UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Bioingegneria

EMG-DRIVEN MOVEMENT DECODING AND CONTROL OF AN HUMANOID ROBOT

Laureando

Andrea Cimolato

Relatore

Prof. Enrico Pagello

Co-relatore

Dr. Luca Tonin

ANNO ACCADEMICO 2014/2015

"Strength does not come from physical capacity.
It comes from an indomitable will."

Mahatma Gandhi

Contents

Abstract

Currently, a wide range of applications in medicine take advantage of robots usage, but this relationship has not yet revealed its full potential. Even if the the idea of robots able to replicate human operations has already been proposed to the popular culture from decades, performing a coordinate human-like movements requires the use of advanced tools, like movement reconstruction and kinematics modelling. Humanoid robots can already autonomously perform complex tasks through human gesture and speech. Instead, implementations of full robotics controls derived from biological signals, like Electromiography (EMG) and Electroencefalography (EEG), are still not fully accomplished, despite the many attempts in the researching panorama. Therefore, the aim of this thesis is to create a reliable system, for real-time applications, able to decode and then replicate, on a humanoid robot, the same motor task of a subject, using as an input only EMG signals. In order to accomplish this result, this study is first undertaking a target-oriented classification employing Support Vector Machine (SVM). The successive analysis, implements a different approach of end effector trajectory reconstruction and deriving from it the kinematic that has to be applied to the robotic limb. The use of Multivariate Linear Regression (MLR) models and subsequent Kalman filter for the prediction correction have produced noteworthy results. Moreover, the computational time performance of the algorithm made clear that a real-time application is actually possible. Future developments of this study could be helping the design of new robotic device and exoskeletons able to support patients with neuro-muscular dysfunctions and moreover develop a new concept of rehabilitation and physiotherapy.

Chapter 1

Background and Motivations

Believing or not, robots have turned nowadays to an important, if not essential, part of human reality. Industry, home automation, army, healthcare, vehicles: these are just few of the many area where robotics has found a fertile field for developing and expanding new concepts. The first rudimentary robot appeared in the society starting from the sixties. *The Unimate* was introduced in the industrial process in 1961 [1] and it was designed for accomplish dangerous operations for the human operator. Robotic systems have become since increasingly accurate and complex, solving already an unlimited number of tasks aiming to a future total automatic controlled process.

1.1 Healthcare and robots: A new horizon

In this perspective, where robots are designed as a support and, if necessary, as a replacement of human activities, a great affinity is found between robotics and the medical environment, leading to a promising revolution of the healthcare industry. A wide range of applications in medicine currently take advantage of robots usage, including drug manufacturing, monitoring vitals, dispensing drugs to patients, and performing surgeries.

An other interesting field of application regards robotics introduction in physiatry, a branch of medicine that aims to enhance and restore functional ability and quality of life to those with physical impairments or disabilities. Many rehabilitation procedures are set up on the concept of neuroplasticity[1]. After brain tissue degeneration, due to stroke or aneurysm, the neuro-connectivity of patients' brain is undermined and if the motor-planing area is involved this could lead to the loss of motor ability. In this case, rehabilitation aims to restore, at least partially, the brain-muscles connection by

[1] Ability of brain to modify his neurons connection under specific stimuli.

inducing in the brain new neural paths. This means that until now medical practitioners had to manually induce re-activation of peri-lesional regions by means of a repetitive passive movements. Of course, the extensive use of human operators is a huge economical burden for the hospitals. This way, the introduction of assistive robotic device permits to single practitioners to treat at the same time multiple patients and to cut the costs of rehabilitation. On the other hand, the use of actual technologies able to communicate with patients through biological signals could improve drastically the performance of patient care, especially for those affected by motor disabilities. The adoption of EMG-driven and EEG-driven movement decoding could be used to counterbalance to degenerative diseases like muscular dystrophies or amyotrophic lateral sclerosis. Theoretically would be possible, once is generated a proper model, to read the degenerated bio-signals from the disabled patient and afterwards predict and produce the movement, for instance, in a exoskeleton for supporting and helping him in completing the motor task. The same perspective can be held regarding humanoid robots. They could be used as an alternative effector for interacting with the world by patients who are unable to wear exoskeletons. In this case, research is focusing on the use Brain-Machine Interface (BMI) for the robot control using as a input EEG signals, adressing this technology also to patients in locked-in status.

