Universitá degli Studi di Padova

Department of Information Engineering

Master Thesis in Computer Science

Hybrid Brain-Computer Interface for robots' navigation

Supervisor
Emanuele Menegatti
Universitá di Padova

Co-supervisor
Gloria Beraldo, Stefano Tortora
Universitá di Padova

Master Candidate
Michele Benini

Universitá degli Studi di Padova

Department of Information Engineering

Master Thesis in Computer Science

Hybrid Brain-Computer Interface for robots' navigation

Supervisor
Emanuele Menegatti
Universitá di Padova

Co-supervisor
Gloria Beraldo, Stefano Tortora
Universitá di Padova

Master Candidate
Michele Benini

Abstract

Brain-Computer Interface (BCI) technology allows to use brain signals as a new communication and controll channel. In the past two decades, we have seen an increase in the sophistication of BCI-driven applications. In this work we want to introduce a hybrid-BCI approach to control mobile devices for navigation. The purpose of this BCI is to enable people to drive a mobile robot in order to join relatives and friends and to participate in their activities or can be used to move a wheelchair through the space.

For this purpose we developed a hybrid Brain-Computer Interface. A hybrid Brain-Computer Interface is a BCI composed by two ore more BCIs. The proposed one is based on motor imagery and P300 evoked potential paradigms where both are used simultaneously to generate commands. The commands generated represents the directions that the user wants the robot to take.

In this work, we introduce a novel probabilistic model to integrate over time the commands coming from the two BCI systems in order to provide a unique command to the robotic device. This model can be easily integrate with a shared control approach. The commands are represented in the model as Gaussian distribution. The mean indicates the correspondent direction of the command, and the standard deviation defines the uncertainty of the command.

The model was tested in a specific application, a user pilots the Pioneer 3 AT through the laboratory. When in front of the robot there are four recognized people the user can decide to follow on of them by a P300 command and/or adjust its direction with motor imagery commands.

Sommario

Le Interfaccie Cervello-Computer (BCI) permettono di usare i segnali del cervello come un nuovo canale di comunicazione. Negli ultimi vent'anni abbiamo visto un aumento della commplessità delle applicazioni basate su BCI. In questo lavoro vogliamo introdurre un paradigma BCI di tipo ibrido per controllare dispositivi mobili. Lo scopo di questa BCI è consentire alle persone di guidare robot mobili per incontrare delle persone e partecipare alle loro attività o può essere utilizzato per spostare una seddia a rotelle nello spazio.

A questo scopo abbiamo sviluppato un interfaccia cervello-computer ibrida. Un'interfaccia ibrida è una BCI che unisce due o più paradigmi. La BCI ibrida che proponiamo si basa sui paradigmi di motor imagery e P300 in cui entrambi vengono utilizzati per generare comandi utili a pilotare il robot. I comandi generati rappresentano le direzione che un utente vorrebbe far prendere al robot.

Vogliamo introdurre un nuovo modello per rappresentare le probabilità relative alle direzioni che il robot potrebbe prendere ricevendo comandi della BCI. Questo modello può essere facilmente integrato con un approccio di shared control. I comandi sono rappresentati nel modello come distribuzioni Gaussiane dove la media indica la direzione data dal comando e la deviazione standard dipende dal tipo di comando ricevuto.

Il modello è stato testato in un'applicazione specifica, l'utente pilota il Pioneer 3 attraverso il laboratorio. Quando davanti al robot abbiamo 4 persone riconosciute, l'utente può decidere di seguire una di queste inviando un comando P300 e può regolare la direzione con i comandi di motor imagery.

Contents

1 Introduction

The aim of BCI research is to establish a new communication system that translates human intentions into control signals for an external device, such as a computer, an assistive appliance or a neuroprosthesis. Humans can send commands to such external device directly without the involvement of peripheral nerves and muscles. BCIs use signals recorded from the scalp, the surface of the cortex, or inside of the brain to enable users to communicate with computers or output devices. BCIs are widely used in various areas such as neuroscience, clinical diagnosis, rehabilitation, engineering, computer science, robotics, etc. The pipeline underlying each BCI system contains five steps: signal acquisition, signal enhancement, feature extraction, classification, and the control interface[1]. As regards the signal acquisition, there are different methods such as: electroencephalography (EEG), magnetoencephalography (MEG), positron emission tomography (PET), functional magnetic resonance imaging (fMRI), and optical imaging. EEG requires relatively simple, inexpensive and convenient equipment. The signal enhancement is usually performed by filters witch improves the signals. Features extraction and classification are commonly solved as machine learning problems, in order to identify the best signal characteristics and decision rule to discriminate between different mental tasks. This steps are strictly related to the BCI approach that we want use. Among the various approaches we can find:

- Visual Evoked Potential (VEP), caused by sensory stimulation of a subject's visual field and reflect visual information processing mechanisms in the brain.
- P300 Evoked Potential (P300), that are response of the brain to infrequent or particular stimuli, typically evoked in the EEG, as a positive peak at about 300 ms.
- Motor Imagery, where the mental imagination of movements is visible in the EEG.

The BCIs can be very inaccurate, also for simple task. With the objective to improve this communication channel, researchers start to join more BCI approaches creating hybrid BCI. Returning to the last step, the control interface is usually a simple graphic interface, it gives to the user a feedback and she/he can behave accordingly it. The purpose of this thesis is to develop a Hybrid BCI approach that join P300 signals and motor imagery signals. This Hybrid BCI approach will be applied to drive a robot through the space.

1.1 State of art

The Brain Computer Interface studies started when Hans Berger, Professor of Psychiatry at the University of Jena in Germany, discovered that electrical signal produced by the human brain could be recorded from the scalp, in 1924. Berger, in the next years, published the first 14 articles that established electroencephalography (EEG) as a basic tool for clinical diagnosis and brain research.

The first demonstrations of Brain–Computer Interface (BCI) technology occurred in the 1960s when Grey Walter used the scalp-recorded electroencephalogram to control a slide projector in 1964[2] and when Eberhard Fetz taught monkeys to control a meter needle and thereby earn food rewards by changing the firing rate of a single cortical neuron [3, 4].

In the 1970s, Jacques Vidal developed a system that used the scalp-recorded visual evoked potential (VEP) over the visual cortex to determine the eye-gaze direction in humans, and thus to determine the direction in which a person wanted to move a computer cursor [5, 6]. Vidal coined the term "Brain–Computer Interface." Since then and into the early 1990s, BCI research studies continued to appear only every few years.

In 1980, Elbert et al. showed that people could learn to control slow cortical potentials (SCPs) in scalp-recorded EEG activity and could use that control to adjust the vertical position of a rocket image moving across a TV screen [7]. In 1988, Farwell and Donchin [8] reported that people could use scalp-recorded P300 event-related potentials (ERPs) to spell words on a computer screen. Wolpaw and his colleagues trained people to control the amplitude of mu and beta rhythms (i.e., sensorimotor rhythms) in the EEG recorded over the sensorimotor cortex and showed that the subjects could use this control to move a computer cursor rapidly and accurately in one or two dimensions [9, 10].

The pace and breadth of BCI research began to increase rapidly in the mid-1990s and this growth has continued almost exponentially into the present. Conventional "simple" BCIs rely on only one signal so it has begun to study hybrid BCI that join more signal to have better precision or to complete more complex task. The work over the past 20 years has included a broad range of studies in all the areas relevant to BCI research and development, including basic and applied neuroscience, biomedical engineering, materials engineering, electrical engineering, signal processing, computer science, assistive technology, clinical rehabilitation, and human factors engineering. The central goal of BCI research and development is the realization of powerful new assistive communication and control technology for people severely disabled by neuromuscular disorders such as amyotrophic lateral sclerosis (ALS), stroke, spinal cord injury, cerebral palsy, multiple sclerosis, and muscular dystrophies.

In addition, in recent years a number of investigators have begun to explore possibilities for developing BCIs for the general population. These include systems for enhancing or supplementing human performance in demanding tasks such as image analysis or continuous attention, as well as systems for expanding or enhancing media access, computer gaming, or artistic expression.

1.1.1 State of arts in robotics

The concept of robots that helps humans has been studied a lot in recent years, in both domestic and industrial areas. Human-machine cooperation of mobile assistive robots are common in systems where the human operator cannot guarantee safe navigation due to challenging circumstances or decreased visual or cognitive capacity of the user [11, 12]. Examples of such systems are intelligent wheelchairs [13] or intelligent electrical pallet truck-like robots [11]. A fully automated system might increase safety in these systems due to high navigational capabilities in the form of precision, perception and reaction time. However, human operators are better than automated systems at interpreting complex scenarios and performing complex reasoning [12]. Therefore, an assistive system should only provide help when it is needed. If the automated system has complete authority over robot control, the user often experience that the robot is out of control and tries to reclaim the control of the robot [13]. The users rejection or disagreement of control is an unsatisfied behaviour for assistive mobile robots and may even cause dangerous behaviours [13]. Some studies show that navigation assistance should appear in a gradual and continuous manner only when it is needed [14]. The navigational assisting systems often use a shared controller to regulate the influence of human and machine control to compensate for the users' control ability. Shared control can be defined as a situation in which both a human and a machine have an effect on how that machine achieves a certain goal (i.e. navigation) [15]. Xavier Perrin [16] proposes a general black-box solution for low-throughput control of mobile robots with the intention of decreasing the users involvement in controlling the robot. At every location where a navigational decision needs to be made, the robot will stop and propose a strategy for future navigation. The user can either agree or disagree to this proposal, which would make the robot execute the proposed strategy or propose an alternative action.

Applying BCI for continuous robot control is a rather new research area with the first brain-controlled robot [17] developed in 2004 [18]. Because of the many challenges of BCI systems and specifically of the non-stationarity of EEG signals [18, 19], the brain-actuated robots have not yet seen much of outside lab environments, as this has caused unreliable behavior of the robot. However, maintaining a controlled environment, previous studies show promising results in EEG-driven robot manoeuvring by using pre-defined high- and/or low-level commands executed through discrete state transitions in finite state automatons with the assistance of external subsystems [17, 19, 20, 21]. Research in hybrid BCI systems for brain-actuated robot control is heading in a promising direction [18]. Independent of whether the additional subsystem consists of additional BCI or an intelligent navigational system, Bi et al. [18] highlights the importance of hybrid solutions for the brain-actuated robotics field. A brain-actuated robot system developed with modularity and platform independence in mind could therefore use developed subsystems to find compositions that increase performance for that specific system. For instance, Leeb et al. [22] proposes future work to implement a hybrid BCI system by adding an already developed subsystem for reliable start and stop of the robot. Additionally, Leeb et al. [22] agrees with Bi et al. [18] that hybrid approaches for robot manoeuvring is promising for overcoming existing shortcomings, further pushing the boundaries of brain-actuated robots. Siegwart et al. [23] emphasizes the importance of modularity in general mobile robots. Additionally, Beraldo et al. [24] show as the Robot Operating System (ROS) has useful tools to develop a suitable modular BCI-Robot system to mentally drive a telepresence robot.

1.2 Objective

The goal of this thesis is to provide a new hybrid BCI-based system. To achieve this goal we decide to combine P300 evoked potential and motor imagery approaches to pilot a robot through the space.

We start supposing that P300 can provide to us a specific target while motor imagery can provide only vague information about where the user wants to go. In this work, we supposed that there are a group of people that the user knows so we use them as specific P300 targets if they are in the robot's field of view. This group of people is known a priori and the robot uses their faces, detected with a face detection algorithm, to estimate their positions. While the motor imagery BCI allows the user to modify the direction that the robot has already take. We decide to encode the information provided by BCIs in a probability distribution of directions that the user could taken. Each BCI approaches give us different information about the target direction so we developed a model to join them together.

1.3 Thesis structure

This Section will summarise the content of the thesis for the next Chapters.

Chapter 2

The Brain-Computer Interface Chapter introduces the Brain-Computer Interface system, showing which kinds of BCIs exist. We want to give you a generic knowledge of what the BCI are, focusing on non-invasive BCI and its paradigms.

Chapter 3

The Material and Methods Chapter contains all the information about the devices and the libraries which we used. We describe the hardware of the BCI and the robot. The whitoolkit* is a library used to process the signal derived from the BCI and to communicate with ROS. ROS library is the most used in this thesis and allows us to develop modular code and interface whit the robot. The OpenCV library is used for all the operations that concern the images beacuse it is already provided in ROS and is simple to use.

Chapter 4

Hybrid Interface explains the model that we propose. After the mathematical presentation of the model, an example of its application is provided.

Chapter 5

In this Chapter we want to describe the fully implementation of the hybrid BCI presented before in the ROS context, explaining the nodes, the messages and the services which we used. We focus also on the Visual interface because we must give a good feedback to the user and it is necessary for P300 approach.

Chapter 6: Experiments

In the Experiments Chapter we want to explain how the model works with the help of a simulator and we want to show how the system works with simulated commands in a simulated environment.

*IAS-Lab, University of Padua

2 Brain Computer Interface (BCI)

A Brain Computer Interface (BCI) is a hardware and software communication system that allows cerebral activity to control computers or external devices. The role of the central nervous system (CNS) is to respond to incentives in the environment or in the body by producing appropriate outputs.

A brain computer interface (BCI) gives the CNS new output that is not neuromuscular or hormonal. We can define a BCI as a system that measures CNS activity and converts it into artificial output that replaces, restores, enhances, supplements, or improves natural CNS output. Thereby it changes the ongoing interactions between the CNS and the external or internal environment [25].

2.1 From Brain to Computer

The CNS integrates the received information, coordinates and influences the activity of all parts of the body. The CNS is composed of the brain and the spinal cord.

The brain is the most complex organ of the human body. The cerebral cortex contains approximately 14–16 billions neurons and in the cerebellum are estimated 55–70 billions [26]. Each neuron is connected by synapses to several thousand of other neurons. These neurons communicate with one another by long protoplasmic fibers, called axons. Axons carry trains of signal pulses, called action potentials, to distant specific cells. Physiologically, the function of the brain is to exert a centralized control over the other organs. The brain acts by generating patterns of muscle activity and by driving the secretion of chemicals, called hormones. This centralized control allows rapid and coordinated responses to changes in the environment.

CNS activity comprises the electrophysiological, neurochemical, and metabolic phenomena (such as neuronal action potentials, synaptic potentials, neurotransmitter releases, and oxygen consumption). These phenomena can be monitored by measuring electric or magnetic fields, hemoglobin oxygenation, or other parameters employing sensors on the scalp, on the surface of the brain, or within the brain. BCI records brain signals, extracts particular measures from them, and converts the features into new artificial outputs that act on the environment or on the body itself.

To measure brain activity, the methods can be divided in three main categories: invasive, partially-invasive and non-invasive measurements.

2.1.1 Invasive measurements

Invasive BCI requires surgery to implant electrodes under scalp for communicating brain signals. The main advantage is to provide more accurate reading; however, its downside includes side effects from the surgery. After the surgery, scar tissues may form which can make brain signals weaker. In addition the body may not accept the electrodes which may cause medical complications[27].

2.1.2 Partially-invasive measurements

Partially invasive BCI devices are implanted inside the skull but they are located out the grey matter. They produce better resolution signals than non-invasive BCIs, because the bone tissue of the cranium do not deflect and deform signals and have a lower risk of forming scar-tissue [28].

2.1.3 Non-Invasive measurements

The non-invasive BCI measures signals from outside of the skull. The big advantage is that is not needed any surgery, but the signals are deformed and deflected by the bone tissue of the skull that creates noise and makes it harder for a computer to interpret [29].

Electroencephalography (EEG) is the most studied non-invasive interface, mainly due to its fine temporal resolution, ease of use, portability and low set-up cost. However it is susceptible to noise. Electroencephalography is an electrophysiological monitoring method to record electrical activity of the brain. This method uses electrodes placed along the scalp. EEG measures voltage fluctuations resulting from ionic current within the neurons of the brain.

The brain's electrical charge is maintained by billions of neurons which is electrically polarized. Neurons are constantly exchanging ions. Ions of similar charge repel each other. When many ions are pushed out of many neurons at the same time, they can push their neighbours, who push their neighbours, and so on, in a wave. This process is known as volume conduction. When the wave of ions reaches the electrodes on the scalp, they can push or pull electrons on the electrodes. From metal conduction in the electrodes, the push and pull of electrons can easily be measured by a voltmeter. Recording these voltages over time gives us the EEG.

The electric potential generated by an individual neuron is far too small to be picked up by EEG. EEG activity reflects always the summation of the synchronous activity of thousands or millions of neurons that have similar spatial orientation. If the cells do not have similar spatial orientation, their ions do not line up and create waves to be detected. Pyramidal neurons of the cortex are thought to produce the most EEG signal because they are well-aligned and fire together. Activity of the farthest neurons from the electrodes is more difficult to detect because voltage field gradients reduce with the square of distance.

Scalp EEG activity shows oscillations at a variety of frequencies (Table. 2.1). Several of these oscillations have characteristic frequency ranges, spatial distributions and are associated with different states of brain functioning. These oscillations represent synchronized activity over a network of neurons [30].

Band	Frequency (Hz)
Delta	< 4
Theta	4–7
Alpha	8–15
Beta	16–31
Gamma	> 32
Mu	8–12

Table 2.1: Table of types of waves

The EEG is recorded by electrodes, usually using a conductive gel for better transduction of charge between scalp and electrode. The electrodes placed over the scalp are commonly based on the International 10-–20 system [31], which has been standardized by the American Electroencephalographic Society (Figure. 2.1). The 10–20 system uses the nasion, located at the top of the nose at the same level as the eyes and the inion, which is found in the bony lump at the base of the skull. The transverse and median planes divide the skull from these two points. The electrode locations are determined by marking these planes at intervals. The letters in each location correspond to specific brain regions in such a way that A represents the ear lobe, C the central region, Pg the nasopharyngeal, P the parietal, F the frontal, Fp the frontal polar, and O the occipital area.

