the project.

- Libraries: The results obtained and the algorithms prepared are meant to be used in the online manipulation. The data is stored in files, but to be able to use the functions easily in the robot control they are stored as libraries. Based on the test codes several functions where developed to grant a direct access to the information relevant during an online manipulation.

The code developed, the trajectories generated and the matrices obtained are available in

the following link:

`https://drive.google.com/open?id=1a5Br3dUKidvcLLHaYhdfE63wMPp35jb7`

3.3.3.1 Base Trajectories

In addition, a set of trajectories was created to represent the base of some domestic activities. To generate that trajectories, it was needed a reference from the real scenario. If some critical points delimiting the sensible locations are obtained with the robot, the movement in between can be interpolated. To make them more natural the interpolation is obtained by combining the shapes of first, second and third order polynomials and sigmoidal functions.

The result of this combination is a piecewise-defined function where the intervals composing it are usually defined by the measured relevant points (in Table 3.3.3.1 an example of this point selection is shown). For a completely smooth movement the combination of this segment could keep the function differentiable in each point. In future works the development of trajectories can take more complexity and include also the user's feedback to evaluate these base shapes.

		POSITION			ORIENTATION			
V3 P&P	With orientation 4,5	X	Y	Z	x	y	z	w
Pick_Right_0	far from table, ready to start aproach	0,74317	-0,25991	0,25101	-0,15335	0,70737	0,17641	0,66707
Pick_Right_1	placed in the object position (ready to grab)	0,91462	-0,2076	-0,06565	-0,15335	0,70737	0,17641	0,66707
Place_Right_0	placed in the object position (object grabbed)	0,91462	-0,2076	-0,06565	-0,15335	0,70737	0,17641	0,66707
Place_Right_1	on top of the first obstacle	0,93016	-0,09953	0,27326	-0,15335	0,70737	0,17641	0,66707
Place_Right_2	base after first obstacle	0,85468	0,01	-0,06565	-0,15335	0,70737	0,17641	0,66707

Table 3.3: The table presents the different milestones necessary to define the base trajectory V3(Pick and Place) used in the robot experimentation. The orientation remains through the whole movement and all the values have the robot base and orientation as the reference frame.

3.4 Robot Arm Motion Planning

This block takes care of the robot arm movement and the integration of the appropriated work environment. Using the incoming information from the previous blocks, this part must control the actuators of the robot to reach the desired positions.

The main task in this block is therefore the motion planning, which is the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement.

This block uses mainly already prepared functions (from the SDK of the robot or from the other blocks in this project) to find the next movements and execute them. Working along with humans requires that this process can be stopped at any time in case of an emergency.

3.4.1 Objective

Some of the problems arousing in the previous block are still a burden inherited in this block. Also, some new objectives are proposed to deal ensure a smooth and clear functioning:

- Time Response: As in previous cases we must build the different elements considering that a fast response would make the "online" control more real. In this part (Robot Arm Motion Planning) there are several cases where the communication among elements can lead to time delays.

- Robustness: the construction of the motion planning scheme must include all the required corporations that lead to a robust solution, where the planner can handle all the proposed scenarios.

- Smoothness: there are different ways to command the movement and to manage the actuators, a better control of those elements help to generate a smooth movement. One of the global objectives in the project is to provide a comfortable experience while working with robots, the online control is not enough if the movement is not smooth.

- Neat Structure: Many elements are being summoned from the motion planner, a lot of data need to be accessed and there must be several communications. The final code must be able to operate all these elements while remaining clear and understandable. Since the project doesn't reach a perfect solution, a neat structure is necessary for future works extending this project.

3.4.2 Proposal

A control system must take care of the movement control according to the requirements and objectives set. The control is based on the motion planning theoretical background, but it also has to include the solutions addressed in the previous blocks.

This is not the case of a fully autonomous robot since the user's input determines the movement. However, this can be seen as a case of replanning, where a certain event requires that the trajectory is adapted. The events (user's input) are taking place constantly and different variations from the original trajectories are appearing.