The idea of robots able to replicate men operations and acting like humans has already been proposed to popular culture thanks to television, media and entertainment from decades. Humanoid robots can already autonomously perform complex task decomposition required to high-level commands given through human gesture and speech [2]. Furthermore, there have been also by now a successfully implantation of prosthetic limbs with neurological connection [3]. Humanoid robotics presents to us much more than a interaction tool, but a research instrument for understanding how human body and brain work and interact eache other. The biomechanics studies about physiological systems and functions have found a new stimulation and a new perspective to look into human motor mechanisms. Pathological conditions that once seemed untreatable could have now new openings, providing either tools to design rehabilitation strategies or instruments and simulation environment for surgical procedure, like Da Vinci Robot®[4].

What is important to understand is that humanoid robots and exoskeletons have a very different configuration with respect to today's industrial robots. Performing a coordinate human-like movements, besides, requires a different planning and the use of advanced tools like movement reconstruction and kinematics modelling. The introduction of humanoids robot may as well have brought a new perspective in the way of thinking about human-machine interface. Humans have learned since in developing and adapting their abil-

ities to use of technology through use of keyboards, joy-pad and monitors. Humanoid robots are changing the way we used to think of technology interaction: technology has learn how to adapt to us, no more the other way round. The concept of performing a "spontaneous interconnection" between human and robots with respect to deal with other constrained modalities, took rapidly place in the robotic research panorama. Many studies, in fact, today are based on the idea that human motion and/or physiological signals, like EMG and EEG, could actually provide all the necessary information for models creation.

1.2 Aim of the thesis

In this context, my thesis aims to show that it is possible to reconstruct the trajectory of a limb's patient from EMG signals. From litterature [2][5] is known that at the current state of art it is not possible manipulating only the EMG signals to reproduce univocally the exact limb kinematics. This is due to the fact that the extracted features from the EMG data are not straightforward related to Degrees of Freedom (DOF) to be handled. In fact, different motor tasks or different modalities of task execution could be reflect by the same EMG. Moreover, EMG acquisition is affected by high noise presence that prevents us to capture the exact muscular activation using only extra-cutaneous sensors. Taking into account these limitations, the purpose of this thesis is, thus, not to reconstruct the entire limb kinematics during a motor task counting only on EMG, but exclusively the end effector's one. The goal of this study is to provide a starting tool for the design of rehabilitation/support exoskeletons: these results could be afterwords developed in a more powerful instrument for designing robotic prosthetic limbs.

Specifically, the mayor interest of this thesis is not only to implement an algorithm able to decode different tasks and replicate the subject movement starting from EMG, but also to design it with a fast computational time in order to perform the process in real-time. For this reason I decided to divide my research in two part: respectively target-oriented classification and kinematic reconstruction analysis. The first part of the study aims to present the problem of EMG-driven decoding and it will try to solve it through the extraction of parameters suggested from literature[5][6][7][8]. The chosen parameters are then used to feed a Linear SVM for the trials classification. This section should be therefore a preamble, showing off that is actually possible to discriminate different tasks from EMG signals. The kinematic reconstruction analysis, instead, will try a different approach by relating directly the end effector kinematics to the EMG acquistion. Linear regression model is

for this preferred, applying a linear predictor function which parameters are estimated directly from the distinct channel samples. Last, the classifier is tested in an on-line simulation and the estimated trajectory points finally addresed towards the robot via a Robotic Operating System (ROS) network.

1.3 Physiology of motor system

Before going any further with the thesis development, here is displayed a short introduction to the physiology of the motor system. A full understanding of muscles physiology and of EMG generation process is necessary to recognize how these physiological signals are related to muscular contraction and motor production.