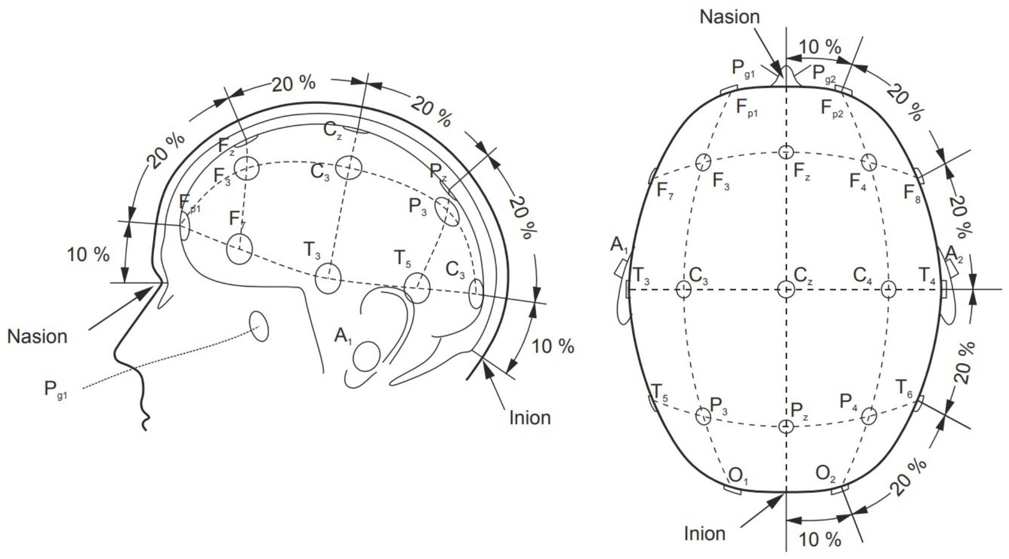

Figure 2.1: 10-20 system defining the electrodes positions.

2.2 Non-invasive BCIs paradigms

BCIs are used to interpret intentions of the user monitoring cerebral activity but brain signals involve numerous cognitive tasks. A lot of this phenomena are still incomprehensible but some have been decoded and people may learn how to modulate these phenomena in order to control the BCI systems.

Current BCIs can be classified according to the nature of the signals that they use as input:

- Endogenous/Exogenous: Endogenous are related to the self-paced control of electrophysiological activity, such as amplitude in a specific frequency band in EEG recorded over a characteristic cortical area. Therefore the BCI is based only on spontaneously generated brain patterns. While exogenous ones rely on an external source that generates specific brain patterns. Therefore the BCI is based on brain responses to external stimulus [32].
- Dependent/Independent: dependent BCI is essentially an alternative method for detecting messages carried in natural CNS outputs. Although it does not give the brain a new output because it is independent from natural outputs, it may still be useful. In contrast, an independent BCI does not depend on natural CNS output; in independent BCIs, muscle activity is not essential for generating the brain signals that the BCI uses. For example,

in BCIs based on EEG sensorimotor rhythms (SMRs), the user may employ mental imagery to modify SMRs so as to control the BCI output [32, 33].

Numerous studies have described many brain signals that might serve as control signals in BCI systems. We want to discuss the following:

- Visual Evoked Potentials, in particular Steady-State Visual Evoked Potentials, exogenous, independent
- P300 Evoked Potentials, exogenous, independent
- Motor Imagery, endogenous, independent

2.2.1 Visual Evoked Potentials (VEPs)

Visual Evoked Potentials are caused by sensory stimulation of a subject's visual field and reflect visual information processing mechanisms in the brain. Stimulation of the central visual field evokes larger VEPs than peripheral stimulation. EEG-based BCIs systems with visual evoked potentials have been studied extensively [34, 35].

In a VEP based BCI, each target is coded by a unique stimulus sequence which evokes a unique VEP pattern. So when an user is visually focused on a point, a VEP based BCI can identify a target. In order to ensure a reliable identification, VEPs must come from different stimulus sequences, that should be orthogonal or near orthogonal to each other. Depending on the specific stimulus sequence modulation approach used, current VEP based BCIs can be organized into three categories:

- Time modulated VEP (T-VEP) BCIs [36, 37]
- Frequency modulated VEP (F-VEP) BCIs [38, 39]
- Pseudorandom code modulated VEP (C-VEP) BCIs [40, 41]

Steady-State Visual Evoked Potentials

SSVEP are signals that are natural responses to visual stimulation at specific frequencies. When the retina is excited by a visual stimulus ranging from 3.5 Hz to 75 Hz [42], the brain generates electrical activity at the same (or multiples of) frequency of the visual stimulus.

Regan started experimenting with long stimulus trains, consisting of sinusoidally modulated monochromatic light[43]. These stimuli produced a stable VEP of small amplitude, which could be extracted by averaging over multiple trials. These EEG waves were named "steady-state" visually evoked potentials (SSVEPs) of the human visual system.

VEPs elicited by brief stimuli are usually transient responses of the visual system. Transient evoked potentials are responses of the system under study to sudden changes (jumps or steps) in the input [44].

According to the original definition, steady-state potentials are to be distinguished from transient potentials, because their constituent discrete frequency components remain closely constant in amplitude and phase over a long time period [45]. This "constant" characteristic is not a time-domain observation. What remain constant is the spectral distribution (in the frequency domain), not the raw EEG amplitude in time domain: SSVEP contains stationary periodic oscillations. Consequently, the amplitude distribution of the spectral content of SSVEP, with characteristic SSVEP peaks, remains stable over time.

The shape of the response in time domain usually is not sufficient to distinguish SSVEPs from transient VEPs; instead, spectral response in the frequency domain is visible. The transient response is a complex resonance response to a stimulus. When the stimulus is repeated, this complex response should correspondingly be repeated; this combined response may become organized (with stationary periodic oscillations) or disorganized (without stationary periodic oscillations), and this shows the difference between transient VEPs and SSVEPs.

2.2.2 P300 Evoked Potentials (P300)

The P300 (P3) wave is an event related potential (ERP) component elicited in the process of decision making. Rare or particular stimuli typically evoked in the EEG, over parietal cortex, a positive peak at about 300 ms [46, 47, 48], this peak was called P300. The use of the P300 as a control signal has the great advantage of being a rapid process, it has fairly high communication speed and no training is required because it is a process naturally present in all people. While the main downside is that it is a rather mutable process. Phenomena such as habit or poor attention of the subject can lead to deterioration and therefore to more difficult detection.

P300 is usually used as speller for people who cannot communicate in other way[49, 50, 51]. The user faces a matrix of letters, numbers, and/or other symbols or commands (Figure. 2.2 [51]). Periodically, a single row or column flashes; and, in a complete trial of flashes, each row or column flashes twice.

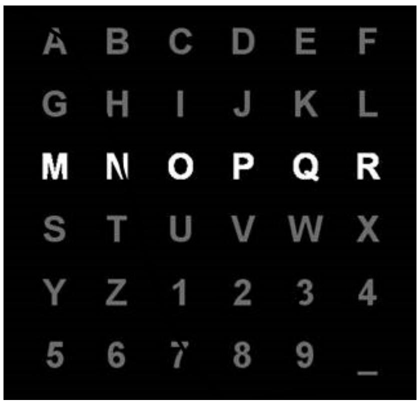

Figure 2.2: Matrix for P300 speller.

The user selects one the symbols. When this symbol flashes, the P300 wave is evoked. EEG over parietal cortex is digitized, in order to detect the P300 wave, the average response to each row and column is computed and P300 amplitude for each possible choice is computed. P300 is prominent only in the responses elicited by the desired choice and the BCI uses this effect to determine the user's intent. P300-based BCI could yield a communication rate of one word (i.e. 5 letters) per minute and also suggest that considerable further improvement in speed should be possible [33].

2.2.3 Motor Imagery

Mental imagination of movements involves similar area or function of the brain which are involved in programming and preparing actual movements [52]. The main difference between performance and imagery is that in the latter case execution would be blocked at some corticospinal level [53].

Preparation and planification of the movement leads to amplitude suppresion of the signal recorded by EEG, called event-related desynchronisation (ERD). This is followed by an amplitude enhancement, called event related synchronisation (ERS).

In the alpha/mu band, the desynchronisation starts 2.5 seconds before movement, it peaks after movement-onset and recovers back to baseline within a few seconds. In the beta band, the desynchronisation is only short-lasting, immediatly followed by synchronisation reaching a maximum in the first second after the movement. In the gamma band, synchronisation reaches a maximum right before movement-onset, but these gamma oscillations are rarely found in a human EEG [54]. The most prominent EEG changes will be localised over the corresponding primary sensorimotor cortex contralateral to the movement[55].

2.3 Closed-loop BCI System

Brain Computer Interfaces (BCI) usually are based on detecting an “intention to act”in the brain and transforming this intention into actual actions. The visual and proprioceptive feed-back that results from the systems’ generated action following the subjects’ captured intention, is thought to close the BCI loop. Closing the loop is a key feature of BCI-systems that allows a better control on the system.

When BCIs form a closed loop, they include five elements, as it is shown in Figure 2.3*. These are “Control paradigm,” “Measurement,” “Processing,” “Prediction” and “Application”[56]. Through these five steps, the BCI interprets a user’s intention or mental state and uses the information to run the application. Until termination of the system, this closed loop is repeated between the user and the application, with the four modules forming an interface between them. The

*Image from Researchgate.net

details of the BCI elements are:

- Control paradigm: To convey information to the system, the user has to execute a particular task that is related to the used BCI paradigm. According to the specific task, in the brain signal we expect a particular features that the system tries to detect. However, BCI requires a “Control paradigm” for which the user is responsible. As examples we presented P300 paradigm, motor imagery paradigm and Steady-State visual evoked potentials.

- Recording: Brain signals can be recorded invasively or non-invasively.

- Processing: The brain signals are processed to maximize the signal-to-noise ratio and select target features. In this step, various algorithms are applied, such as spectral and spatial filtering, to reduce artifacts as well as to extract informative features. The selected target features are used as inputs for classification or regression modules.

- Prediction: This step makes a decision regarding the user’s intention or quantifies the user’s level of emotion and mental states. For prediction, classifiers are usually employed, such as threshold, linear discriminant analysis, support vector machine, or artificial neural network.

- Application: Once the user’s intention is determined in the prediction step, the output is converted to an action. The user finally receives a feedback from the application. The feedback is essential because the user can understand how the system works and improves its control on it. In some cases an user can learn how to control its brain signals to better manage the system.

The use of closed-loop interaction with biological nervous systems for observation and control purposes goes back to the beginnings of electrophysiology in the 1940s when the voltage clamp technique was developed [57, 58]. Later on, the dynamic clamp technology to implement artificial membrane or synaptic conductances [59, 60] has produced many examples of successful closed-loop interactions with neural systems at the cellular and circuit levels (see [61, 62, 63, 64]).

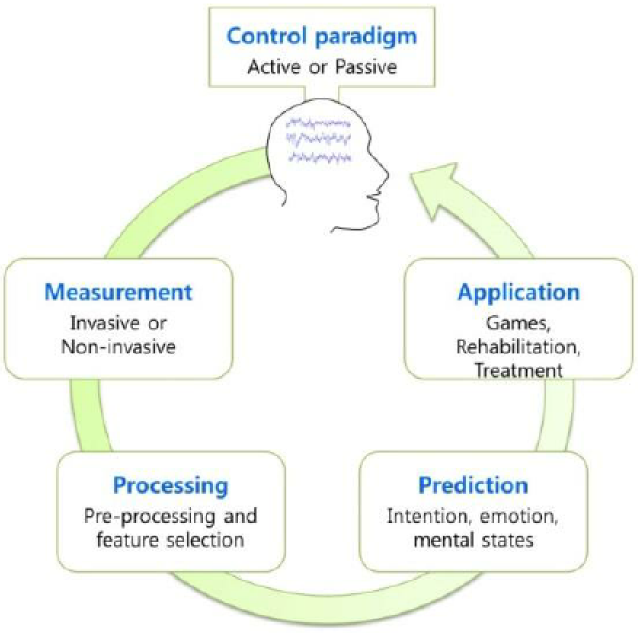
Control paradigm
Active or Passive
Measurement
Invasive or Non-invasive
Application
Games, Rehabilitation, Treatment
Processing
Pre-processing and feature selection
Prediction
Intention, emotion, mental states

Figure 2.3: Closed-Loop BCI System

2.4 Hybrid BCIs System

The purpose of this thesis is to propose a new Hybrid Brain-Computer Interface system based on P300 and motor imagery.

A hybrid BCI (hBCI) is composed of two or more BCIs. This hBCI detects at least two brain patterns in a simultaneous [65, 66] or sequential manner [67, 68]. In a simultaneous hybrid BCI, both systems process in parallel. Input signals used in simultaneous hybrid BCIs can be two or more different sources. One of them include brain signals, the other can be also an another kind of input.

In sequential hybrid BCIs, the output of one system is used as the input of the other system. With the use of two or more brain patterns, hybrid BCIs can achieve specific goals more effectively than conventional BCI systems [69]. Allison et al. demonstrated that by detecting MI and SSVEP simultaneously, classification accuracy can be improved, especially for BCI-blind subjects [65]. Pfurtscheller et al. proposed a hybrid BCI, where an MI-based brain switch was used to turn ON/OFf an SSVEP-based BCI [67].

3

Materials and Methods

In this Chapter we want to introduce libraries and devices that we used. These tools are used to create the software that implements the hybrid interface model described in the Chapters 4 and 5.

3.1 EEG Device

We use g.USBamp* to read the electroencephalogram from the user's scalp. G.USBamp is a biosignal amplifier with high-performance and high-accuracy and an acquisition and processing system.

The system includes: amplifiers, acquisition software and analysis software. G.USBamp is USB enabled and comes with 16 simultaneously sampled biosignal channels with 24 bits. A total of 4 independent grounds guarantee no interference between the recorded signals. The amplifier connects to the USB socket on the PC and can be used for data recording. A synchronization cable allows that all devices are sampling with exactly the same frequency. The amplifier has an input range of ± 250 mV, which allows the recording of DC signals without saturation.

*www.gtec.at/Products/Hardware-and-Accessories/g.USBamp-Specs-Features

Figure 3.1: gUSBamp EEG device.

3.1.1 Brain-Computer Interface System

In this Section, we explain our BCI system. In details we specify for each of the paradigm we consider the methods we applied in terms of measurement, processing and classification.

Closed-loop in motor imagery

- Control paradigm: motor imagery
- Recording: non-invasive BCI EEG based receiving in input 16 channels placed over the pre-motor area and the primary motor area.
- Processing: : EEG was preprocessed by applying a Laplacian spatial filter. The Power Spectral Density (PSD) of the signal was continuously computed via Welch's algorithm (1 second sliding window, 62.5 ms shift) in the frequency range from 4 to 48 Hz (2 Hz resolution).
- Prediction: the most discriminative features (channel-frequency pairs, subject-specific) were extracted by the power spectral density by Canonical Variate Analysis (CVA) and classified online by means of a Gaussian classifier previously trained during the calibration phase.
- Application: please refer to Chapter 5.

Closed-loop in P300

- Control paradigm: P300 evoked potential
- Recording: non-invasive BCI EEG based receiving in input 16 channels placed principaly over temporal and occiptal areas.
- Processing: the signal inside the 0.45 s time-window epoch after every stimulus is acquired and a butterworth 4th order digital band pass filter in the range 1-24 Hz is applied, to remove baseline drift and high-frequency noise. Then, a common average reference (CAR) filter is performed and the signal is decimated by a factor of 8. To reduce the effect of eye-blink, eye-movement and muscle artefact, the signal from each channel is winsorized, by computing the 10th and 90th percentiles of the signal amplitude and by replacing every sample outside this range with the value of 10th and 90th percentile, respectively. Finally, a z-score normalization is applied to account for trial-to-trial and day-to-day variability.
- Prediction: the resulting samples for each channel were concatenated, creating a features vector for each trial. Once this set of features was extracted, to classify the features we applied the Bayesian Linear Discriminant Analysis (BLDA), that was extensively studied for P300 classification problems.
- Application: please refer to Chapter 5.

3.2 Mobile Robot Pioneer 3 AT

The Pioneer 3 AT[†] is a compact, computer-controlled mobile robot for research on self guided system, exploration of unknown environments and indoor applications on mobile robotics.

Pioneer 3 AT offers an embedded computer option, opening the way for: on-board vision processing, Ethernet-based communications, laser, DGPS, and other autonomous functions. The Pioneer 3 AT uses 100 tick encoders with inertial correction recommended for dead reckoning to compensate for skid steering. Its sensing is extend with laser-based navigation options, integrated inertial correction to compensate for slippage, GPS, bumpers, gripper, vision, stereo rangefinders, compass and a rapidly growing suite of other options.

[†] $www.generationrobots.com/en/402397-robot-mobile-pioneer-3-at.html$

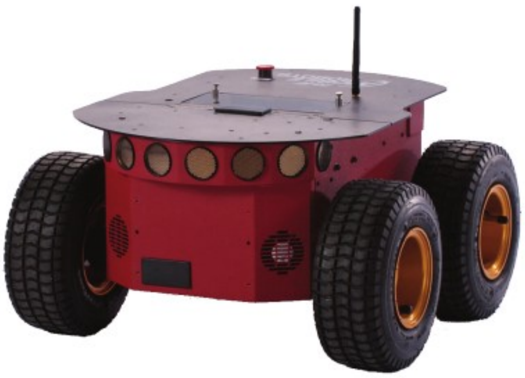

Figure 3.2: Mobile Robot Pioneer 3 AT.

The robot base includes Pioneer SDK plus optional onboard PC which can be drove and controlled by keys, wireless or tethered joystick or software whole-platform velocity control; communicate robot state control information including (x,y,theta) position estimate, user I/O, and battery charge data, run programs and simulations.

ROS offers also a set of packages that makes easily to implement program that use the pioneer and its sensors.

3.3 Development Environment

3.3.1 Robot Operating System (ROS)

The Robot Operating System[‡], also known as ROS, is an open-source, meta-operating system for the robot. It provides the services of an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. In this work we use ROS Kinetic, LTS version.

‡ $www.ros.org/about-ros/$

The reason why this operating system is spreading more and more, is because of its innovative way of describing and handling the many parts that are going to characterize the final behaviour of the intended robot. This result has been achieved with a modular programming method and stimulating collaborative work. The ROS framework is easy to implement in any modern programming language. It has been implemented in Python, C++, and Lisp. It has also experimental libraries in Java and Lua. ROS has three levels of concepts: the Filesystem level, the Computation Graph level, and the Community level. These levels and concepts are summarized below.

ROS Filesystem Level

The Filesystem level concept mainly covers ROS resources, that are present on disk, such as:

- Packages: Packages are the main unit for organizing software in ROS. A package may contain ROS runtime processes (nodes), a ROS-dependent library, datasets, configuration files, etc. that is organized together.
- Metapackages: Metapackages are specialized Packages which only serve to represent a group of related other packages.
- Package Manifests: Manifests (package.xml) provide metadata about a package.
- Repositories: A collection of packages which share a common VCS system.
- Message (msg) types: Message define the data structures for messages sent in ROS.
- Service (srv) types: Service define the request and response data structures for services in ROS.

ROS Computation Graph Level

The Computation Graph is the peer-to-peer network of ROS processes that are processing data together. The basic Computation Graph concepts of ROS are nodes, Master, Parameter Server, messages, services, topics, and bags, all of which provide data to the Graph in different ways.