3.4.2.1 Global Planner

Taking the structure typically used in motion planning, the elements managing this constant replan and the definition of these new trajectories, will be part of a Global Planner. It implies that the functions developed in the previous blocks must be integrated to define the modifications in the trajectory.

The first element to handle in the global planner is the user's EEG signal classification result, indicating how to modify the trajectory. The problem relies in the error rate given by this classification, making a single classification unreliable. That is the reason for using the integral of the previous values (section 3.3.2.1), leading to a more robust response.

Unless there is a clear tendency towards one value the effect on the trajectory won't be noticeable. On the other side, the response when the user tries to switch to the opposite modification is slower. To limit that effect there is a saturation limit for the resulting value of the integral, limiting the size of the gap to overcome when these changes are applied.

With the value that will define the modification updated, the global planner asks for the modified trajectory (section 3.3.2.1.1). To obtain the new trajectory, it requires as well the direction where this change will be applied. The direction in the code is a vector that weights the influence of the modification in each dimension (being the weight 0 when a certain dimension must remain still).

With the new set of points obtained, the GP check the feasibility (section 3.3.2.2) of each of them before sending them to the LP. The function used for the check returns the coordinates of a point, the same as the given one if it is feasible or a new point if it is not.

Once this check is done the GP awaits to send the valid new point to the LP until the buffer in the LP is empty. This means that the robot has reached its last goal and request a new one from the buffer where the next position is stored.

All the communication and information flow bring together the different elements and the LP and GP must handle them meanwhile they manage the movement. the pseudocode that would run in the GP to acomplish a whole trajectory is shown in algorithm 2.

Comanda 2 GP loop sequence

Data: The given data are a base trajectory and the matrices for check the feasibility. The base trajectory (divided in N_p parts of N_g goals each) is defined as a vector of matrices $Tr \rightarrow (N_p)(3, N_g)$.

The given matrices $Mat, Gradients$ are necessary to check the feasibility.

The trajectory modification and the feasibility check are carried out by functions loaded from the created libraries, also *reached*() informs of the LP feedback (the last goal that it has completed) and *eegInput*() of the EEG input treatment.

It is necessary a direction that indicates how the input will influence each variable, stored in the variable *rep*.

Result: Sequence of points (goals) adapted to the input values that are also feasible, *nextGoal*

Initialization:

$copyTr = Tr$
$partSize = (N_{g1}, N_{g2}...)$
$state = 0$
$nextGoal = Tr(state)(:, 0)$ $n = -1, i = 0$, **while** $state < N_p$ **do**
  if $reached() || n < 0$ **then**
    $n = n + 1$
    $eeg = eegInput()$
    $nextGoal = modifyTrajectory(eeg, rep, Tr(state)(:, i))$
    $checkGoal = checkFeasibility(nextGoal, Mat, Gradients)$
    $i = i + 1$
    if $i == partSize(state)$ **then**
      $i = 0$
      $state = state + 1$
    end
    Send *checkGoal* **to** Local Planner
  end
end

3.4.2.2 Local Planner

The LP is in charge of translating the coordinates of the next goals into the commands for each actuator. In this case where the only movements are in the arm joints, to reach a certain position (defined in the Cartesian space) is only necessary the Inverse Kinematics. It obtains the position necessary in each joint, and since the motors are geared with encoders it is possible to obtain their current position.

With the API functions we can communicate easily the orders to each joint and request the following position. The function provided allow to communicate through the ROS platform and generate the desired movement. This function has an internal control to limit the force and stop in case of any contact with obstacles, as a collaborative robot necessary property.

The LP needs to communicate also with the GP to ask for new goals once the given ones are completed. To avoid any pause waiting for the position, the LP will have a buffer to store the next goal, so that it substitutes the current one as soon as it is reached.