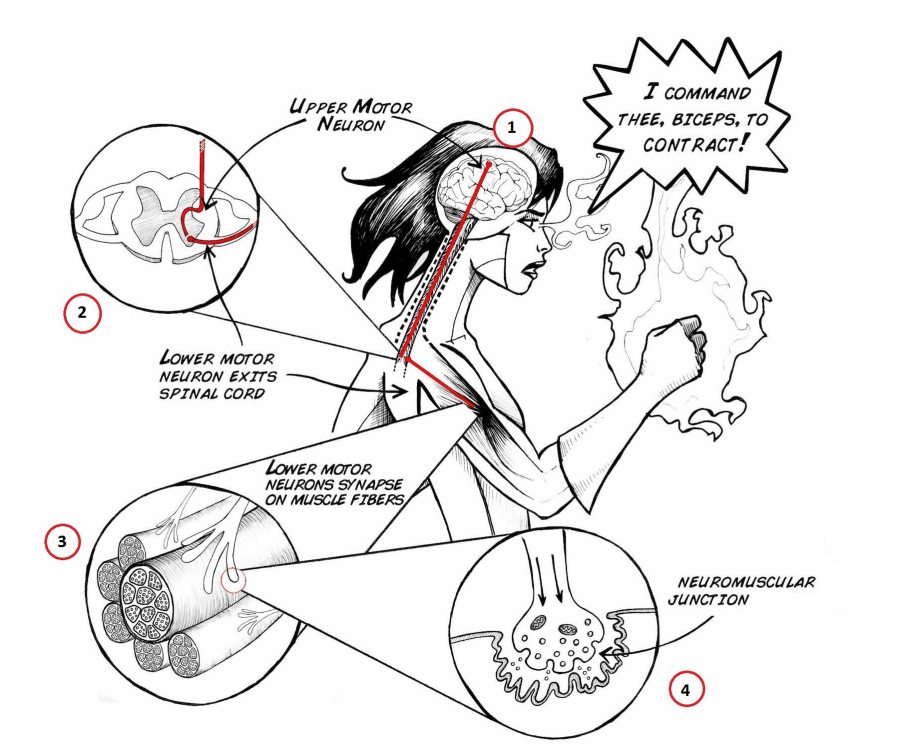


Figure 1.1: Neuro-muscular activation: 1) Upper motor neuron runs down the corticospinal tract. 2) Synapse with the *alpha* motor neuron inside the spinal cord. 3) The lower motor neuron innervates the muscle fibers. 4) The neuronal signal activates the acetycholine release in the neuromuscular junction, causing action potential and consequently muscular contraction[9]

Movement is the result of the combination of the nervous system and muscles, which interconnection is known as the neuro-muscular system. As

displayed in Figure 1.1, high level commands in the brain end up in descendant nervous signals that are charged for skeletal muscles control. When a action is planned in the motor cortex, the brain recollects a motor scheme and decides which muscle has to be activated to tackle the specific task. The *upper* motor neuron is thus stimulated and the neuronal signal sent down to the spinal cord. These particular nerve cells hold long axons, those starting from the brain cortex go into the spine where they synapse with their respective *lower* motor neurons. *Lower* motor neurons, or *alpha* motor neurons, connect them-self directly with the voluntary muscle that was planned to move: these two elements together constitute the motor unit[2](Figure 1.2).

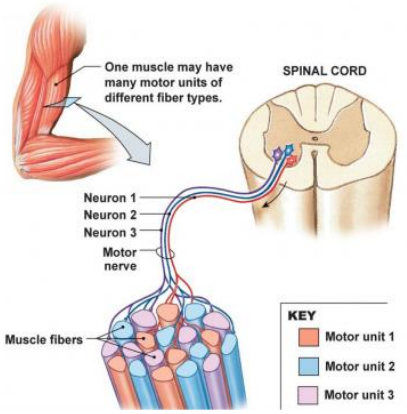


Figure 1.2: Motor unit representation

What is interesting in muscle fibers is that they are a particular type of cells capable of changing their size thanks to myosin/actin chains sliding across each other. A typical muscle is served by several alpha motor neurons and each neuron is subdivided in proximity to the muscle into tiny branches. The end of these subdivisions (presypnatic terminal), with its connection with the muscular fibers, is named neuromuscular junction. The electrical signal from the brain travels down the nerves and prompts the release of the chemical Acetylcholine (ACH) from the presynaptic terminals. This chemical is picked up by receptors on sarcolemma[3] changing, thus, the electrical potential on the muscular fibers. An ionic difference is maintained by ion pumps between the inner and outer spaces of a muscle cell, generating a resting potential at the membrane around -80mV. Looking at Figure 1.3, once the electrical potential is taken over the threshold (roughly +30mV),

[2]It is the smallest functional unit to describe the neural control on muscular contraction process: the elements that compose it are cell body and dendrites of a motor neuron, multiple branches of its axon and the muscle fibers that innervates it.