- Nodes: are processes that perform computation. ROS is designed to be modular at a fine-grained scale; a robot control system usually comprises many nodes.

- Master: The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.

- Parameter Server: The Parameter Server allows data to be stored by key in a central location. It is currently part of the Master.

- Messages: Nodes communicate with each other by passing messages.

- Topics: Messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by publishing it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic.

- Services: The publish / subscribe model is a very flexible communication paradigm, but its many-to-many, one-way transport is not appropriate for request / reply interactions, which are often required in a distributed system. Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply. ROS client libraries generally present this interaction to the programmer as if it were a remote procedure call.

- Bags: Bags are a format for saving and playing back ROS message data. Bags are an important mechanism for storing data, such as sensor data, that can be difficult to collect but is necessary for developing and testing algorithms.

The ROS Community Level

The ROS Community Level concepta are ROS resources that enable separate communities to exchange software and knowledge. These resources include: ROS Distributions that are collections of versioned stacks that you can install, repositories, ROS relies on a federated network of code repositories, where different institutions can develop and release their own robot software components. Others resources are the ROS Wiki, ROS Answers, A QA site for answering your ROS-related questions and the ros.org Blog provides regular updates, including photos and videos.

The management of this three levels of concepts makes ROS a very useful tool, in particular it makes a modular programming easier, ideal for robotics.

3.3.2 OpenCV

OpenCV (Open Source Computer Vision Library)[§] is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now.

3.3.3 Whitoolkit

The whitoolkit is a software library developed by Intelligent Autonomous Systems Laboratory of the University of Padua. This library contains functions and methods to interface with g.USBamp. Whitoolkit processes and classifies EEG signals then it sends the results of classification to ROS nodes.

[§] *opencv.org/about/*

4 Hybrid Interface

In this work we achieve a Hybrid Brain-Computer Interface by combining two types of BCI paradigms to send the commands: P300 evoked potential and motor imagery. The model that we propose is based on Gaussian distributions created by the user's intention. The user wants to send a command but sometimes the commands are not accurate, so we can only suppose what is the intention of the user. According to the type of BCI approach used, we can determine and represent a level of uncertainty of the user's command. Any commands can be represented as two Gaussians. The first Gaussian is related to the first command and the second one shows how commands are modified by the previous one. Both the Gaussians were represented as a vector of probability. This probabilities are multiplied by a ration, that depends from the Gaussian. Once both the Gaussian were scaled, they were summed component to component. Finally, the user sends commands to the robot.

4.1 Model

In this Section we explain the probabilistic model which the hBCI is based on. Let $I \subset R^n$ the space that represents all the possible commands of the user. Let $\mathcal{N}(x,\sigma)$ a Gaussian distribution with mean $x \in I$ and standard deviation $\sigma \in R^n$. Let $x_{t-1} \in I$ the mean of the previous Gaussian of the previous command and

σ_{t-1} its confidence. Let $\mu_1, \mu_0 \in [0,1]$ such that $\mu_1 + \mu_0 = 1$. If we receive a new command, represented with a Gaussian distribution with mean x_1 and some level of confidence σ_t, so we can obtain the distribution of intentions D_t as:

$$D_{t|t-1:t} = \mu_0 \cdot \mathcal{N}(x_t, \sigma_t) + \mu_1 \cdot \mathcal{N}(x_{t-1}, \sigma_{t-1}) \tag{4.1}$$

This model does not keep track of the history of the commands, but it includes only the information on the last commands. To include a memory term, we can extend the model as follow. We insert in the calculus the complete previous distribution multiplied to a new $\mu_2 \in R^n$ such that $\mu_0 + \mu_1 + \mu_2 = 1$. So, to calculate the commands distribution at time t we can use:

$$D_{t|t_0:t-1} = \mu_0 \cdot \mathcal{N}(x_t, \sigma_t) + \mu_1 \cdot \mathcal{N}(x_{t-1}, \sigma_{t-1}) + \mu_2 \cdot D_{t-2|t_0:t-3} \tag{4.2}$$

In this way we can keep memory of all the past commands and manage better the result of the past command and the last command assuming that they are the most important.

4.2 From model to robot control

In this work, the model is applied to find the direction that the user wants the robot follow. We have a mono dimensional space that is represented by a scale of degree, from 0 to 359. These degrees are absolute directions, not depending on the robot reference system. Above the space, there is a probability distribution. Let X be the space, $X \subset R$, and $\Sigma \subset R$ the space of possible standard deviation. The direction that the robot takes is where the probability distribution is maximized.

We have two types of commands, that derived from P300 and motor imagery. In our model, we assume hypothesis that commands coming from P300 are less frequent but with less uncertainties, while motor imagery commands are more frequent but less reliable.

For a better explanation of the model you will find a representation in the following images. The images derive from a simulation code that you can find in: GitHub[*].

[*] $https://github.com/michelebenini/HybridBCI-Test/tree/master/2-BCIcmdSim$

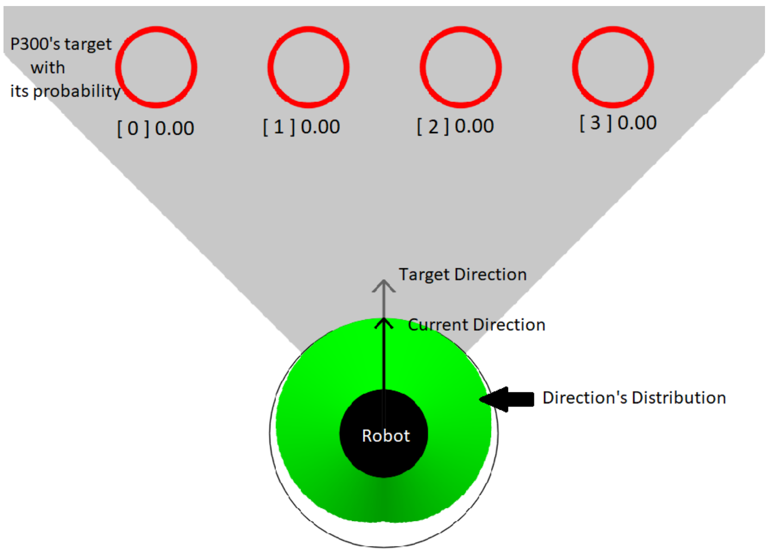

Figure 4.1: This image represents the simulation interface in the start configuration of the robot.

- red circle: targets that are able to generate P300, under each circle there is an id and the associated probability
- black circle: robot.
- black arrow: current direction of the robot.
- green part: probability distribution over the directions
- grey arrow: mode of distribution, that represents the target direction
- grey background: robot's field of view

4.2.1 No commands

When the user does not send commands we can assume that the current direction is the right direction because the user must maintain the focus while robot moves. So, there is no new commands, $\mu_0 = 0$, and the past direction must be more important. The formula becomes:

$$D_{t|t_0:t} = \mu_1 \cdot \mathcal{N}(x_{t-1}, \sigma_{t-1}) + \mu_2 \cdot D_{t-2|t_0:t-3} \tag{4.3}$$

This type of distribution is applied when the user does not send commands for a specific amount of time.

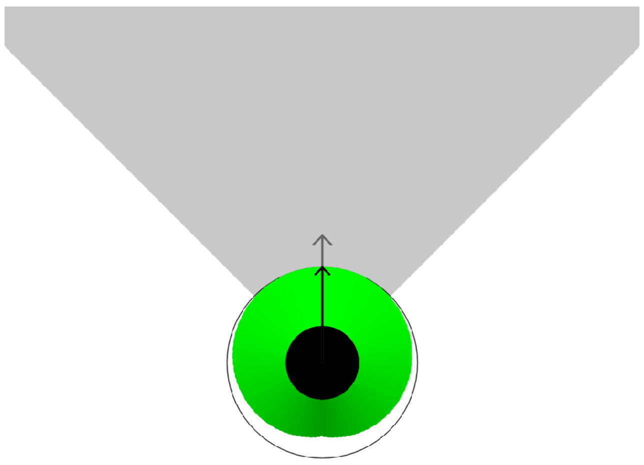

Figure 4.2: This image represents the situation before that the no commands Gaussian is added to the system.

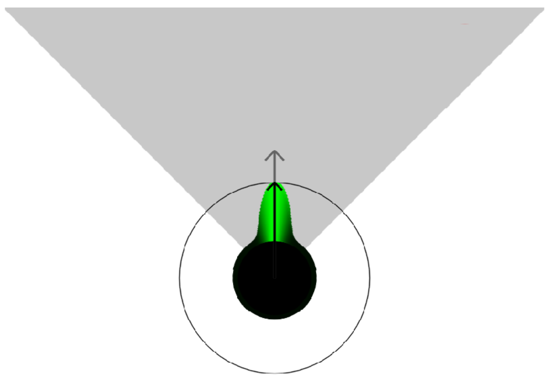

Figure 4.3: In this Figure we can see the direction-based probability distribution when the robot moves. The distribution is created starting from the Figure. 4.2 and applying no-commands five times.

4.2.2 Motor-imagery events in the model

The second type of commands derive from the motor imagery. For this type of commands we use the complete model:

$$D_{t|t_0:t} = \mu_0 \cdot \mathcal{N}(x_t, \sigma_t) + \mu_1 \cdot \mathcal{N}(x_{t-1}, \sigma_{t-1}) + \mu_2 \cdot D_{t-2!t_0:t-3} \tag{4.4}$$

In this case σ_t have to be big to enlarge the past intention's confidence and the μ_1 must be quite small because of the low voluntariness of the command. The x_{t+1} will be equals to $x_t \pm d$, $d \in X$ represents how many degree the user wants to turn right (+) or left (-) respectively on the motor imagery command. Example is shown in figure. 4.5.

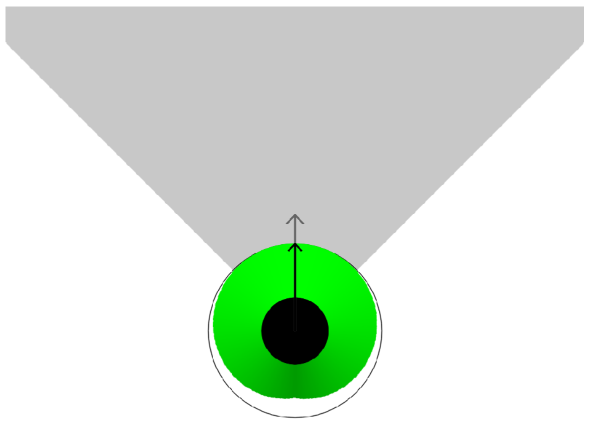

Figure 4.4: This image represents the situation before that the motor imagery Gaussian is added to the system.

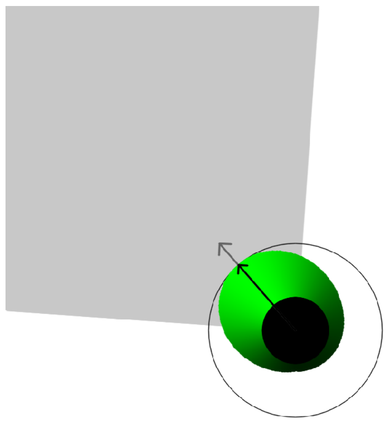

Figure 4.5: We can see the direction-based probability distribution. To obtain this distribution we had the start configuration Figure. 4.4 and we sent a left motor imagery command.

4.2.3 P300 events in the model

As we said in the Chapter 3, P300 wave is generated from visual target. In this thesis we consider the visual target as people. From the P300's output, if we have N targets in front of the robot, we receive N probabilities p_i each one related to one person and its position. The P300 command was codified according to the Formula 4.6 the Gaussian correspondents to the new input has a target-dependent standard deviation (Formula. 4.5) and it has mean equal to the direction of the this target.

We use the information provided by the P300 commands to insert a level of uncertainty on the Gaussian's standard deviation. The standard deviation σ_t becomes a function of all the probability returned by the P300. Let define $p_i \in [0, 1]$ the probability returned by the BCI that is relative to the person i. So $\sum_{i=11}^{N} p_i = 1$, without loss of generality, we can assume that $p_0 > p_1 > ... > p_N > 0$. Example are shown in Figures. 4.6, 4.7, both the Figures are obtained starting from the same situation of the last commands.

$$\sigma_t(p) = \sigma_{p300} \cdot (1 + e^{\frac{2p_1}{p_0}}) \tag{4.5}$$

Where σ_{p300} is a default value. So, a P300, changes the distribution over the intended direction as follow:

$$D_{t|t_0:t} = \mu_0 \cdot \mathcal{N}(x_t, \sigma_t(P)) + \mu_1 \cdot \mathcal{N}(x_{t-1}, \sigma_{t-1}) + \mu_2 \cdot D_{t-2|t_0:t-3} \tag{4.6}$$

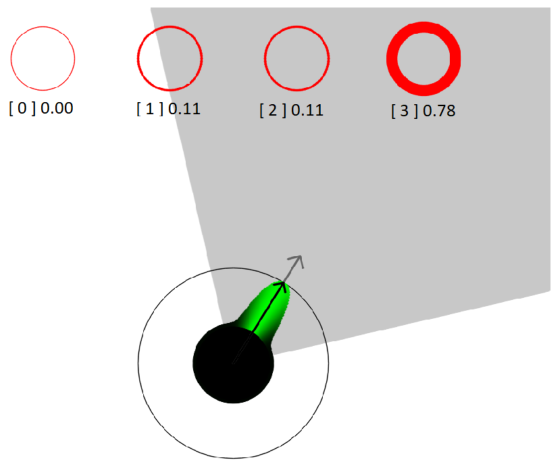

Figure 4.6: We can see the direction-based probability distribution when a user sends a P300. The P300 was sent with probability on the target [3] equals to 0.7 and the other targets equal to 0.1.

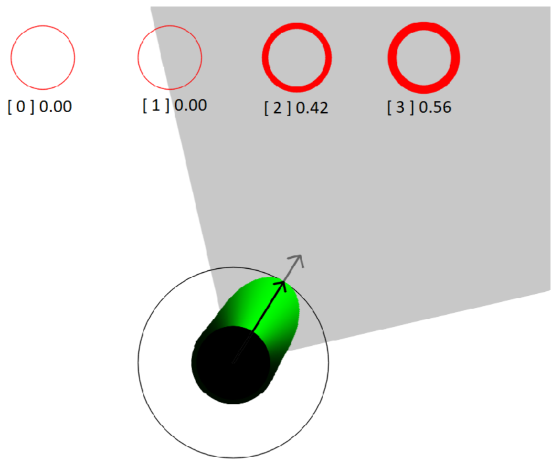

Figure 4.7: We can see the direction-based probability distribution when a user sends a P300. The target [3] has probability 0.56, the target [2] 0.44 while the others 0.0.

4.3 Parameters of simulation

In the following Table 4.1 you can find all the parameters used for the simulation. The section of the Table relative to the initial command represents the starting situation.

Initial command						
x_t	σ_t	μ_0	x_{t-1}	σ_{t-1}	μ_1	μ_2
180	180	1	0	0	0	0
No-Commands						
x_t	σ_t	μ_0	x_{t-1}	σ_{t-1}	μ_1	μ_2
0	0	0	180	10	0.03	0.97
Motor Imagery						
x_t	σ_t	μ_0	x_{t-1}	σ_{t-1}	μ_1	μ_2
180-45	50	0.5	180	100	0.3	0.2
P300						
x_t	σ_t	μ_0	x_{t-1}	σ_{t-1}	μ_1	μ_2
210	5	0.7	0	0	0	0.3

Table 4.1: Table of simulation data.

5 Hybrid Interface Implementation

In this Chapter we explain how the system was implemented. We used ROS to implement the system, so its modularity allows us to create nodes that have to carry out only one task and communicate each others through messages. We had also developed protocols that allow the BCI to publish on ROS topic. In this way we can see the BCI system that receives, processes and classifies the brain's signals as a simple ROS node that publish and receives messages.

5.1 From simulation to real implementation

As first we have to explain the biggest differences between the simulation and the real implementation.

In the simulation the position of the robot is simple to calculate and the calculus is accurate. In the real implementation we use the ***nav_msgs/Odometry*** message provided by ROS. The robot sends a ***nav_msgs/Odometry*** message which stores an estimate of the position and velocity of a robot in free space. The pose in this message corresponds to the estimated position of the robot in the global frame along with an optional covariance for the certainty of that pose estimate.

The second biggest difference between the simulation and the real implementation is the position of the P300 targets. In the simulation we know their exact positions. In the real implementation we do not need the exact position, also because it is very hard to calculate. In the system we know the robot field of view's properties and where the robot is watching. We used the field of view's degrees to divide the image in vertical parts, where each one correspond to a degree offset. The central part corresponds to the robot's current direction that we can convert in absolute degree with the information provided by the $nav\_msgs/Odometry$ messages (Figure. 5.1) using the method $tf :: getYaw(orientation)$ provided by the tf ROS package.

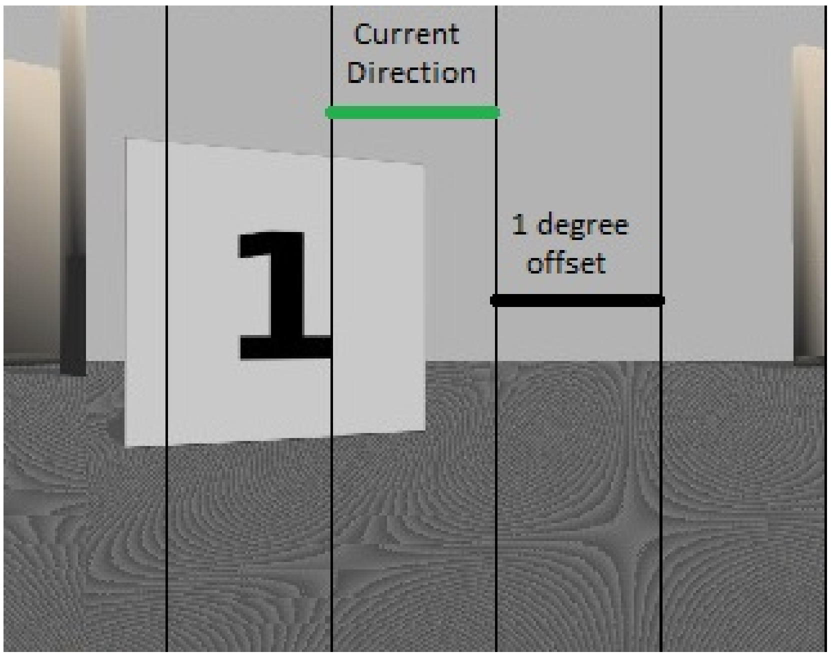

Figure 5.1: Supposing the field of view of the robot corresponds to five degrees, we divide the full image in five vertical parts (black lines) and then we calculate the offset of the target we wants. (e.g. the one has an offset of -1).