To consider that a certain position has been reached, the end-effector must be at a certain Euclidean distance from it. This threshold helps to make the movement smoother, and specially in the movements that doesn't require precision but speed. For example, in the pick and place cases, the movement towards the new position.

Another technical consideration is that the control of the gripper grasp is controlled through another interface provided by the API, allowing us to request the distance between the fingers. This is a practical measure, and in the same sense, the movement of the wrist independently from the other joints is a very useful option.

All this connections allow the GP and the LP to work separately with different elements and compartmentalize the different problems faced. A clear view of the basic communications that take place in this control proposed is seen in Figure 3.11

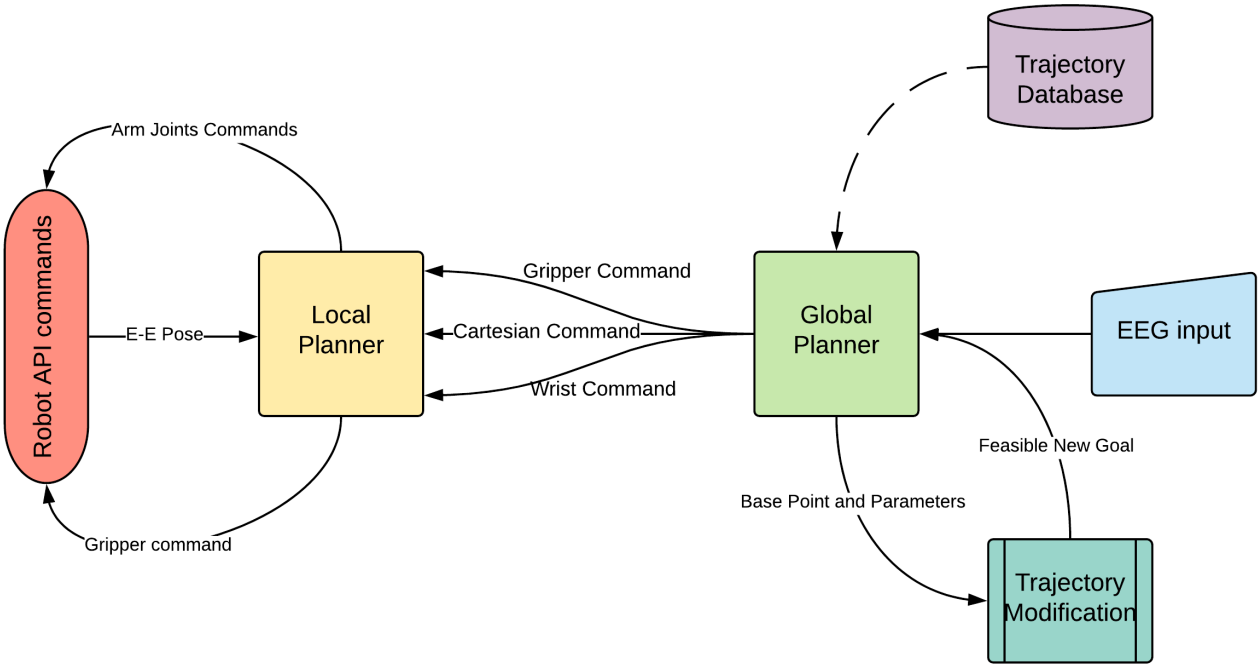


Figure 3.11: The block diagram show the different elements that provide the information involved in the planning strategy.

3.4.3 Implementation

The implementation of the motion planning was consisting mainly in the study of the different API functions and the tuning of the parameters in the planners. Even though the proposal remain the same through the implmentation, the side functions necessary required the set up of the apropiate environment.

As one of the tools used to run this iterative testing, and register the different options tested, a Git repository was established. With this tool it is possible to test different combinations and trajectories, and label the performance of each of them (as seen in the following url, where the different commits register the changes done and the effects).