[3]Sarcolemma is the cell membrane of a striated muscle fiber cell.

thanks to ACH stimulation, an actual action potential occurs. The action potential propagates across the muscle membrane, causing voltage gated calcium channels to open. This event triggers a cellular cascade that finally causes muscle contraction.

Summing up the neuromuscular transmission steps:

1. nerve action potential;
2. calcium entry into the pre-synaptic terminus;
3. release and diffusion of ACH;
4. consequently ACH combination with post-synaptic receptor and End Plate Potential (EPP) increase;
5. opening of Na^+/K^+ channels;
6. post-synaptic membrane depolarization up to muscle action potential firing.[10]

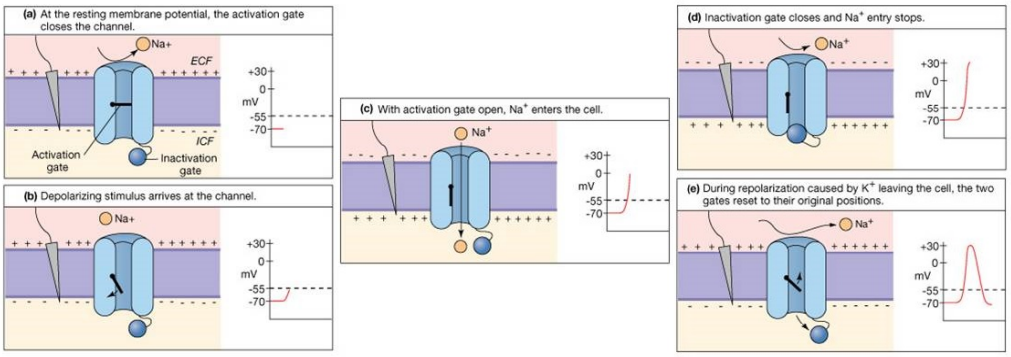


Figure 1.3: Depolarization-Polarization cycle within excitable membrane

1.3.1 EMG signals

EMG can be recorded using different types of electrodes: the most commons one are surface electrodes and fine wire. The first modality is among all the most used. Despite fine-wire Electromiography (fEMG) can provide a better quality and muscle-specific signals (especially for deep muscles), it has to face its invasiveness and the discomfort for the subject. Simultaneously, setting up a surface Electromiography (sEMG) is really easy, totally ergonomic and

ensure anyway satisfying acquisitions [11]. This methodology is used as well for the data set of this thesis.

Myoelectric signals are formed by physiological variations in the state of muscle fiber membranes [12]. EMG is a diagnostic medicine technique for evaluating and recording the electrical activity of voluntary muscles. EMG is performed using an instrument called an electromyograph, to produce a records called electromyogram. Studying EMG allows not only to observe muscle response but to analyze and measure muscular performance in order to create new bio-medical models or treatments.

EMG signals are based on the action potential recording at the muscle fiber membrane resulting from depolarization and repolarization processes above explained [13]. However, it is not obvious how this activity is reflected and described in the electromyogram. Since a motor unit consists of many muscle fibers, what the electrode "see" is the magnitude of all the fibers belonging to the motor unit: the so called Motor Unit Action Potential (MUAP) can differ in size and form, depending on fiber disposition. Moreover, it has to be taken into account the superposition of MUAPs. In fact, each electrode detects all the active motor units under its site, as shown in Figure 1.4. The signal resulting from the acquisition is therefore named *interference pattern*, or more commonly *raw* EMG [13].

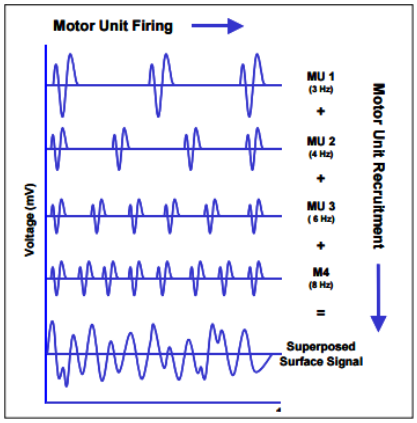


Figure 1.4: Recruitment and firing frequency of motor units modulates the muscle torque and is reflected in the superposed EMG signal [14]

The central nervous system has the ability to modulate muscular contraction and therefore, muscle strength thanks two main mechanisms: increasing the number of active motor units (spatial recruitment) or increasing the individual motor unit firing rate to take advantage of the summed generated tension (temporal recruitment) [15]. For this reason, these two strategies together are the most influencing factors on the magnitude and density of a

interference pattern. Thus, stronger is the torque generate from the muscle, higher will be the recruitment characterizations on the EMG signals. Raw sEMG can range between +/-5000µV with distinctive frequency contents range between 6 and 500Hz, showing most frequency power between ∼20 and 150Hz [13].