So we calculate the offset in degree between the field of view's center and the P300 target dividing the image of the robot's cam by the degree of the field of view creating some buckets of one degree each one. We watch in which bucket the P300 target's image is centered and we calculate the offset as the difference between the degree correspondent to the full image center's bucket minus the degree correspondent to the P300 target's bucket.

5.2 ROS structure

The structure of this system is complex (Figure. 5.2). We can divide the structure in two main components:

- Hybrid BCI's System: this part contains the hybrid Brain Computer Interface, the `CNBI loop` receives, processes and classifies the brain's signals, the handlers get the messages from the classifiers and they transform these messages to commands for the `Hybrid BCI` node. The handlers deal with the visual interface to close the BCI loop and the `P300 Handler` calls also a service that executes a face classification algorithm to take the P300 targets. The `Hybrid BCI` node incorporates the model presented in Chapter 4. The `CMD Simulator` node make the system start and it can simulate the BCI commands without using the `CNBI Loop` and the Handlers.

- Robot's System: this part is simpler, the Move node receives the direction distribution from the `Hybrid BCI` node and transforms it to commands that moves the robot. The `View Distribution` node is an optional node that allows the user to see the directions distribution with the shape of the simulations. As Robot we identify the robot's or the simulation software that allows us to communicate with it through messages and topics.

All this nodes and services exchange messages through specific topic. In the following sections we explain extensively both the part of the system with their communication channels.

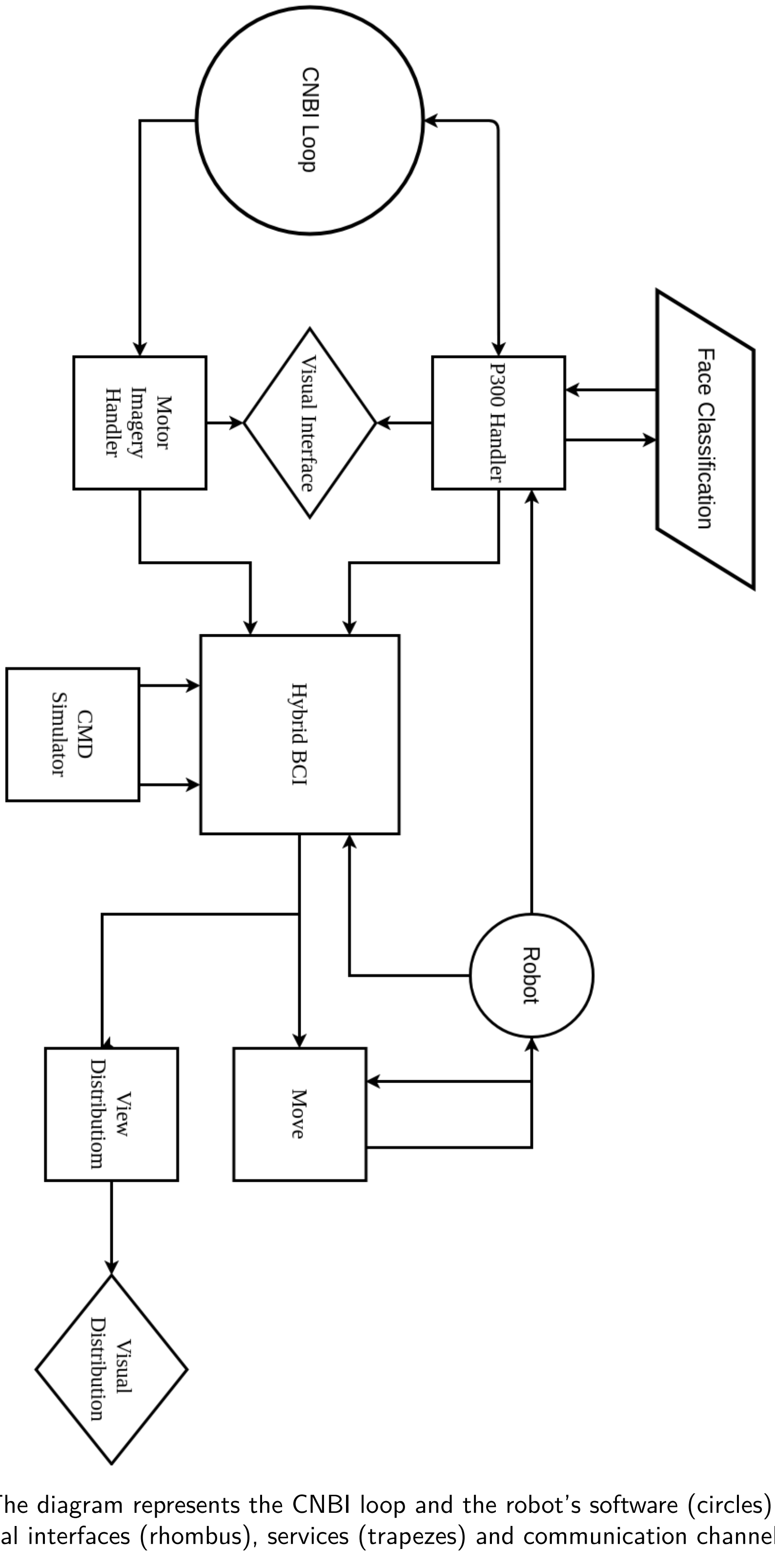

Figure 5.2: The diagram represents the CNBI loop and the robot's software (circles), nodes (squares), visual interfaces (rhombus), services (trapezes) and communication channels (arrows) of the system.

5.2.1 Hybrid BCI's System

This system receives the brain's signals directly from the BCI device as input and produce the directions distribution as output. This system receives information also from the robot, such as: its position and images from the robot's cam (Figure. 5.3).

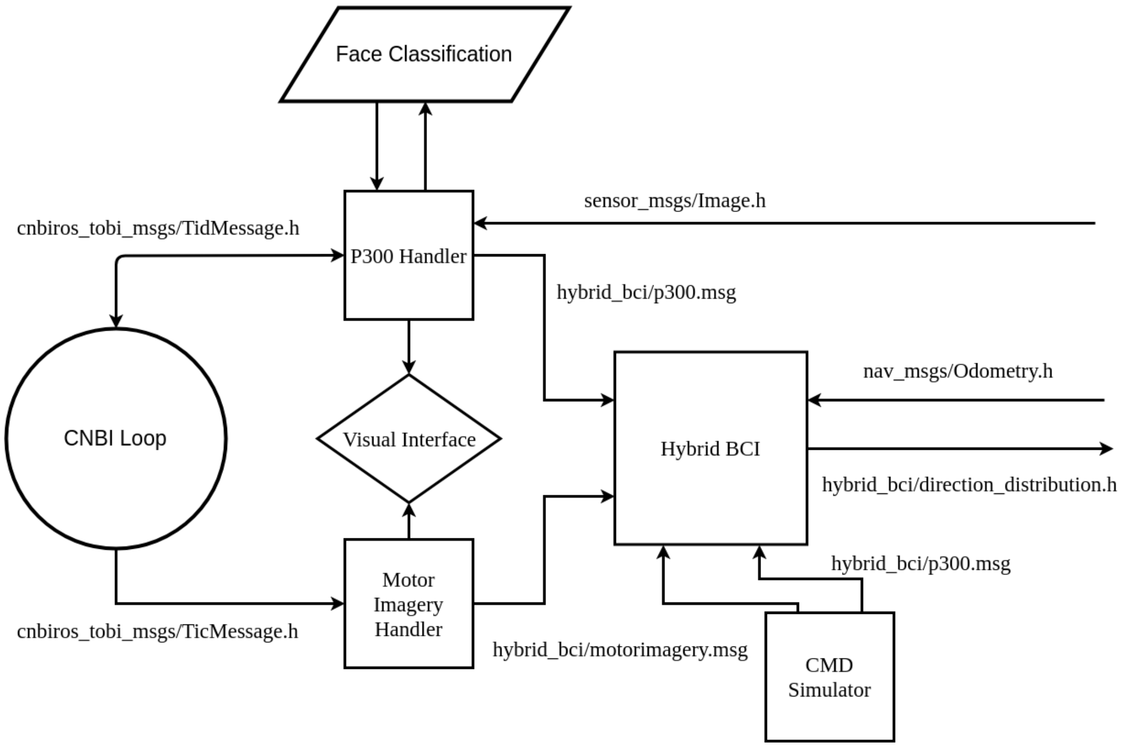

Figure 5.3: The Diagram represents the Hybrid Brain Computer System with the CNBI loop (circle), nodes (squares), visual interfaces (rhombus), services (trapezes) and communication channels (arrows) of the system with their type of messages.

CNBI Loop

The CNBI Loop is provided by the whitoolkit that receives, processes and classifies the brain's signals. We created two protocols allowing the CNBI Loop to communicate with ROS in the correct way. Both protocols are based on codes settled by the whitoolkit (Table. 5.1).

Event	Code (Hex)		Event	Code (Hex)
off	0x8000		attempt	0x0700
trial	0x0300		mi_hand_left	0x0301
go	0x0144		mi_hand_right	0x0302
nogo	0x0145		mi_both_feet	0x0303
cue	0x030f		mi_tongue	0x0304
wait	0x0001		mi_both_hands	0x0305
fixation	0x0312		mi_rest	0x0306
beep	0x0311		mi_smr	0x0307
cfeedback	0x030d		p300_task1	0x0201
dfeedback	0x030e		p300_task2	0x0202
targethit	0x0381		p300_task3	0x0203
targetmiss	0x0382		p300_task4	0x0204
eyeartifact	0x0400		p300_navigation	0x0205
flash	0x0500		stop_navigation	0x0206
prediction	0x0600		unknown	0x0207

Table 5.1: Table of event's codes of whitoolkit, wtk_events.xml .

The protocol for the motor imagery is simple, it configures the communication channels, selects the classifier previously trained and acquires the signals, at each instant the signals are classified and the protocol send the result to ROS through a $cnbiros\_tobi\_msgs/TicMessage$ message on the topic $/rostic\_cnbi2ros$ and it will be received and used by the Motor Imagery Handler. This message contains:

- Header header
- string pipe
- string version
- int32 frame
- TicClassifier[ ] classifiers

Each TicClassifier contais:

- string name
- string description
- int8 vtype
- int8 ltype
- TicClass[ ] classes: this field contains a label for the class selected and the probability of the correspondent class.

The protocol for the P300 is more complex. Initially the protocol selects a classifier and configures the communication channels. The `P300 Handler` makes a target flash sending to the protocol the event $p300\_taskN$, with N that corresponds to the target flashed, plus the event $flash$ through a $cnbiros\_tobi\_msgs/TicMessage$ message on the $/rostid\_ros2cnbi$ topic. The classifier collects the data and when the `P300 Handler` send the event $attempt$ plus the event off the protocol gets the class predicted. Once received this event the protocol sends to the `P300 Handler` the event $prediction$ plus the event off plus the event correspondent to the class predicted. If the class predicted has a probability greater than a threshold we add to the event accumulator the event $p300\_taskN$

while, if not, it adds the event $\boldsymbol{unknown}$. This communication happens through a $\boldsymbol{cnbiros\_tobi\_msgs/TicMessage}$ messages on the $\boldsymbol{/rostid\_cnbi2ros}$ topic for the messages from the `CNBI Loop` to the `P300 Handler`, while it uses the topic $\boldsymbol{/rostid\_ros2cnbi}$ for the messages from the `P300 Handler` to the `CNBI Loop`.

The TicMessage has the following structure:

- Header header
- string version
- int32 event
- int32 family
- string description
- string pipe

Motor Imagery Handler

The `Motor Imagery Handler` node receives $\boldsymbol{cnbiros\_tobi\_msgs/TicMessage}$ messages from the `CNBI Loop`, these messages represent the will of the user to turn left or right. The handler integrates the values passed in time. So we obtain the probability that the user wants turn left and the probability that the user wants turn right. If one of this probability overtakes a threshold the node sends a $\boldsymbol{hybrid\_bci/motorimagery}$ message to the Hybrid BCI node on the topic: $\boldsymbol{/motorimagery}$.

This message contains the following fields:

- int64 pkg_id : an integer that represent the index of the message.
- bool dir : a boolean that represent with true the will to turn right and with false the will to turn left.

This messages are sent to the `Hybrid BCI` node that converts them following the model. The node creates also its relative visual interface explained in the Visual Interface sub section.

P300 Handler

The `P300 Handler` node exchanges $cnbiros\_tobi\_msgs/TidMessage$ messages with the `CNBI Loop`, receives the images from the robot and uses the `Face Classification` service. Initially the handler initializes the `Face Classification` service calling $face\_classification/RegisterFace$, sending the faces of known people. The `Robot` sends the images on the topic $camera/rgb/image\_raw$ as a $sensor\_msgs/Image$ messages. The handler receives the images and shows it on the Visual Interface. After a fixed amount of time, repeatedly, an robot's cam image is classified by the `Face Classification` service calling $face\_classification/ClassifyAll$ and returns the position in the image of found faces. The faces found are our P300 target and each 300ms a random face is flashed. When a face flash the handler sends to the `CNBI Loop` the event $flash$ plus the event $P300\_taskN$, with N as the correspondent number of the target flashed. Before that a new image is classified, the handler send the event $attempt$ plus the event off so the `CNBI Loop` responds with the event correspondent to the result of the prediction. If the predicted target is known the handler estimate the offset degree between the target and the center of the robot's field of view and send a $hybrid\_bci/P300$ message to the `Hybrid BCI` node on the topic: $/p300$. The P300 message has the following fields:

- int64 pkg_id: an integer that represent the index of the message.
- int64 tot_people: an integer that represent how many people are present in the scene.
- P300_person[] person: an array represent the people sent.

The type P300_person has the following field:

- int64 id: integer that represent the people.
- float64 p: probability that is the correct target.
- int64 dir: offset in degree between the target and the center of the robot's field of view.

Face Classification

The Face Classification service is provided by the Intelligent Autonomous Systems Laboratory. This ROS service allows to register faces in a classifier to detect them in other images.

Visual Interface

The Visual Interface is composed by two parts. The first one is relative to the `Motor Imagery Handler` (Figure. 5.4). In this interface we have one bar that represent the probability that the user wants to turn left or wants to turn right. The red line divide the bar in two, the green part moves left and right representing how change the probability that the user wants to turn.

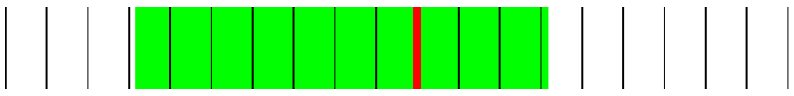

Figure 5.4: Motor imagery bar for BCI feedback.

The second part of the Visual Interface is related to the `P300 Handler` (Figure. 5.5). The handler, at intervals of one second, calls the `Face Classification`, passing the current image passed by the robot's cam. The handler asks for the classify all service, that returns the position in the image of the known faces. The handler makes an empty black rectangle on the known faces. When a face is flashed the empty rectangle became filled in green.

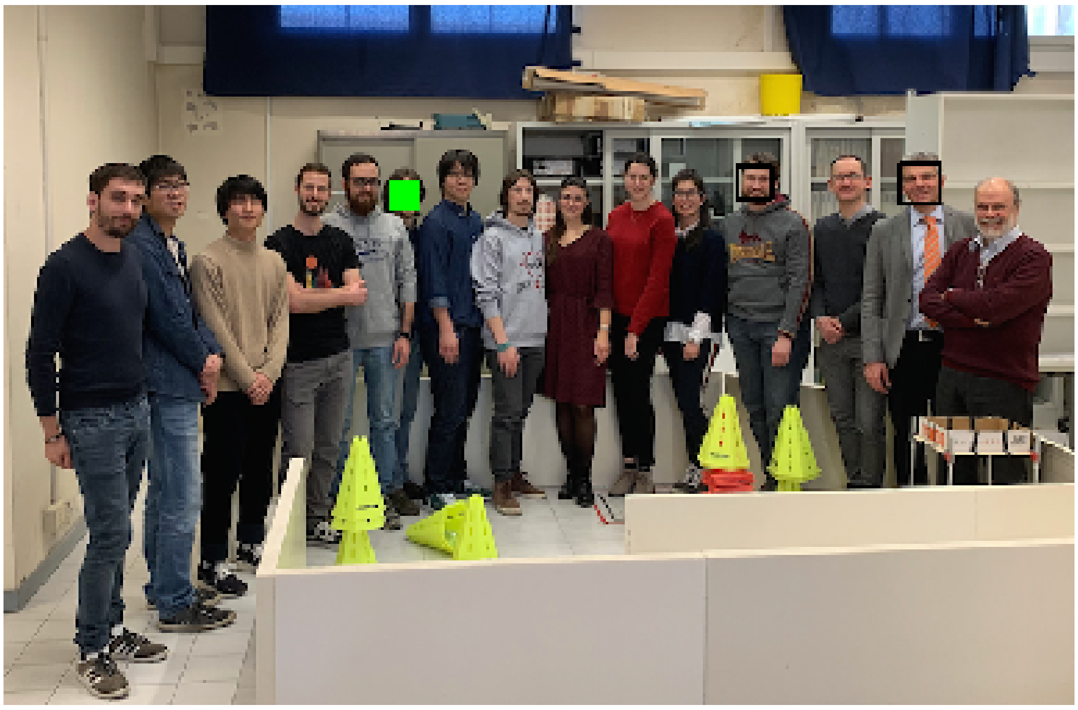

Figure 5.5: P300 target with empty black rectangles on their faces while the green rectangle is on the P300 target flashed in that moments.

When the `CNBI Loop` sends the predicted target, the black rectangle of this target begin filled in red for two seconds.

CMD Simulator

The `CMD Simulator` node allows to simulate P300 and motor imagery commands, sending respectively $hybrid\_bci/P300$ and $hybrid\_bci/motorimagery$ messages on the topics $/P300$ and $/motorimagery$. The possible commands are in Table. 5.2 .

Key of the keyboard	Action
p	Play
q	Quit
a	Send Motor imagery left message
d	Motor imagery right message
s	Send p300 message
1, 2, 3, 4	Set the p300 target

Table 5.2: Commands of the simulator.

The Play action is the only one that the user have to press to make the system start. The Quit action close the robot's system nodes. When the user press a or d sends a $hybrid\_bci/motorimagery$ message that is handled by the Hybrid BCI node. The commands for the P300 are more complex. Pressing a number from 1 to 4 the probability of the correspondent P300 target are set to: 0.7 while, for the others, it is set to 0.1. The degree's offset of this simulated targets are:

- -90° degree for the target 1.
- -45° degree for the target 2.
- 45° degree for the target 3.
- 90° degree for the target 4.