`https://SergioGalve@bitbucket.org/SergioGalve/planner.git`

The main parameter to tune was the threshold used to check if a position was reached. A global solution was not very effective in this case, since some of the actions required a faster motion meanwhile other required more precision. The easiest solution was to set the threshold values as variables and set them as recorded parameters associated to each action.

Another change necessary in the implementation was to work with the gripper in order to grab certain objects. Since the GP loop (algorithm 2) only considers the Cartesian displacement of the end-effector, the motion and the grab actions were intercalated. Now the actions were launched from a main function that request different motion actions, wrist turnings and gripper activations. In a future work the ideal case would be to create an interface with the user in charge of deciding the sequence of actions.

3.4.3.1 Trajectory Design

The design of the trajectories could be considered as part of the implementation process. Once the motion planning was set up, it was necessary to find the points that would work as milestones to be interpolated (as mentioned in section 3.3.3.1).

The selection of the aproximate positions was an attempt of mimic a humans in simple tasks, but the robot required exact values. This values would be read by the robot as referenced from it's own base, representing the position and orientation of the end-effector.

To obtain this possitions the robot had a control mode that allow the user to grab the wrist and move the robot to the desired position. This function used the sensors that define the robot as a collaborative device by measuring the forces applied to it from each joint.

In this way, the robot was placed in several positions that were registered. After so the space between points was interpolated with the appropriate function. This values were stored and passed to the GP that would read them and send them to the LP for the motion execution.

3.5 Results

The results from the robot control dealt with the hardware control and the motion adaptation. In order to evaluate the performance the different problems solved will be discussed, alongside with the limitations encountered.

First of all, the method prepared for the design of the actions allow to obtain complete trajectories (divided in subparts) that the robot could follow. The problem for the definition of these paths was that they were completly dependant on the robot relative position to a known scenario. This was an inevitable consequence of the lack of sensors that would relate the robot frame reference to the environment.

The robot control achieve to elaborate two functions that can modify the trajectory in fast computation, returning a modified point. This changes were applied according to a EEG state (constantly updated), that had only one degree of freedom. The limitation of this modification is that a direction was needed to indicate how the inputs will modify the trajectory. This directions were taken intuitively as those that could be more useful for each action.

Both of the two options (Virtual Force and DMP) can be properly tuned to obtain a smooth modification of the trajectory. However, the virtual force modification generates a stronger reaction to the sign change in the EEG integral value. The DMP modification is not as reactive to these changes, depending on the scenario a smoother response could be preferable rather than a more reactive one. Both cases are compared when the input signal is the same, and in Figure 3.12 the plot illustrates the mentioned differences.