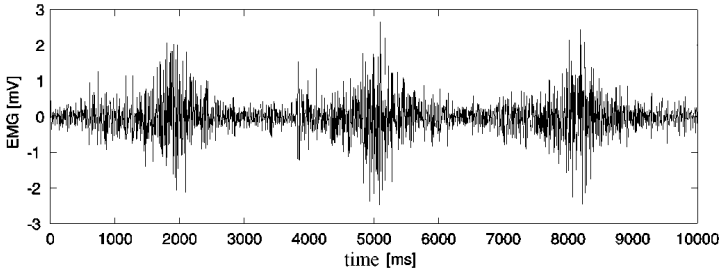


Figure 1.5: Raw EMG recording of Vastus Lateralis muscle

From engineering perspective, these characteristic frequencies are very important for filters design during the pre-processing phase. Noise can influence the signal in different circumstances. Signal distortions could appear due different tissue characterization: tissue type, thickness, physiological changes and temperature may vary electrical conductivity between the electrode and the motor unit. In addition *Cross-Talk* phenomena, external noises and alterations in the geometry between muscle belly and electrode site could lead to artifacts introduction. Furthermore, by its nature, raw EMG spikes are of random shape [13] (Figure 1.5). This means that one raw recording burst can be conceptualized as the output of a random stochastic process[4]. Therefore, it is not possible to precisely reproduce the exact shape of EMG envelope, even if the motor task is the same. This is due to the fact that the actual set of recruited motor units constantly changes within the matrix/diameter of available motor units [16].

1.3.2 Arm muscular anatomy

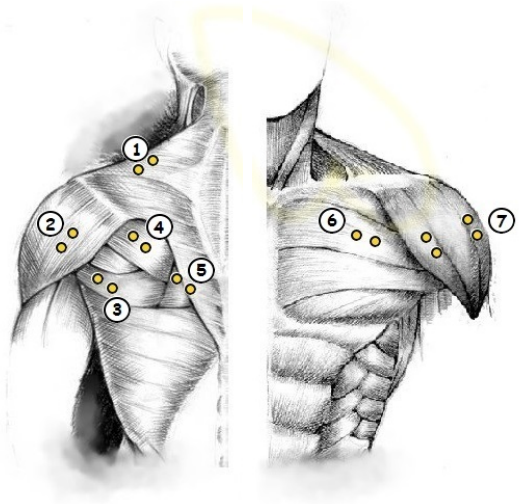
In is shortly shown a summary of the muscular anatomy of a human superior limb. Table 1.1 lists the principal muscles involved in arm motion. In Figure 1.6, 1.7 is possible to see their anatomic representation, respectively, of muscles located in the torso and in the arm. The yellow dots reflect approximately the sensor site for the sEMG.

[4]A stochastic process is a collection of random variables, representing the evolution of some system over time.

Tagged Muscles		
Torso	DeltA	Anterior Deltoid (Clavicular)
	DeltM	Medial Deltoid (Acromial)
	DeltP	Posterior Deltoid (Spinal)
	TrapSup	Superior Trapezius (Descendent)
	RhombMaj	Rhomboid Major
	Infrasp	Infraspinatus
	TeresMaj	Teres Major
	PectClav	Pectoralis Major (Clavicular part)
Arm	TrLat	Lateral (Head) Triceps
	TrMed	Medial (Head) Triceps
	BicShort	Short (Head) Biceps
	BicLong	Long (Head) Biceps
Forearm	BrRad	Brachioradialis
	Supin	Supinator
	Brac	Brachialis
	PronTer	Pronator Teres

Table 1.1: Table of the tagged muscles in the sEMG acquisition. First column marks the anatomical location; second column shows the label used in the data set and in the code; the last one presents the names of muscle.