When the user press s, she/he sends a $hybrid\_bci/P300$ message with the parameters described before.

Hybrid BCI

The `Hybrid BCI` contains the model presented in Chapter 4. This node receives the commands on the topics $/P300$ and $/motorimagery$, translate the messages in probability distribution following the model and calculate the resulting directions distribution. To calculate the probability distribution over the directions receives, in the topic $/odom$, $nav\_msgs/Odometry$ messages that are useful to transport the offset passed by the P300 in absolute directions of the system. At each instant the node sends a message of type $hybrid\_bci/direction\_distribution$ on the $move$ topic. This message represent the current directions distribution and it has the following fields:

- int64 best_dir: representing the mode of the distribution.
- float64[ ] directions: the full directions distribution.

The no-command Gaussian is generated and used in the model when the target direction, that is the mode of the directions distribution, remains the same for a fixed amount of time.

5.2.2 Robot's System

This part of the system takes care of the robots actions. As input of this part we have the directions distribution and as output we have the actions of the robot. As you can see from the previous part, the BCI System and the Robot's system are joined by tree communication channels, such as the $nav\_msgs/Odometry$, the $sensor\_msgs/Image$ and the $hybrid\_bci/direction\_distribution$ messages. While the first two messages are sent from the Robot's System to the BCI's System and they are ordinary messages, the last one is the only one that comes from the BCI's System to the Robot's and it is created from us. This system is composed by three nodes, the `Move` node, the `Robot` software and the `View Distribution` node with its visual interface (Figure. 5.6).

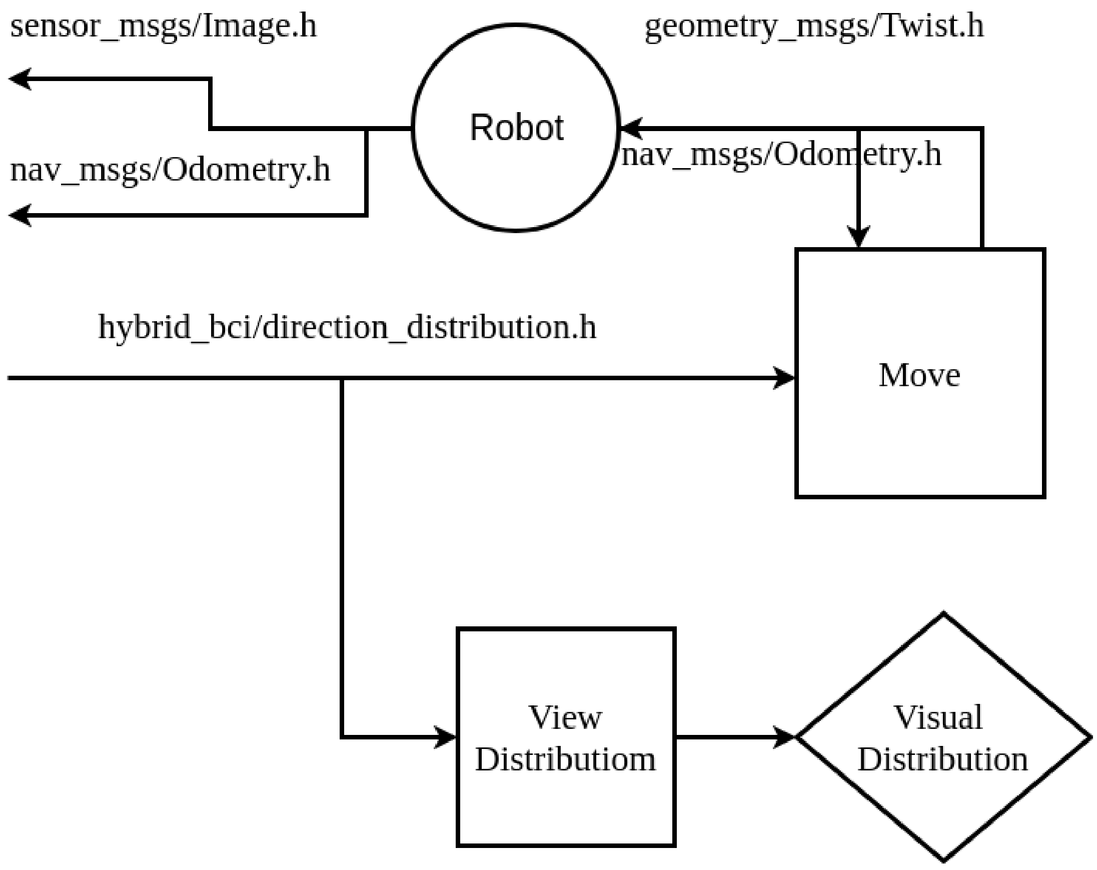

Figure 5.6: The Diagram represents the Robot's System with nodes (squares), visual interfaces (rhombus), services (trapezes) and communication channels (arrows) of the system with their type of messages.

Move

The `Move` node is a translator that converts the $hybrid\_bci/direction\_distribution$ in messages that the robot can actually use, in this case we convert them in $geaometry\_msgs/Twist$ messages. This node, once received the target direction, calculates if it has to go straight or turn. Because of the little precision of the Odometry, that receives from the robot with $nav\_msgs/Odometry$ messages, the robot verifies if the target directions is in a neighborhood of its current direction. If it is not the robot has to turn, right or left, in the direction of the target direction.

View Distribution

The View Distribution node is an optional node, it is not necessary for the functioning of the system but it is useful to see how change the visual distribution when the user sends commands. The image created is with absolute coordinates, so when the system is in function we can see the probability of all the directions (Figure. 5.7). This node receives the information about the distribution through *hybrid_bci/direction_distribution* messages on the topic *move*.

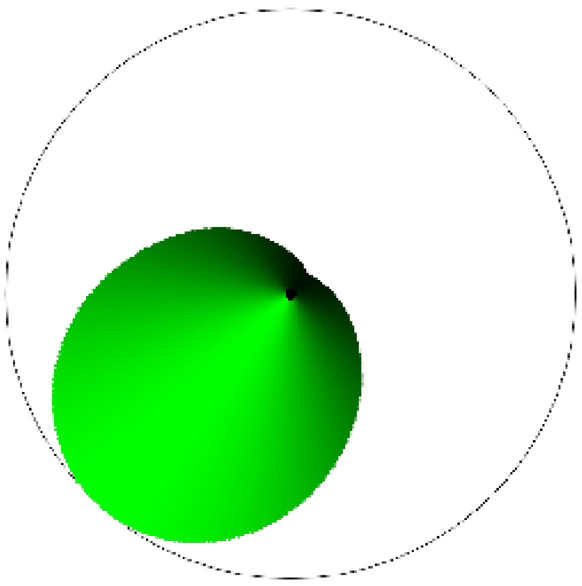

Figure 5.7: Direction distribution in the visual interface of the view distribution node.

Robot

The `robot` software is usually provided from the ROS community and it can send a lot of messages. For this thesis we used a turtlebot simulator provided by ROS. The modularity of ROS allows us to describe which messages this node has to provide and to receive so we are not constrained to use a single robot but the system can be re used more times, with different robots. The robot has to provides:

- *sensor_msgs/Image* messages: that are the images registered in front of the robot, used for the face detection of the P300 targets.
- *nav_msgs/Odometry* messages: that tell us the position of the robot in the space.

The robot has to receive:

- *geometry_msgs/Twist* messages: that are commands velocity commands send to the robot.

6 Experiments

In this Chapter we want present the test environments with the relative results. In the first Section you will find the result found with the simulated system presented in Chapter 4.

The second Section presents the results obtained in a simulation on Gazebo[*], Gazeebo provides an editable virtual environment where we insert a virtual robot and some photos to simulate the targets' faces.

All the data and the code of the simulations will be online on Github [†]. The BCI Simulator folder contains codes and results for the simulation, while the others directories contains code to execute the system in a real environment or in the Gazebo's virtual environment. We wants to remember that to compile and to execute the code you need to install all the libraries presented in Chapter 3, Section 3.

6.1 Simulator Experiments

The experiments made on the simulator are important to estimate the system's parameters. So we have an ideal environment to make tests unless to find all

[*]http://gazebosim.org
[†]https://github.com/michelebenini/HybridBCI-Test

the real issue. As first we want show how the distribution probability over the direction changes with commands.

6.1.1 No-commands simulation

In this Section you can see how the probability distribution over the direction changes when the user does not send commands. The first image represent the start situation (Figure. 6.1), no commands are sent.

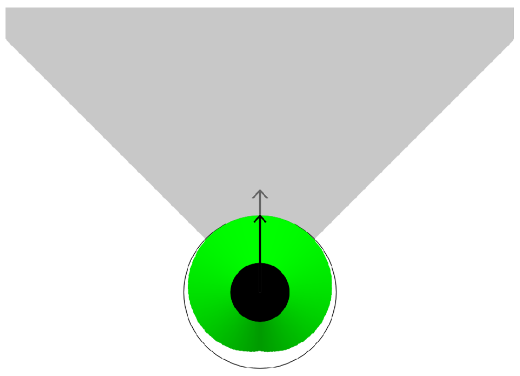

Figure 6.1: Start situation.

The distribution in Figure. 6.2 is obtained when a single no-commands is added to the system.

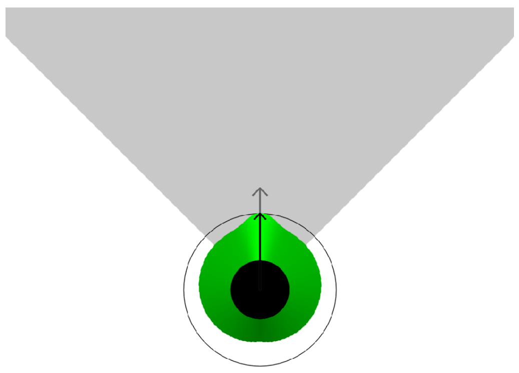

Figure 6.2: No-command sent.

We sent a new no-command (Figure. 6.3), as we can see the Gaussian are becoming more steep.

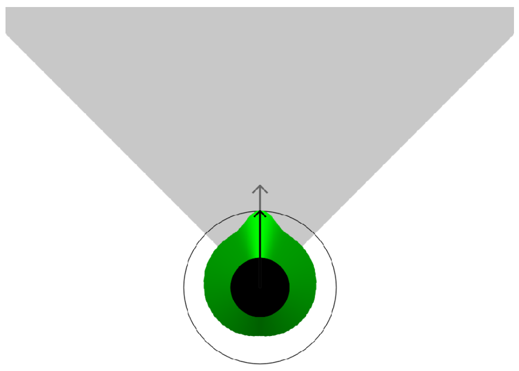

Figure 6.3: No-command sent.

We sent other tree no-commands (Figure. 6.4). So we have start situation and 5 no commands.

Recalling the formula, we had:

$$D_{t|t_0:t} = \mu_1 \cdot \mathcal{N}(x_{t-1}, \sigma_{t-1}) + \mu_2 \cdot D_{t-2|t_0:t-3} \tag{6.1}$$

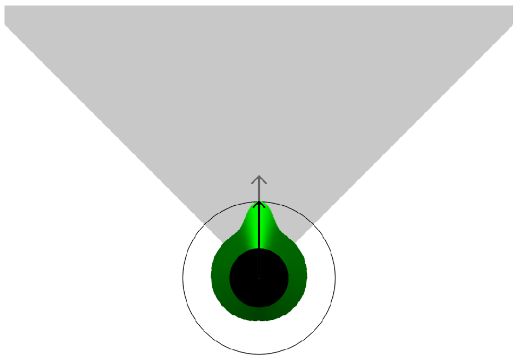

Figure 6.4: A new tree no-commands is added.

The next image (Figure. 6.5) show the no-commands saturation, when t tends to infinity $\mu 1$ is greater than $\mu 2$. This situation is obtained when the user does not send commands for a lot of time. The no-commands Gaussians generated becomes stronger each new instant because at time t, the past, D_{t-2}, is also composed by no-commands Gaussian. This makes the distribution before the no-commands trifling.

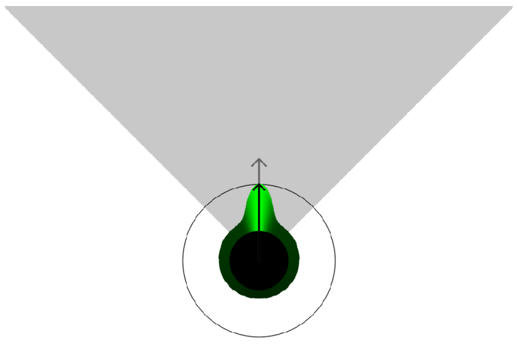

Figure 6.5: In this distribution we sent many no commands to bring the system to no-commands saturation.

The Figure. 6.6 show all the distribution together. We plot the probability over the direction. The current direction of the simulated bot is at the center of the image. In the initial condition we have the lowest Gaussian, at each no-commands the distribution had a higher probability on the mode of the distribution.

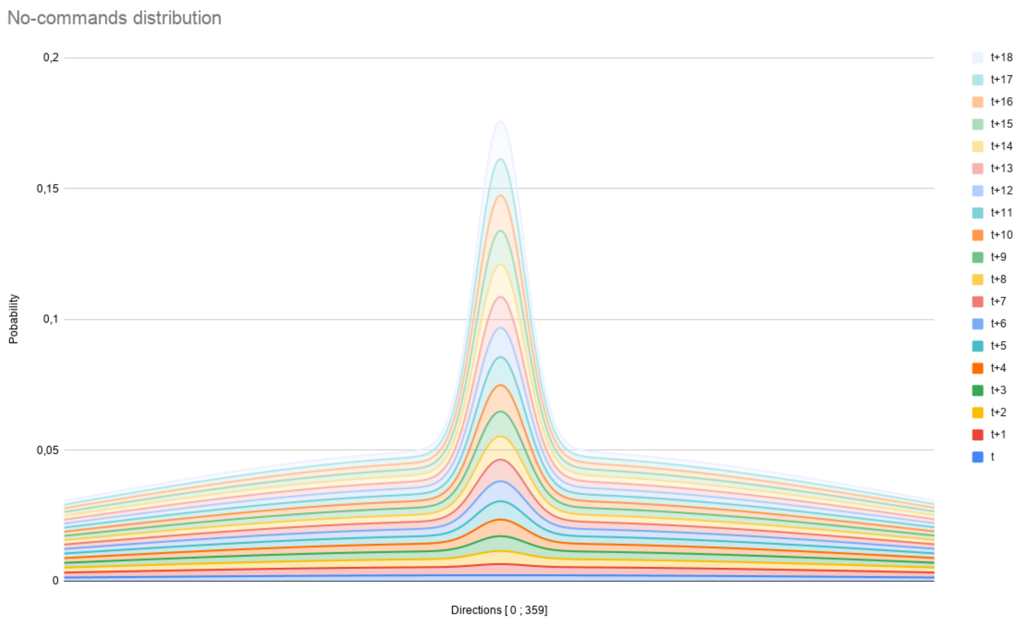

Figure 6.6: The current direction correspond to 180° degree. The Blue area at the bottom represents the start situation while each curve above represents a no-command added to the distribution below it.

The parameters used for this simulation are in Table. 6.1.

Initial command						
x_t	σ_t	μ_0	x_{t-1}	σ_{t-1}	μ_1	μ_2
180	180	1	0	0	0	0

No-Commands						
x_t	σ_t	μ_0	x_{t-1}	σ_{t-1}	μ_1	μ_2
0	0	0	180	10	0.03	0.97

Table 6.1: Table of no commands simulation data.

6.1.2 Motor Imagery simulation

In this Section you can see how the probability distribution over the direction changes when the user sends only motor imagery commands in sequence, unless generating no-commands. The degrees offset of the generated Gaussians is set to 45° degrees.

The start situation is in Figure. 6.7.

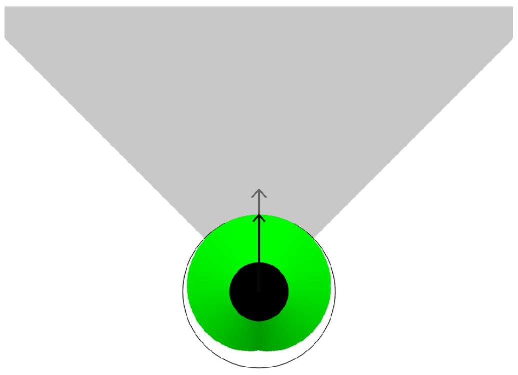

Figure 6.7: This is the start situation, no commands are sent.

We sent a single right motor imagery command (Figure. 6.8).

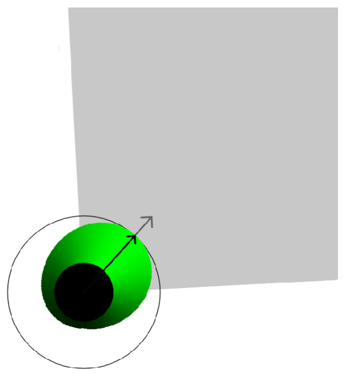

Figure 6.8: Motor imagery right command is sent.

We sent another motor imagery right command (Figure. 6.9).

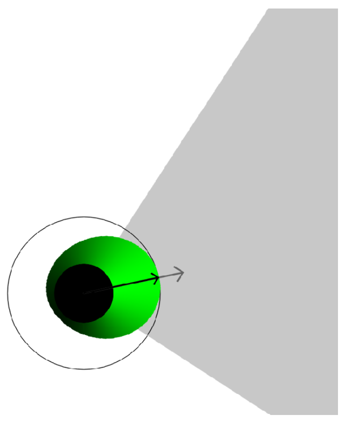

Figure 6.9: A new right motor imagery command is added.

We sent a motor imagery left command (Figure. 6.10). So we have start situation and 2 motor imagery right commands and 1 motor imagery left command.

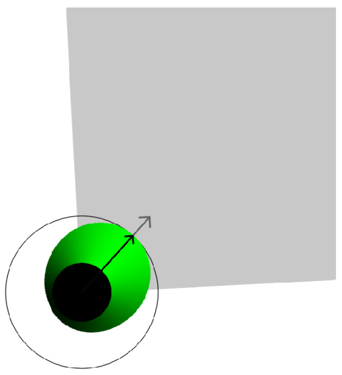
Figure 6.10: In this distribution a new left motor imagery command is added.

We sent another motor imagery left command (Figure. 6.11), because we sent two motor imagery commands in both the direction we can suppose that the robot turn in the position described in Figure. 6.7 but the contribution of the Gaussian's tails make the mode of the distribution shift, even if only a little. In the start situation the mode is at 180° degree while after the four commands is at 187°.