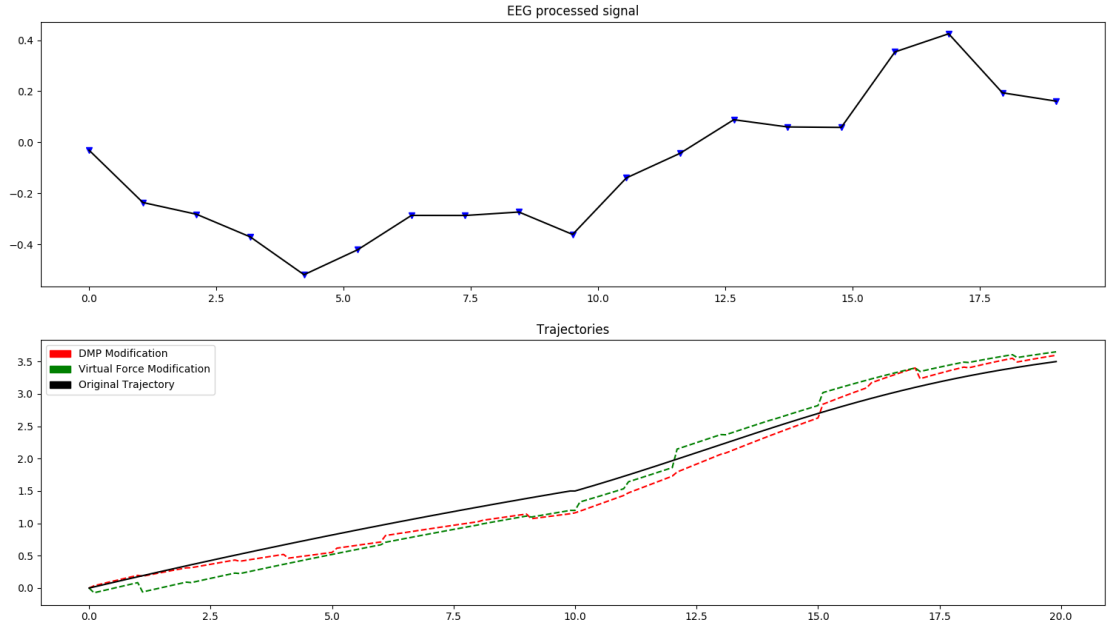


Figure 3.12: The plot above shows the EEG integral value evolving through time with a random emulated input. Below the second plot compares the behaviour of both methods to handle the trajectory modification.

Through the study of the feasibility of the workspace, an algorithm was proposed to change the future unfeasible positions. This method was only limited by the size of the discretization and even in small memory files (less than 30 Mb to obtain 4 cm resolution) all the modified trajectories were adapted to only find feasible points. However, this block brought concern on the importance of the trajectory design because those actions that were in areas of low density of feasible points, would always perform poorly.

Finally the robot control also took care of the motion planning, taking the previous elements an integrating them with the robot functions. The movement of the robot was less smooth when the precission requested were higher but still reach all the required positions.

To show an example of all the elements working together the following link contains a video with the robot performing several actions in a row with the proposed control:

`https://youtu.be/rJ9OwjIIQ6U`

Chapter 4

Final Implementation

The project have had a modular structure, where there were several blocks almost independent from each other. However, the different proposals took into account that there was necessary the integration of all the elements. In this chapter the problems and results obtained while bringing together the results of both parts, EEG signal interpretation and robot control.

At this point the robot control block is able to manipulate the robot given a base trajectory. In addition, the methods used for modifying the trajectories (DMPs and Virtual Forces) are prepared to react to an input like the one generated in the EEG signal Interpretation. Therefore, to communicate both parts is just necessary to send the results of the EEG classification to the Global Planner (as seen in the block diagram from Figure 3.11).

The internal communication between all the different parts in the robot control is done through ROS messages, where different communication channels are set. From Matlab scripts (where the EEG section was developed) is possible to send messages to ROS, being feasible to classify data meanwhile the messages are sent.

Even if the classification can be online, is required that the data acquisition is also an online process. In this case the equipment to record the EEG signal and the robot were in different scenarios. Since the objective is to allow the user to have a degree of control over the robot, both need to be together. In this way, the user can decide if the action should be modified and how should it change, by looking at the robot while it is moving.

Unfortunately if both equipment's (robot and EEG recording system) are not in the same space (where the base trajectories have been designed) the implementation can't be completed. However, it was possible to take recordings from the laboratory and pass them to the robot as if they were a real signal.

With this option is possible to observe the robot behaviour with the modified trajectories and adjust the required parameters. Specially important the tuning of those parameters that determine the magnitude of the changes that the EEG can generate. These parameters are the saturation threshold for the EEG integral value, the weight of each new sample classified over the integral value and the direction where it is applied.

By playing records of the EEG signal or using random input (as EEG substitution) it was possible to tune the parameter so that the modified trajectories are in a feasible range and the user still has a noticeable influence over the robot action.

Chapter 5

Budget

In order to complete the report, this chapter presents a brief analysis of the expenses associated to the project. A posterior review of the project goal from the resource consumption perspective.