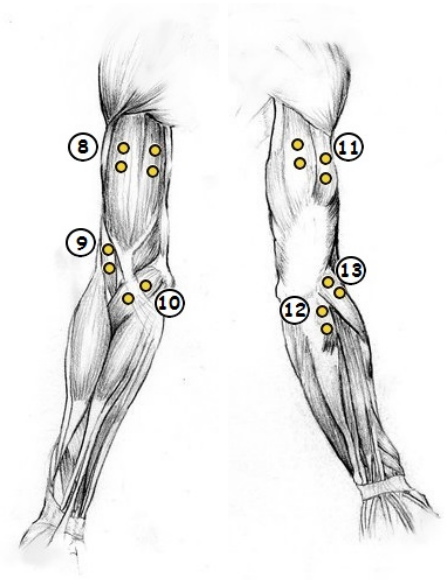
Although my study does not involve muscular modelization and therefore does not imply the knowledge of anatomy, a good understanding of how each muscle affects a basic movement permits to look at the sEMG with a more critical attitude. In fact, thanks to EMG it is possible to observe which of the tagged muscle actually has been active during the movement execution. Exploiting both this information it is possible to understand, at least roughly, which is the task that the subject is undergoing to. On the other hand these observations may result useful during a possible check up post acquisition. It is always reliable take a quick look to EMG acquisitions and control for suspicious and unexpected action potential firing before undertake massive data processing and analysis.



(a) Torso anatomical representation (back)

(b) Torso anatomical representation (front)

Figure 1.6: Yellow dots represent electrodes pair sites in EMG: 1.TrapSup, 2.DeltP, 3.TeresMaj, 4.Infrasp, 5.RhombMaj, 6.PectClav, 7.DeltA & DeltM. Label corresponding to Table 1.1



(a) Arm anatomical representation (front)

(b) Arm anatomical representation (back)

Figure 1.7: Yellow dots represent electrodes pair sites in EMG: 8.BicLong & BicShort, 9.Brac, 10.PronTer, 11.TrMed & TrLat, 12.Supin, 13.BrRad. Label corresponding to Table 1.1

For example during arm flexion we expect activation from Deltoid, Coracobrachialis, Pectoralis Major, Trapezius and others torso muscles. However, if other electrical activities are picked up it could imply presence of noise likely due to set-up errors during the electrodes positioning.

1.4 Technique for EMG classification

All the techniques implemented for the EMG analysis and classification rely on simple concept of "feature". A feature is defined as the value of a certain parameter related to the channel from which it is extracted. Once a parameter is determined in a N-channel EMG collection, it is then possible to extract N feature which are then used for classification problems. The parameters follow mainly two approaches: amplitude signal analysis or time-frequency representation.

The first method is currently based on the recognition and classification of EMG patterns that encode the information. This information is then used to estimate muscles activity from the EMG and finally detect the kind the task. Even if these techniques result to be very successful, it has been demonstrate that amplitude and rate of change of the EMG were not enough to reliably control more than one actuator [17]. To solve this problems researches tried to increase the number of possible states of the models using multiple-channel acquisitions or other statistical measures, such as entropy [18]. Others, instead, preferred a different approach looking to a time-series modelization of the EMG: the results encouraging but unfortunately very sensitive to change of signal amplitude and envelope. On the other hand, Hudgins et al. found out that EMG exhibits approximately the same structure during the initial phase of muscle contraction [8]. Time-domain features such as zero-crossing, mean absolute value and slope were extracted from several EMG segments and then they are classified using an artificial neural network.

Regarding the latter approach, instead, the main instrument for studying the frequency components of a EMG signal is the Fast Fourier Transform (FFT). Every signal, including EMG, can be considered as a summation of sine waves with different oscillatory frequencies (Formula 1.1). The Discrete Fourier Transform (DFT) algorithm decompose that signal to its underlined sinus contents: basically the coefficients coming out from the DFT give a quantitative measure of how much the current signal could be described with that frequency sine wave.

$$X_k \stackrel{\text{def}}{=} \sum_{n=0}^{N-1} x_n \cdot e^{-2\pi i k n/N}, \ k \in \mathbb{Z} \tag{1.1}$$