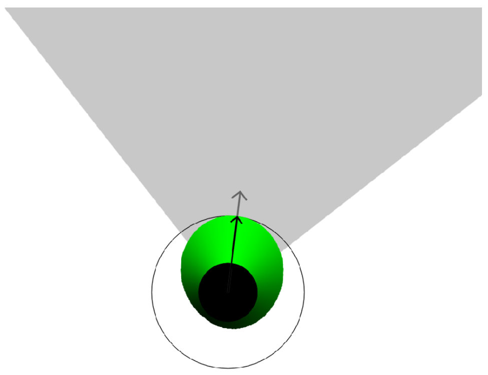
Figure 6.11: In this distribution a new left motor imagery command is added.

In the following image (Figure. 6.12) we can see the distributions presented in the past four images in one chart. As we can see the pics of the Gaussians move according with the commands.

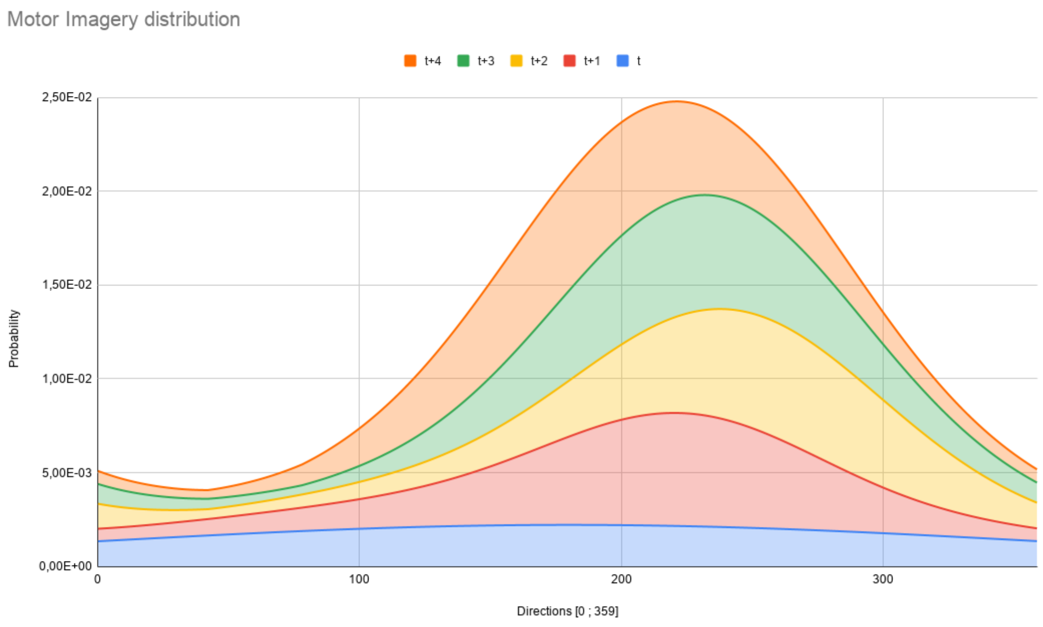

Figure 6.12: Probability distribution over the directions relative to the past four commands. Each color represent the distribution in the past four Figures. In Blue we have the distribution in the start situation, in Red after the first command and so on.

In the Figure. 6.12 the mode of the distribution is initially 180° degree, than it was shifted to 222° degree with the first right command, 258° degree with the second and with the left commands turn to 222° degree and 187° degree.

In the following table you can find the parameters used for the past test. We remember that the centers of the generated Gaussians x_t are relative to the previous distribution's mode, x_{t-1} plus an offset, off.

Initial command						
x_t	σ_t	μ_0	x_{t-1}	σ_{t-1}	μ_1	μ_2
180	180	1	0	0	0	0

Motor Imagery						
x_t	σ_t	μ_0	x_{t-1}	σ_{t-1}	μ_1	μ_2
x_t	50	0.5	x_{t-1}	100	0.3	0.2
Degree's offset in motor imagery commands off				$\pm 45°$		

Table 6.2: Table of motor imagery simulation data.

6.1.3 Motor Imagery & no-commands simulation

In this Section you can see how the probability distribution over the direction changes when the user sends motor imagery commands with pauses between them, so some no-commands distributions are generated.

The initial condition of the system is presented in Figure. 6.13.

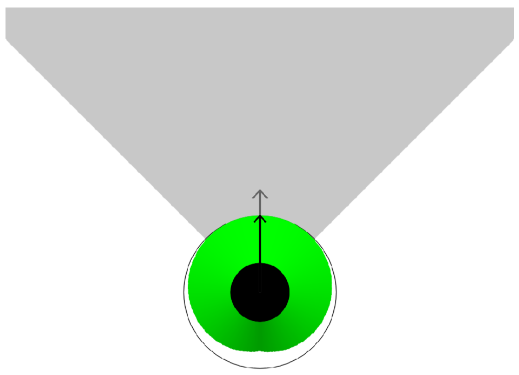

Figure 6.13: This image represents the initial condition of the system.

We did not send commands for eight seconds, making it, the system generated eight no-commands (Figure. 6.14).

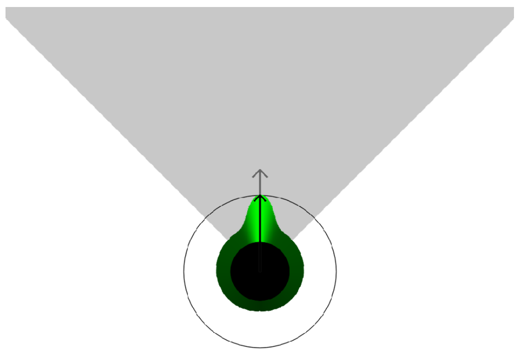

Figure 6.14: Eight no-commands are sent.

After the eight seconds we sent a motor imagery right command (Figure. 6.15).

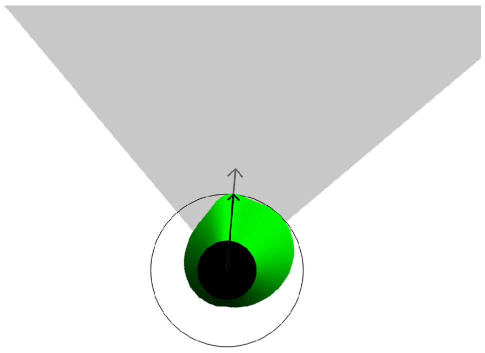

Figure 6.15: We sent a right motor imagery command at the system.

Now we added a new right motor imagery command at the system (Figure. 6.16).

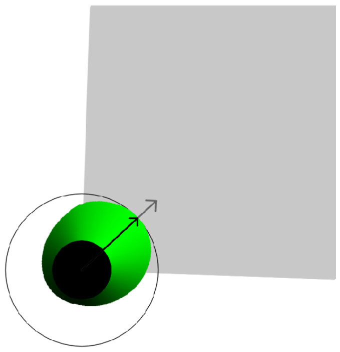

Figure 6.16: We sent right motor imagery command.

We did not send commands for ten new seconds, after all this commands the system has the shape in Figure. 6.17.

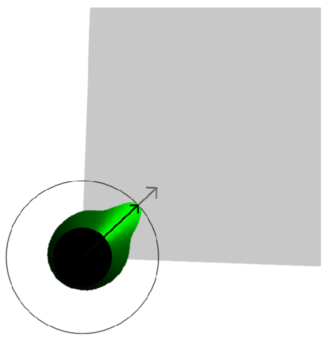

Figure 6.17: We sent ten no-commands at the system.

In this situation the robot when receives the commands change its direction from 180° degree to 185° when it receives the first motor imagery commands and to 225° after the second one.

Supposing that the user does not send the first right command. So after the Figure. 6.14 we continue with another situation.

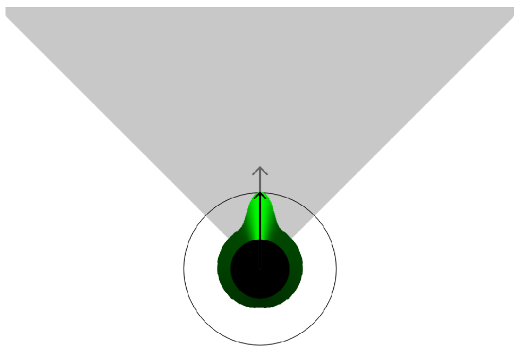

Figure 6.18: Nine no-commands.

In the last distribution (Figure. 6.18) we sent another no-command at the system starting from 6.14. So, the total system is consequence of the initial distribution and nine no-commands.

Then we send a motor imagery right command (Figure. 6.19).

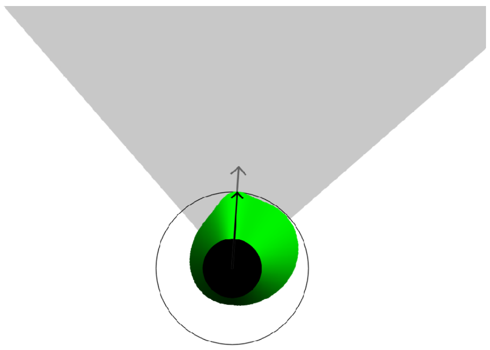

Figure 6.19: We sent a motor imagery right command.

After nine no-commands and the motor imagery right command we sent five new no-commands(Figure. 6.20).

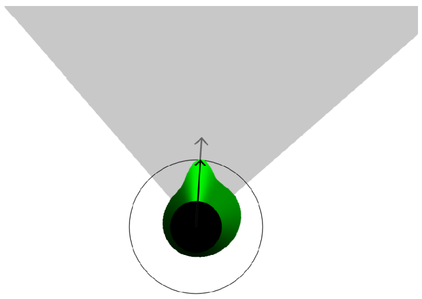

Figure 6.20: Added five no-commands.

In this situation the motor imagery commands are not enough strong to change the mode of the distribution. The idea behind the choose of the parameters that allows this is that, sometime the user can send motor imagery commands accidentally. In this situation the user has to send more than one commands to certificate its idea of change the robot's direction.

In the chart in Figure. 6.21 we want to show the distribution in the first case, when the user changes the mode of the distribution. In the chart in Figure. 6.22 we want to show the distribution in the second case, when the user does not change the mode of the distribution. In both the case we want to show how the distribution change with the commands.

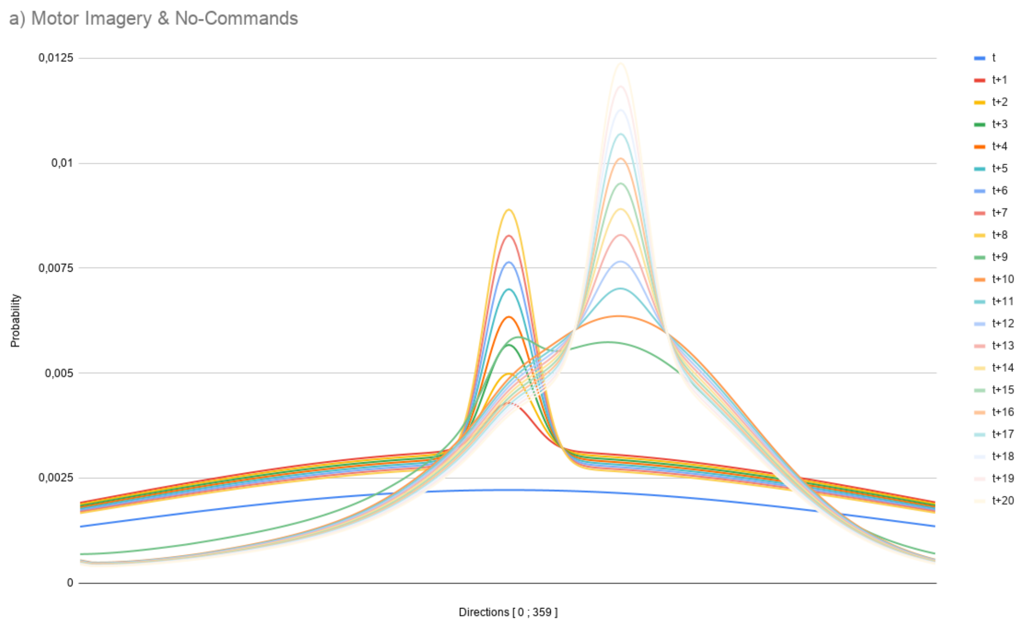

Figure 6.21: This chart represents the distribution when from the initial situation the user sends eight no-commands, two motor imagery right commands and other ten no-commands. The initial condition is represented by the blue curve, when the system receives the no-commands the probability in the center of the image increases. The green curve represents the distribution when the system receives the first motor imagery commands and the orange, the second one. After this the mode of the distribution is shifted and when the system receives the last ten no-commands the probability over the mode increase.

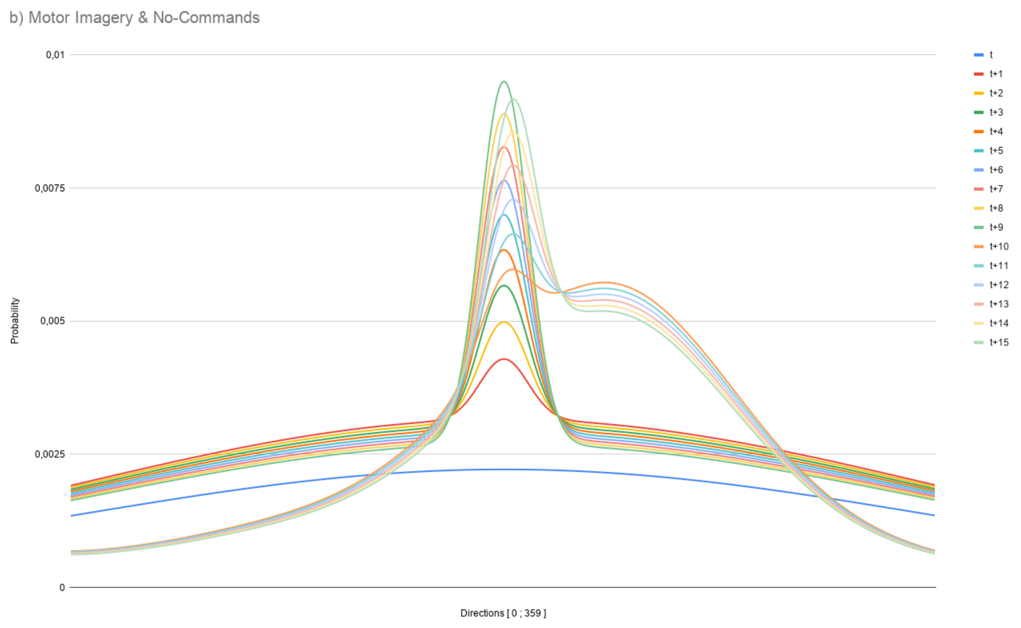

Figure 6.22: This chart represents the distribution when from the initial situation the user send nine no-commands, one motor imagery right commands and other five no-commands. The initial condition is represented by the blue curve, when the system receives the no-commands the probability in the center of the image increase. The orange curve represents the distribution when the system receive the first motor imagery commands. After this the mode of the distribution is the same as before, so, when the system receives the last five no-commands the probability over the mode increase while over the other direction tends to be the same.

The parameters used in this case are the same used before and presented in the Table. 6.1 and Table. 6.2.

6.1.4 P300 simulation

In this Section you can see how the probability distribution over the direction changes when the user sends only P300 commands in sequence, unless generating no-commands or sending motor imagery commands.

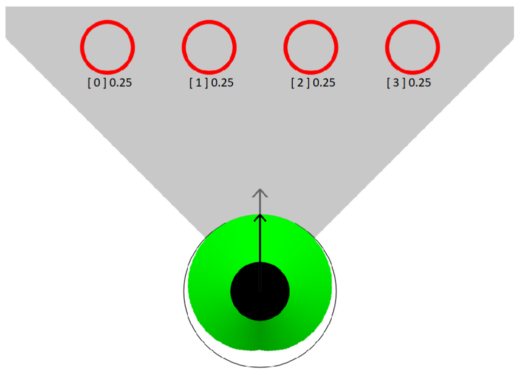

Figure 6.23: Initial system's probability distribution.

From the initial distribution (Figure. 6.23), we sent a P300 command on the last target on the right, the target 3 (Figure. 6.24). This target has probability set to 0.77 while the other targets have probability 0.11.

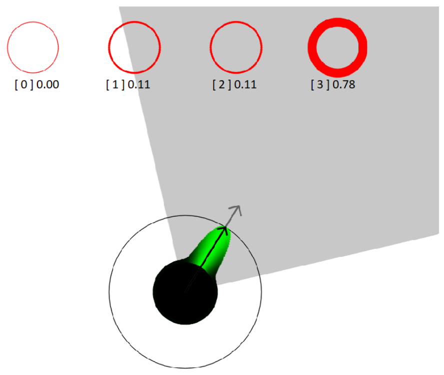

Figure 6.24: P300 command sent on target 3.

After, we sent a new P300 command on the target 2 (Figure. 6.25), with the same probability of before. The target has probability 0.77 and the others have probability 0.11.

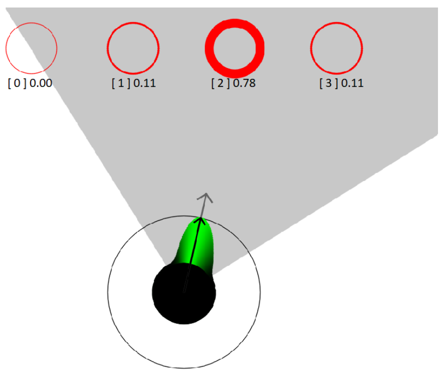

Figure 6.25: P300 command sent on target 2.

We sent another P300 command on the target 1 (Figure. 6.26), as you can see, the probability distribution are losing its Gaussian's shape, this is because for this test μ_1 and μ_2 is not equals to zero.

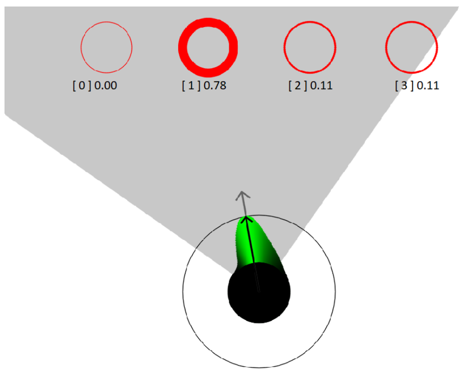

Figure 6.26: P300 command sent on target 1.

To see the difference of using μ_1 equals or not to zero you have to see the next chart. Both the distribution are derived by sending a P300 command on target 3 followed by one on target 1 (Figure. 6.27). To generate this chart we start from the initial condition (Figure. 6.23).