The first point to analyze would be the description of this activity from the economic. The work developed is a research project, which ultimate aim are the results presented in this report. Since there is no marketable product there is no need of a production or distribution analysis for the moment, only the different equipment, workforce and the basic laboratory expenses consumed will be listed and priced.

The Table 5 shows the list of equipment used and to calculate the cost relative to the project we will consider the approximate amortization cost of each one. The wear caused to them with the use becomes an expense in this way, but the cost and the amortization times are estimated values.

Equipment	Cost (€)	Amortization Time (years)	Expenses (€/m)
Baxter Robot	22000	15	122,22
EEG Recording Equipment	5000	20	20,83
Laboratory Computer	600	6	8,33
Portable Computer	930	4	19,38
		Total	170,76

Table 5.1: Estimation of the costs associated to the amortization of the equipment used in the project.

The workforce would consist on an graduated engineer working 7 hours a day during a period of six months. This project was under the Master Program scholarship from IBEC, considering the net salary of 550€/month it would translate to 598 €/mmonth [18]. Finally the laboratory expenses are the general costs associated to the workplace, those values could

estimated per worker as if it were an office [19].

The total cost is obtained by adding all the montly expenses and multiplying them per 6 months as shown in Table 5.

Monthly Expenses	Cost
Amortization	170,76
Workforce	598
Cleaning Service	78,4
Electricity and water	116
Administrative	16
Monthly Total	979,16
TOTAL EXPENSES	5874,98

Table 5.2: The table presents all the monthly costs associated to the project and the total expenses of the whole period based on estimated values.

Chapter 6

Conclusions and Future Work

The use of robots in domestic environments is a possibility that is closer with the advance of robotic control systems but also, to a large extent, due to the new possibilities of interaction with them. This work has tried to study new options that help to close the gap between robots and humans.

The proposal has tested and explored different options to allow the online trajectory manipulation of a robot arm, with the use of EEG as an input. The final implementation of the different blocks (EEG and robot control) couldn't be completed but, by treating them separately, both were fully developed and they are ready to work together.

The EEG signal interpretation achieve to generate global models that could fit different users when the measures where stable. The problems in our recording process made that the classification of the new data couldn't fit those global models. However, these new data show new viable options that might make the interaction easier, like the use of MI drawing (squares and circles). The classification of these data gave poor results due to the problems with the electrodes (a bad contact made the signals more noisy and unstable) and due to the small amount of samples. Still, the results under such conditions were significantly better than the random choice, up to 47% accuracy with only 447 samples (being 33% the chance of the random classification).

The robot control manage the manipulation of the trajectories online and the execution of them b the robot. The modification methods were developed to use the EEG input and generate a change according to the user's intention (as seen in the EEG signal). The problem with the constant modification was not to make sure if the points were feasible for the robot, but to find which new position should the robot aim for. The solution was obtained from the

exploration of the whole workspace, an algorithm capable of finding the closest feasible point (to replace those who were not valid). Finally all this functions and the hardware manipulation were integrated in the Global and Local planners, capable of integrating the different elements.

The project proposed a series of subobjectives that have been fulfilled individually, but require of a scenario where both hardware devices (EEG equipment and Baxter robot) can be used to achieve the final implementation. Also, during the development of the project there were different elements that were found relevant for future work proposals:

- EEG for Action Selection: To make this application even more realistic, the user could use the same mechanisms from the EEG section, to decide among several action options for the robot (and then manipulate the trajectories freely). This new feature would require a visual interface where the options are displayed and the users can visualize what they is choosing.

- Larger Dataset: one of the problems in the project was with the data recording, where the amount of samples was reduced, with only three electrodes and from only one subject. In future cases a larger dataset with different subjects and more channels would boost the results. To achieve so a better recording method must be applied to avoid the problems with the signal observed in this project.

- Sensor Inclusion: the current control doesn't take any information of the environment appart from the force sensors, including a visual sensor would help to adapt the trajectories to the given scenario. In the case of the feasibility analysis, the inclusion of a 3D scene where the objects are identified could be an easy new utility to avoid obstacles.