The approach based on time-frequency has shown that a representation on both the two domains could actually drastically improve the classification performance. Feature extraction is usually implemented thanks to Short-time Fourier Transform (STFT) or Wavelet Transform (WT) [7]. The mean and median frequency are more over the most important parameters for analyzing the frequency components of the EMG signals[6]. The study of Reaz at al. [6] demonstrates that these two parameters increases with the increase of muscle contraction, meaning that it is possible to distinguish different muscle actions using these characteristic frequencies. The changes in the power spectrum in fact are caused from the presence of newly recruited motor units during the task. The rising firing rate of the MUAP is reflected in the increase of the mean and median frequency (translation of the spectrum toward higher frequencies). Procedure of wavelet decomposition have been approached from many researchers and there are many papers displaying real-time classification thanks to WT and STFT. All these studies anyway were applied on EMG acquisitions with a low number of channels. In fact the WT requires a great amount of computational load and for a data set of 16-channeled sEMG it could take too much time, overtaking real-time boundaries.

1.5 NAO robot

Figure 1.8: Aldebaran NAO robot v3.3 (Academic edition)

The final purpose of this thesis is to replicate the same movement of a subject into a robotic element. The NAO robot (Figure 1.8) has been chosen for this objective as the most applied humanoid robot in education, research and supported by many different programming environment. The

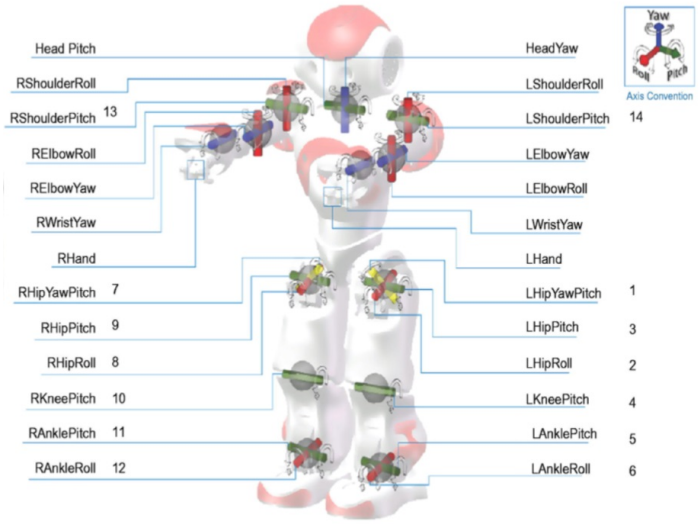


Figure 1.9: Caption

NAO robot is developed by the worldwide leading manufacturer of humanoid robots Aldebaran Robotics and is on the market since 2008 [19]. The Nao robot is equipped with a 1,6GHz *Intel atom* processor, two HD cameras, four microphones, two loudspeakers as well as a WLAN interface. Besides, NAO provides several sensors to perceive its environment (More detailed documentation of the NAO platform can be found under Aldebaran-Robotics) [19]. The NAO robot is equipped with five kinematic chains (head, two arms, two legs). It is 58cm tall and it has about 5Kg of mass.

1.5.1 Arm kinematic chain

The NAO's arm is an articulated manipulator capable of interacting with the environment. It is possible to represent this kinematic chain as the sequence of mechanic links connected each other by joints. In NAO robot, the arm is composed entirely of rotatory joints, except for the fingers junction. The combined action of the actuators generate different configuration of the robotic links, but not all the position are valid or feasible due to positioning constrain and collision between links themselves. It is for this reason defined the "*joint space*" reflecting all the valid combinations of joint rotation value. The total number of actuators in a kinematic chain corresponds to its DOF: higher are the DOF more flexible is the robot. Robot kinematics is the application of geometry to the study of kinematic chains with multiple degrees of freedom. More specifically, robot kinematics provide the transformation

from the joint space, where the kinematic chains are defined, to the Cartesian space, where the robot manipulator moves, and vice versa [20].

Each arm of the NAO has 5 DOF:

- Shoulder Pitch
- Shoulder Roll
- Elbow Yaw
- Elbow Roll
- Wrist Yaw

One of the leading reason for implementing this study on NAO robot is the great similarity between its Range of Motion (ROM) to the human's one. The comparison between the two of them is displayed in Table 1.2. The close affinity between the two kinematic chains, indeed, makes much more easier the replication and the control of the end effector motion in to the robotic element.