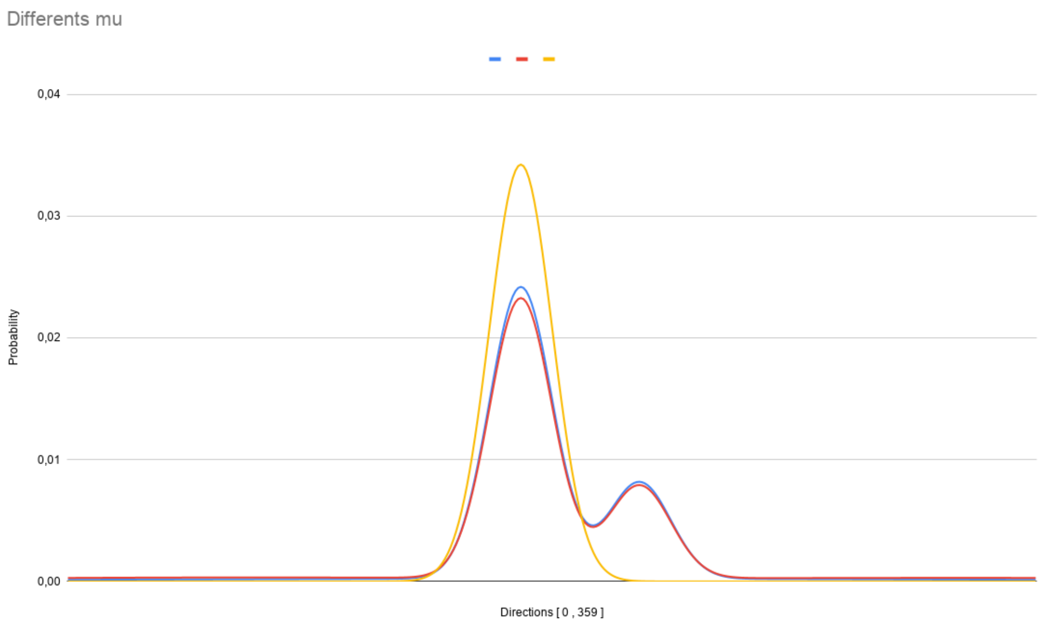

Figure 6.27: Distribution generated by changing μ. The yellow distribution is generated with μ_1 and μ_2 equal to zero, the blue one with μ_1 equals to 0.03 and μ_2 equals to zero. While the red one is generated with μ_1 equals to 0.03 and μ_2 equals to 0.3.

The next char wants to show how the generated distribution change with the targets' probability change (Figure. 6.28).

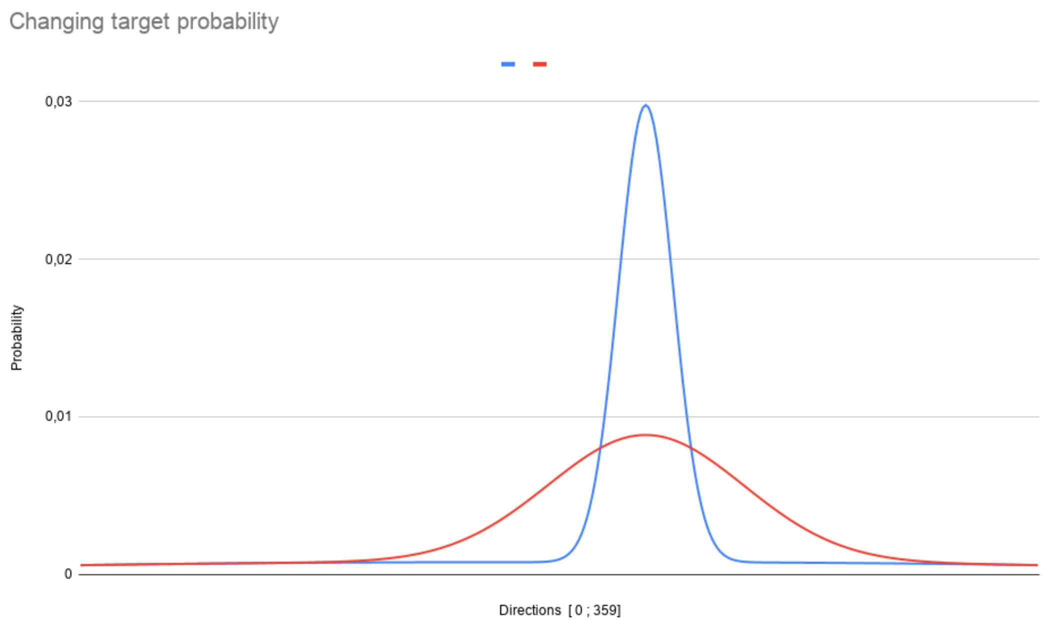

Figure 6.28: Distribution generated by changing the probability of the P300 targets. The blue distribution is generated sending a P300 with probability 0.98 on the target 3 and 0.02 on the target 0. While the red one is generated sending a P300 with probability 0.52 on the target 3 and 0.48 on the target 0.

As you can see, there are not indication in the chart of where is the target 0 because this distribution generation model use the probability only to change the standard deviation of the Gaussian corresponding to the best target.

6.1.5 All commands

In this Section you can see how the probability distribution over the direction changes when the user sends P300 and motor imagery commands, generating also no-commands.

From the initial condition (Figure. 6.29) we want show the distributions produced using all types of commands.

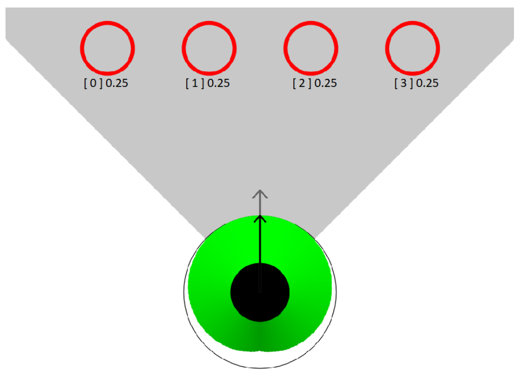

Figure 6.29: P300 command sent on target 1.

In the Figure. 6.30 we can seen the distribution generated after tree no-commands and a P300 command on the target 3.

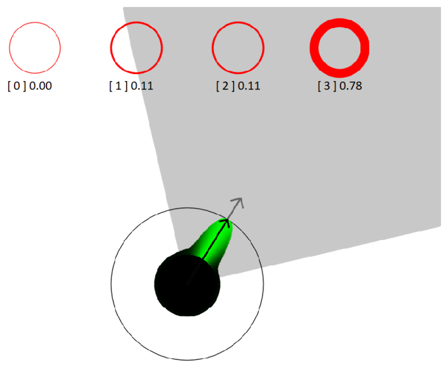

Figure 6.30: Tree no-commands and one P300 command.

If we send a P300 command on the target 3 and then a left motor imagery command we obtain the distribution in Figure. 6.31.

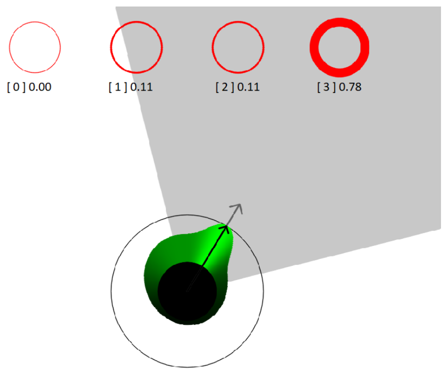

Figure 6.31: P300 command and motor imagery left.

While if we send a P300 command on the target 3, a left motor imagery commands and wait five seconds to send another motor imagery left command we obtain the distribution in Figure. 6.32.

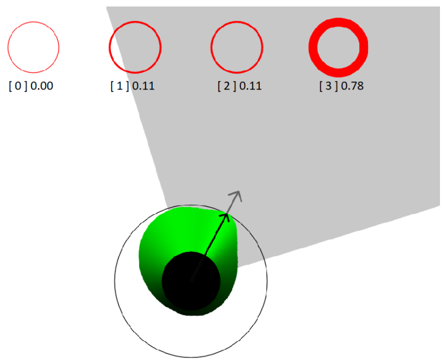

Figure 6.32: P300 command and motor imagery left with no commands.

6.2 Gazebo Experiments

The following result are related to experiments with a fully simulated environment. The BCI commands are simulated by the node ***CMDSimulator*** or they were simulated sending ROS messages from the command line with ***rostopic pub /topic***. In all the experiments the robot moves at the same velocity (0.1 m/s).

The user sees the elements in the following Figure (Figure. 6.33). As explained in Chapter 4 we have:

- P300 Visual interface: the upper part shows which the robot sees, in the Figure below we can see two P300 target, where one is selected. The red rectangle is not centered on the face of the subject because the face detector is not fast as the robot.
- Motor imagery interface: the central part represents the motor imagery bar, this bar has to show the probability that the user wants to turn. In this picture the bar is empty because we simulated the commands so we did not need the bar.
- Direction distribution interface: the lower part shows the direction distribution of the robot in absolute degrees. This interface, also if it is not mandatory, it is very useful to learn how use the system.

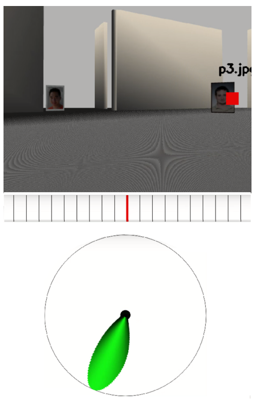

Figure 6.33: All components of visual interface.

The Figure 6.34 represents the simulated test environment. The user had a fixed path to travel, in yellow, from the start position to the goal position. In the environment we can see also the positions of the P300 targets.

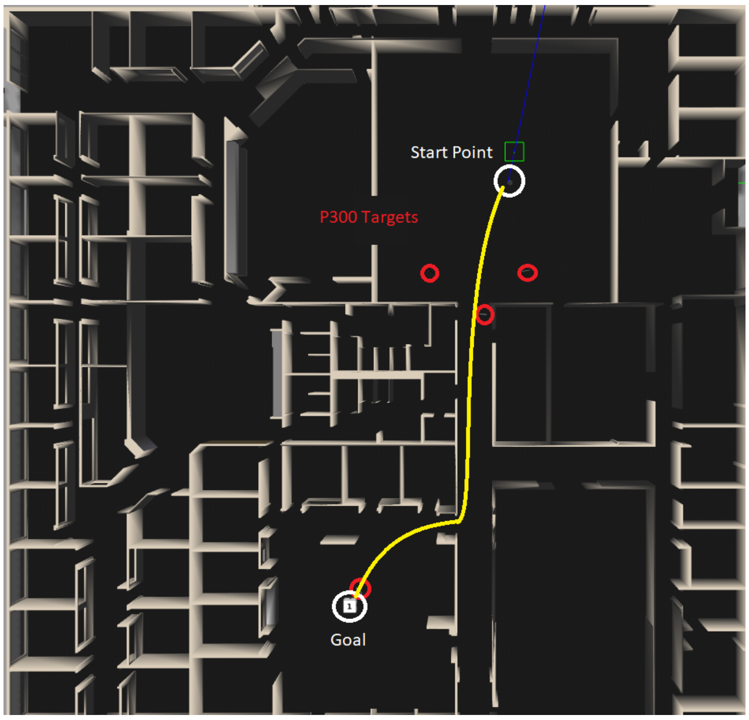

Figure 6.34: Simulated test environment. The red circles indicate the P300 targets, the white circles indicate the start position and the goal position. The yellow line indicates the estimated path that the robot have to do.

All the following results were obtained from the same user that, sequentially, tries ten times to use the system without P300 commands and after tries ten times to use the full system. So, in the first section on trials the user learned how the systems work to work better in the second session.

6.2.1 Motor imagery Only

In this Section we want to show the result obtained driving a robot on a fixed path with the system presented in Chapter 5 using only motor imagery commands. The motor imagery commands are sent at least two second from each other. In the following table you can find the results obtained, we kept trace of the number of commands used, the number of command used per type and the time that the robot takes to go from the start to the end.

Test number	Motor Imagery	No-Commands	Total	Time
1	21	398	419	6m 56s
2	22	477	450	7m 40s
3	24	413	437	7m 3s
4	27	471	498	8m 5s
5	19	406	425	7m 6s
6	20	408	408	7m 01s

Table 6.3: Table of motor imagery results pt.1

Test number	Motor Imagery	No-Commands	Total	Time
7	19	408	427	6m 59s
8	19	419	438	7m 16s
9	17	394	411	6m 44s
10	24	406	430	6m 56s
Average	21.2	422.4	443.6	6m 32s

Table 6.4: Table of motor imagery results pt.2

In the following figure (Figure. 6.35) you can see an example of path that the robot follows with the conditions described before.

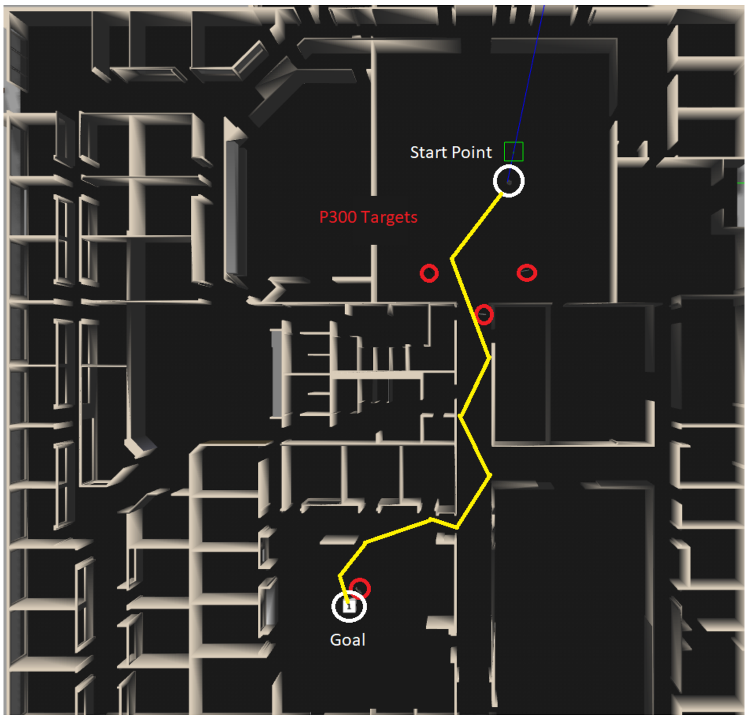

Figure 6.35: Simulated test environment. Motor imagery example path.

6.2.2 All commands

In this Section we want to show the result obtained driving a robot on a fixed path with the system presented in Chapter 5 using motor imagery and P300 commands. The probability of P300 target is In the following table you can find the results obtained, we kept trace of the number of commands used, the number of command used per type and the time that the robot takes to go from the start to the end.

Test number	Motor Imagery	P300	No-Commands	Total	Time
1	20	1	426	447	7m 7s
2	14	2	362	378	6m 9s
3	14	2	369	385	6m 10s
4	9	2	364	375	6m 5s
5	16	2	383	401	6m 25s
6	16	3	394	413	6m 35s
7	13	2	373	388	6m 10s
8	16	2	378	396	6m 19s
9	11	2	343	356	5m 43s
10	14	2	370	386	6m 10s
Average	14.3	2	376.2	392.5	6m 17s

Table 6.5: Table of all commands results

As you can see comparing the Tables 6.3 and 6.4 and the Table 6.5 usually with the P300 commands we can pilot the robot with less intentional commands. In the Table 6.5 you can see that we sent a different number of P300 commands, this is because sometimes we had not the need because we was in the trajectory of the target with only motor imagery commands or sometimes we sent one more because we tried to go near the goal using a different path.

In the following Figure (Figure. 6.35) you can see an example of path that the robot follows with the conditions described before.

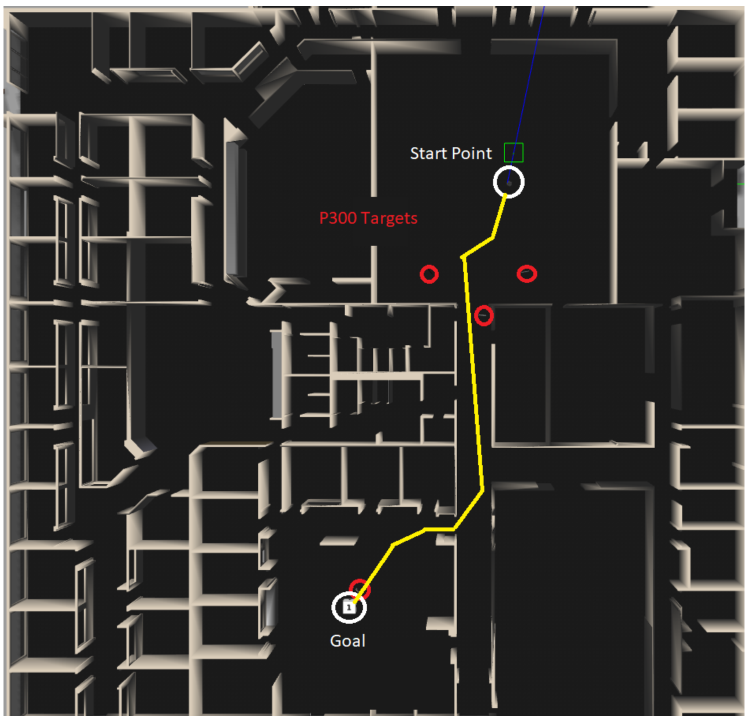

Figure 6.36: Simulated test environment. All commands example path.

6.2.3 Keyboard teleop

In this Section we present the results obtained using commands sent in a normal way with the keyboard to see the difference between our system and a common driver system. The only difference is that in these part we make only five test because this driver method tends to have no substantial variations. The node we used is provided from ROS in the package $turtlebot\_teleop$ and it is `turtlebot_teleop_key`, this node allows to send messages of type $/cmd\_vel$. We used a launch file to send the messages in a specific topic.

This node accepts commands from the keyboard, the user has to keep the key pressed to move the robot so, to compare this commands with the previous, we consider the hold down the key as no intentional commands. When we press another key we see this action as intentional commands. This is because We can consider as intentional commands the motor imagery commands and the P300 commands while the no-commands as no intentional commands auto generated by the model.

Test number	Intentional	No Intentional	Total	Time
1	20	378	398	6m 18s
2	22	365	387	6m 5s
3	24	348	372	5m 48s
4	16	360	376	6m 0s
5	24	370	394	6m 10s
Average	21.2	364.2	385.4	6m 4s

Table 6.6: Table of Keyboard teleop results

An example of the robot's path was presented in Figure. 6.34.