- Trajectory Intelligent Design: the base trajectories define the actions since any new modification is based on them. In future works this trajectories could be design from new perspectives that take into account the limitations of the user's (in the cases where they might have certain disabilities). Also a intelligent design guided by the feasible work areas of the robot would improve the performance.

- ROS compatibility's: The robot Baxter's API was only compatible with the Indigo version of ROS, as a consequence the whole code was done within this version. At the same time Indigo is only fully compatible with the 14.04 version of Ubuntu. These limitations might be a problem in the implementation of new features and find the way to upgrade them to newer version would be beneficial.

The current results and the future perspectives discussed show that this work line has promising opportunities. Methods that make possible the collaboration between humans and robots have helped bring improvements to many fields. Among them the assistive robots need to create an intuitive environment where controlling a robot can be part of the daily activities. This project has proved that the EEG could be used for an online manipulation of the robot actions, future development might achieve a definitive integration of robots in the domestic place with this work line.

Bibliography

[1] "Dynamic movement primitives part 1: The basics." https://studywolf.wordpress.com/2013/11/16/dynamic-movement-primitives-part-1-the-basics/.

[2] A. B. J. O. B. B. Jianjun Meng, Shuying Zhang and B. He, "Noninvasive electroencephalogram based control of a robotic arm for reach and grasp tasks," 12 2016.

[3] N. I.-L. J. K. D. V. A. B. F. M. M. S. N. B. A. Sarasola-Sanz, E. López-Larraz and A. Ramos-Murguialday, "An eeg-based brain-machine interface to control a 7-degrees of freedom exoskeleton for stroke rehabilitation," 03 2017.

[4] A. S. G. Pfurtscheller, C. Brunner and F. L. da Silva, "Mu rhythm (de)synchronization and eeg single-trial classification of different motor imagery tasks," 01 2006.

[5] S. m. I. C. L. W. S. C. T. S. L. Wenchang Zhang, Fuchun Sun, "A hybrid eeg-based bci for robot grasp controlling," 10 2017.

[6] M. D. H. T. B. N. W. J. Schalk, G., "Bci2000: A general-purpose brain-computer interface (bci) system," 10 2004.

[7] M. I. Yongwook Chae, Jaeseung Jeong and I. Sungho Jo, Member, "Toward brain-actuated humanoid robots: Asynchronous direct control using an eeg-based bci," 10 2012.

[8] T. C. Technologies, *10/20 System Posiotining Manual*. 2012.

[9] G. L. H. J. I. P. M. R. M. J. M. G. P. C.-K. S. H. Goldberger AL, Amaral LAN, "Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," 2000.

[10] "Eeg motor movement/imagery dataset." https://www.physionet.org/physiobank/database/eegmmidb/.

[11] J. Fedjaev, "Decoding eeg brain signals using recurrent neural networks," tech. rep., Technische Universitat Munchen, 2017.

[12] "Train neural network for deep learning." `https://es.mathworks.com/help/deeplearning/ref/trainnetwork.html`.

[13] S. Schaal, "Dynamic movement primitives–a framework for motor control in humans and humanoid robotics."

[14] B. Siciliano, *Robotics: modelling, planning and control*. London: Springer, 2009.

[15] "Courses: Stanford artificial intelligence laboratory." `http://ai.stanford.edu/courses/`.

[16] "What does collaborative robot mean." `webhttps://blog.robotiq.com/what-does-collaborative-robot-mean`.

[17] "Ros: Core components." `http://www.ros.org/core-components/`.

[18] "Calculadora sueldo neto." `https://cincodias.elpais.com/herramientas/calculadora-sueldo-neto/#tabla_resultados`.

[19] "Office building expenses and income increased from 2015 to 2016." `https://www.buildings.com/article-details/articleid/21314/title/office-building-expenses-and-income-increased-from-2015-to-2016`.