To compare the results obtained in the previous Sections we plot the average results for intentional commands, motor imagery and P300 commands, no intentional commands, no-commands auto generated by the system, the total of commands and the time (Figure. 6.37).

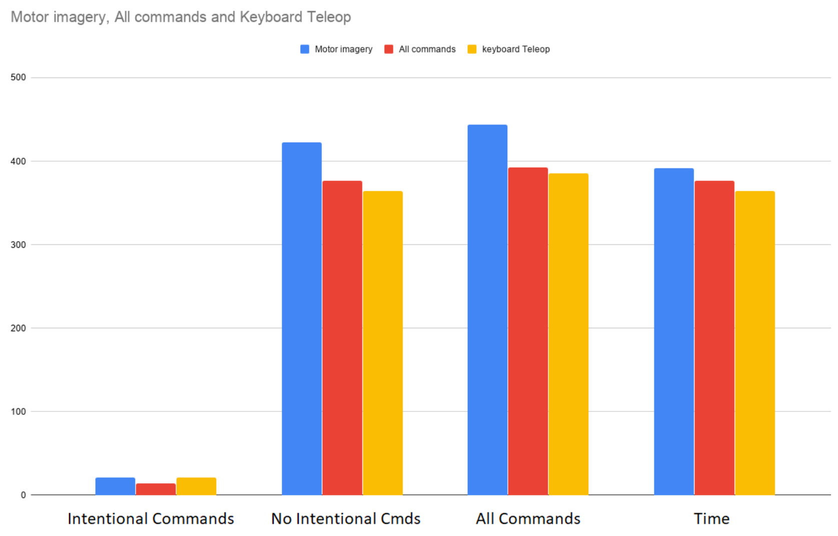

Figure 6.37: Graph to compare results averages.

As we can see from the graph with the full hybrid system we use less intentional commands, this is an important goal to reach. To drive the robot with a BCI is difficult, so, using less commands as possible makes the system more functional. The only motor imagery system is difficult to control and, usually, we had to send commands to try to drive the robot straight ahead. Using the keyboard we obtain better time and better path than using our model, this is because we have more control of the robot.

To have a better comparison of the results we use the following Formulas to calculate two comparison ratios (cmp_{time} and cmp_{cmd}) over the mean of times and intentional commands used in the experiments for the system with only motor imagery commands and with the full system:

$$cmp_{time} = \frac{BCI_{time}}{Keyboard_{time}} \tag{6.2}$$

$$cmp_{cmd} = \frac{BCI_{I.Cmd}}{Keyboard_{I.Cmd}} \tag{6.3}$$

Applying this formulas to our results we obtain for the motor imagery system:

- $cmp_{time} = \frac{6m32s}{6m4s} = 107.69\%$
- $cmp_{cmd} = \frac{21.2}{21.2} = 100\%$

For the full system we obtain:

- $cmp_{time} = \frac{6m17s}{6m4s} = 103.57\%$
- $cmp_{cmd} = \frac{14.3}{21.2} = 67.45\%$

This means that, on average, with only motor imagery commands we sent the same commands as with the keyboard while the trajectories of the robot was probably a bit worse than the keyboard one because we used a 7% on average more time. The results derived by the full system shows us that, with the system, we sent 32.55% fewer commands than with the keyboard but we spend 3.57% more time.

7 Conclusion

This thesis had the purpose to present a new statistical model combining two types of BCI paradigms: P300 evoked potential and motor imagery. This model was studied to drive a robot through the space from a starting point to a Goal. As first in this thesis we wanted to introduce the reader to Brain-Computer Interfaces, we explained which types exists, as invasive, partially-invasive and non-invasive. Focusing on the last one we presented tree paradigms used, Visual Evoked Potential, P300 Evoked Potential and Motor Imagery, and their differences. Then we presented the closed loop used in BCI and its five elements: the Control paradigm, the Measurement, the Processing, the Prediction and the Application and the end of the Chapter 2 we presented the Hybrid BCIs.

In the Chapter 3 we presented the instrumentation and the library that we used for the test or we thought they could be used to reach the purpose. After this we presented out statistical model based on Gaussian distributions and how we codified the BCI commands as input for the model showing a test environment to understand how the system evolves with the commands. Then we presented the real implementation of the system. The implementation is based on ROS and this allows the system to be easily portable on a real system. We showed the main components of the system and how they works and communicates each others.

The Chapter 5 shown the experiments made, initially in she simulated environment to understand better how the system evolves in more situation and changing some parameters. Then it shown the experiments made in the Gazebo environment that simulates real robot and scenarios. From the Gazebo experiments we saw that the model proposed brings some advantages. Comparing the test makes with only motor imagery commands and the test with also the P300 commands we have see a performance improvement about times and number of commands sent. With the hybrid system, using both the motor imagery commands and P300 commands, the user sends, on average, seven commands less and use twenty fewer seconds than using only motor imagery commands to make the same path and this is important because to send commands with BCI is difficult and the actual BCI does not allow to send commands too close to each other. Comparing the hybrid system with a normal keyboard system to pilot the robot we saw that with keyboard the robot uses less time to reach the goal but the commands sent from the user were the 32% grater than the commands sent by the user with the hybrid system. The main problem of this solution to pilot a robot is that the robot needs time to identify the P300 targets and their positions have a great influence on the performance of the system.

In the future we want to implement the system on a real robot using the commands sent by the real BCI. After that, the model is easy to integrate in a shared control policy with obstacles detection and others features to have a better navigation. The purpose would be to develop BCI system that can be used to create more and more useful devices to make the life of the people better and easier.

References

[1] R. N. I. R.-I.-H. I. M. S. . T. F. Khalid, M. B., “Towards a brain computer interface using wavelet transform with averaged and time segmented adapted wavelets,” *2nd International Conference on Computer, Control and Communication*, 2009.

[2] G. P. B. Graimann, B. Allison, “Brain-computer interfaces: a gentle introduction,” *Springer*, vol. 1, pp. 1–27, 1964.

[3] E. Fetz, “Operant conditioning of cortical unit activity,” *Science*, vol. 163, pp. 955–958, 1969.

[4] D. F. E Fetz, “Operant conditioning of specific patterns of neural and muscular activity,” *Science*, vol. 174, no. 1, pp. 431–435, 1971.

[5] J. Vidal, “Towards direct brain–computer communication,” *Annu Rev Biophys Bioeng*, vol. 2, pp. 157–180, 1973.

[6] ——, “Real-time detection of brain events in eeg,” *IEEE*, vol. 65, pp. 633––664, 1973.

[7] W. L. N. B. T Elbert, B Rockstroh, “Biofeedback of slow cortical potentials,” *Electroencephalogr Clin Neurophysiol*, vol. 48, pp. 293—301, 1980.

[8] E. D. LA Farwell, “Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials,” *Electroencephalogr Clin Neurophysiol*, vol. 70, no. 6, pp. 510—523, 1988.

[9] G. N. C. F. JR Wolpaw, DJ McFarland, “An eeg-based brain-computer interface for cursor control,” *Electroencephalogr Clin Neurophysiol*, vol. 78, pp. 252—259, 1991.

[10] D. M. JR Wolpaw, “Multichannel eeg-based brain-computer communication,” *Electroencephalogr Clin Neurophysiol*, vol. 90, pp. 444—449, 1994.

[11] X. P. L. H. Wang, "Adaptive shared control for a novel mobile assistive robot," *Mechatronics*, vol. 19, no. 6, pp. 1725–1736, 2014.

[12] K. A. Tahboub, "Natural and manmade shared-control systems: an overview," *ICRA*, pp. 2655–2660, 2001.

[13] J. W. Q. Li, W. Chen, "Dynamic shared control for human-wheelchair cooperation," *Robotics and Automation*, pp. 4278–4283, 2011.

[14] P. D. Nisbet, "Who's intelligent? wheelchair, driver or both?" *Control Applications*, vol. 2, pp. 760–765, 2002.

[15] E. J. P. F. S. A. Poncela, C. Urdiales, "A new efficiency-weighted strategy for continuous human/robot cooperation in navigation," *Systems, Man and Cybernetics*, vol. 39, no. 3, pp. 486–500, 2009.

[16] X. Perrin, "Semi-autonomous navigation of an assistive robot using low throughput interfaces," *Eidgen¨ossische Technische Hochschule ETH Zurich*, no. 18680, 2009.

[17] J. M. W. G. J. R. Millan, F. Renkens, "Noninvasive brain-actuated control of a mobile robot by human eeg," *Biomedical Engineering*, vol. 51, no. 6, pp. 1026–1033, 2004.

[18] Y. L. L. Bi, X.-A. Fan, "Eeg-based brain-controlled mobile robots: a survey," *HumanMachine Systems*, vol. 43, no. 2, pp. 161–176, 2013.

[19] E. L. P. W. F. G. V. J. P. J. d. R. M. F. Gal´an, M. Nuttin, "A brain-actuated wheelchair: asynchronous and non-invasive brain–computer interfaces for continuous control of robots," *Clinical Neurophysiology*, vol. 119, no. 9, pp. 2159–2169, 2008.

[20] E. L. P. W. F. F. G. M. J. P. H. V. B. M. N. G. Vanacker, J. del R Mill´an, "Context-based filtering for assisted brain-actuated wheelchair driving," *Computational intelligence and neuroscience*, vol. 2007, pp. 3–3, 2007.

[21] S. J. Y. Chae, J. Jeong, "Toward brain-actuated humanoid robots: asynchronous direct control using an eeg-based bci," *Robotics*, vol. 28, no. 5, pp. 1131–1144, 2012.

[22] M. R. L. D. T. C. J. D. R. M. R. Leeb, L. Tonin, "Towards independence: a bci telepresence robot for people with severe motor disabilities," *Proceedings of the IEEE*, vol. 103, no. 6, pp. 969–982, 2015.

[23] D. S. R. Siegwart, I. R. Nourbakhsh, "Introduction to autonomous mobile robots," *MIT press*, 2011.

[24] A. C. E. M. L. T. Gloria Beraldo, Morris Antonello, "Brain-computer interface meets ros: A robotic approach to mentally drive telepresence robots," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[25] E. W. JR Wolpaw, "Brain-computer interfaces: something new under the sun," *Brain-computer interfaces: principles and practice*, vol. 90, 2012.

[26] J. C. S Sanei, "Eeg signal processing," 2007.

[27] R. W. Polikov VS, Tresco PA, "Response of brain tissue to chronically implanted neural electrodes," pp. 1–18, 2005.

[28] G. S. R. P. N. R. E. C. Leuthardt, K. J. Miller and J. G. Ojemann, "Electrocorticography-based brain computer interface–the seattle experience," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 14, no. 2, pp. 194–198, 06.

[29] J. R. W. D. J. McFarland, "Brain-computer interfaces for communication and control," *Communications of The ACM*, vol. 54, no. 5, pp. 60–66, 2011.

[30] K. H. E. Marieb, "Human anatomy physiology," 2006.

[31] H. Jasper, "The ten-twenty electrode system of the international federation." *Electroencephalogr. Clin. Neurophysiol*, vol. 10, pp. 371–375, 1958.

[32] M. Q. R. Mandeep Kaur, P. Ahmed, "Analyzing eeg based neurological phenomenon in bci systems," *International Journal of Computer Applications*, vol. 57, no. 17, 12.

[33] D. J. M. G. P. T. M. V. Jonathan R Wolpaw, Niels Birbaumer, "Brain–computer interfaces for communication and control," *Clinical Neurophysiology*, vol. 113, no. 6, pp. 767–791, 2002.

[34] X. G. B. H. S. G. Y Wang, R Wang, "A practical vep-based brain-computer interface," *IEEE Trans Neural Syst Rehabil Eng*, vol. 14, no. 2, pp. 234–239, 2006.

[35] Y. W. B. H. S. G. G Bin, X Gao, "Vep-based brain-computer interfaces: time, frequency, and code modulations," *IEEE Comput Intell*, pp. 22–26, 2009.

[36] C. H. W. K. K. S. Y. W. P. L. Lee, J. C. Hsieh, "Brain computer interface using flash onset and offset visual evoked potentials," *Clin. Neurophysiol.*, vol. 119, no. 3, pp. 605–616, 2008.

[37] X. R. G. S. K. G. F. Guo, B. Hong, ""a braincomputer interface using motion-onset visual evoked potential," *J. Neural Eng*, vol. 5, no. 4, pp. 477–485, 2008.

[38] G. C. K. J. M. Middendorf, G. McMillan, "Brain-computer interfaces based on the steadystate visual-evoked response," *IEEE Trans. Neural Syst. Rehab. Eng.*, vol. 8, no. 2, pp. 211–214, 2000.

[39] G. S. S. D. Z. M. M. J. J. R. W. B. Z. Allison, D. J. McFarland, "Towards an independent brain-computer interface using steady state visual evoked potentials," *Clin. Neurophysiol*, vol. 119, no. 2, pp. 399–408, 2008.

[40] E. E. Sutter, "The visual evoked response as a communication channel," *IEEE Trans. Biomed. Eng.*, vol. 31, no. 8, pp. 583–583, 1984.

[41] J. Hanagata and K. Momose, "A method for detecting gazed target using visual evoked potentials elicited by pseudorandom stimuli," *Proc. 5th Asia Pacific Conf Medical and Biological Engineering and 11th Int. Conf. Biomedical Engineering (ICBME)*, 2002.

[42] R. S. Jian Ding, George Sperling, "Attentional modulation of ssvep power depends on the network tagged by the flicker frequency," *Cerebral Cortex*, vol. 16, no. 7, pp. 1016–1029, 2006.

[43] D. Regan, “Some characteristics of average steady-state and transient responses evoked by modulated light,” *Electroenceph. Clin. Neurophysiol*, vol. 20, pp. 238–248, 1966.

[44] M. S. C. B.-E. A. A. E. Basar, T. Demilrap, “Oscillatory brain dynamics, wavelet analysis, and cognition,” *Brain Lang*, vol. 66, pp. 146–183, 1999.

[45] D. Regan, “Evoked potentials and evoked magnetic fields in science and medicine,” *Human Brain Electrophysiology*, 1989.

[46] V. A. W. M. A. W. WG Walter, R Cooper R, “Contingent negative variation: an electric sign of sensorimotor association and expectancy in the human brain,” *Nature*, vol. 203, pp. 380–384, 1964.

[47] J. Z. E. J. S Sutton, M Braren, “Evoked correlates of stimulus uncertainty,” *Science*, vol. 150, pp. 1187–1188, 1965.

[48] D. S. E Donchin, “The contingent negative variation and the late positive wave of the average evoked potential,” *Electroenceph clin Neurophysiol*, vol. 29, no. 201–203, 1970.

[49] F. C. S. B. D. J. M. T. M. V. J. R. W. Dean J Krusienski, Eric W Sellers, “A comparison of classification techniques for the p300 speller,” *Journal of Neural Engineering*, vol. 3, no. 4, 2006.

[50] D. T. J. D.J.Krusienskia, E.W.Sellersb, “Toward enhanced p300 speller performance,” *Journal of Neuroscience Methods*, vol. 167, no. 1, pp. 15–21, 2008.

[51] M. T. Cuntai Guan and J. Wu, “High performance p300 speller for brain-computer interface,” *IEEE International Workshop on Biomedical Circuits and Systems*, 2004.

[52] M. J. Jeannerod, “Mental imagery in the motor context,” *Neuropsychologia*, vol. 33, no. 11, pp. 1419–1432, 1995.

[53] M. J. V. B. B. T. R. W. J. M. F. F. J. Decety, D. Perani, “Mapping motor representations with positron emission tomography,” *Nature*, vol. 371, pp. 600–602, 1994.

[54] C. N. G. Pfurtscheller, "Motor imagery and direct brain- computer communication," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001.

[55] M. D. S. L. Bressler, "Event-related potentials," *Wiley Encyclopedia of Biomedical Engineering*, pp. 412–415, 2006.

[56] M. Ahn, M. Lee, J. Choi, and S. Jun, "A review of brain-computer interface games and an opinion survey from researchers, developers and users," *Sensors (Basel, Switzerland)*, vol. 14, pp. 14 601–14 633, 2014.

[57] M. G., "Studies on the axon membrane; a new method." *J. Cell. Physiol.*, vol. 34, pp. 351–382, 1949.

[58] T. S. Cole K. S., "Ions, potentials and the nerve impulse," *Electrochemistry in Biology and Medicine*, pp. 121—140.

[59] K. N. Robinson H. P., "Injection of digitally synthesized synaptic conductance transients to measure the integrative properties of neurons," *J. Neurosci.*, vol. 49, no. 157, 1993.

[60] A. L. F. M. E. Sharp A. A., O'Neil M. B., "Dynamic clamp: computer-generated conductances in real neurons." *J. Neurophysiol.*, vol. 69, pp. 992–995, 1993.

[61] M. E. Prinz A. A., Abbott L. F., "The dynamic clamp comes of age." *Trends Neurosci.*, vol. 27, p. 218, 2004.

[62] E. Goaillard J.-M., Marder, "Dynamic clamp analyses of cardiac, endocrine, and neural function." *Physiology (Bethesda)*, vol. 21, pp. 197–207, 2006.

[63] B. T. Destexhe, A., "Dynamic-clamp: From principles to applications." *Springer.*

[64] F. R. W. J. A. Economo M. N., Fernandez, "Dynamic clamp: alteration of response properties and creation of virtual realities in neurophysiology." *J. Neurosci*, vol. 30, pp. 2407–2413, 2010.

[65] V. K. G. M.-P. C. N. G. P. B. Allison, C. Brunner, "Toward a hybrid brain–computer interface based on imagined movement and visual attention," *J. Neural Eng.*, vol. 7, no. 2, pp. 1–9, 2010.

[66] J. J. F. C.-Y. L. D. H. E. Yin, Z. Zhou, “A novel hybrid bci speller based on the incorporation of ssvep into the p300 paradigm,” *Neural Eng.*, vol. 10, no. 2, pp. 1–9, 2013.

[67] R. O. P. L. G. R. M.-P. G. Pfurtscheller, T. Solis-Escalante, “Self-paced operation of an ssvep-based orthosis with and without an imagery-based ∘ brain switch: A feasibility study towards a hybrid bci,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 4, pp. 409–414, 2010.

[68] S. P. R. Panicker and Y. Sun, “An asynchronous p300 bci with ssvep-based control state detection,” *IEEE Trans. Biomed. Eng.*, vol. 58, no. 6, pp. 1781–1788, 2011.

[69] C. B. G. B. T. S. R. S.-T. Z. G. M.-P. C. N. N. B. G. Pfurtscheller, B. Allison, “The hybrid bci,” *Frontiers Neurosci.*, vol. 4, pp. 1–11, 2010.