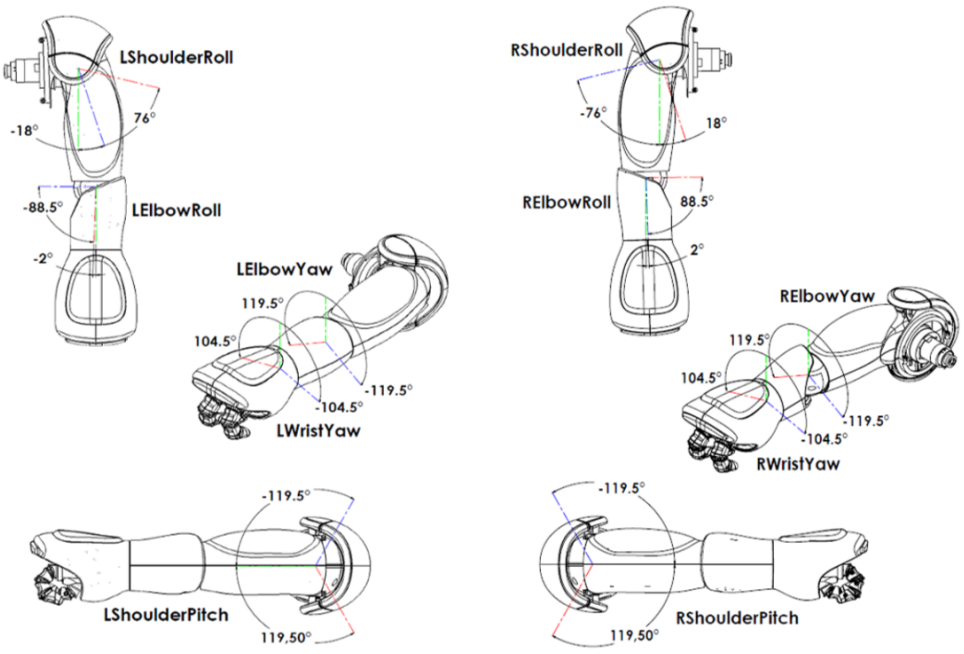


Figure 1.10: NAO's joint ROM

	ROM	
Joint Name	**NAO**	**Human**
LShoulderPitch	[-119.5°, 119.5°]	[-180°, 60°]
LShoulderRoll	[-18°, 76°]	[-45°, 180°]
LElbowYaw	[-119.5°, 119.5°]	[-180°, 75°]
LElbowRoll	[-88.5°, -2°]	[-150°, 0°]
LWristYaw	[-104.5°, 104.5°]	no range
RShoulderPitch	[-119.5°, 119.5°]	[-60°, 180°]
RShoulderRoll	[-76°, 18°]	[-180°, 45°]
RElbowYaw	[-119.5°, 119.5°]	[-75°, 180°]
RElbowRoll	[2°, 88.5°]	[0°, 150°]
RWristYaw	[-104.5°, 104.5°]	no range

Table 1.2: Comparison between the NAO's and human's joint ROM [21]

Robot kinematics is a powerful instrument to control movements execution as well as to calculate actuator forces and torque. There are two main analysis in mechanical links and joints: forward and inverse kinematics.

Despite of the fact that control of actuator is managed by coordinates-free in the joint space, these values result to be actually very little informative to the understanding of the end effector position and orientation. The forward kinematics, for this reason, creates a map from the joint space to the 3D Cartesian space. This means that given a set of joint values (θ_1, θ_2, ..., θ_m), the forward kinematics gives back the position (p_x, p_y, p_z) and the orientation (a_x, a_y, a_z) of the final element of the kinematic chain respect to the global coordinates system. Forward kinematics strength is underlined from the fact that it is independent from the domain, providing always an analytical solution.

Anyhow, usually a common task for a robot manipulator is to follow trajectories or get to a target point. This implies the fact that the objective to solve is usually handed over as a 3D point in Cartesian space. To reach this goal is necessary to project the end effector to the goal, specifying appropriate values of the joints rotation. The inverse kinematic basically computes the reverse action of the forward analysis: produce a transformation from the Cartesian space to the joints one. On the contrary the inverse kinematic

is not a domain-independent problem and it is possible to get different solutions in the joint space (different link configurations) corresponding to the same target point. As the number of DOF increases, a point in the 3D space may have more than one matching points in the joint space. This multiplicity of solutions makes the inverse kinematics a relation, not a mapping [20]. The inverse kinematics can be solved or in analytical way (Closed-form equation) or using numerical method with iterative approximation (Jacobian approximation method).

The NAO arm kinematic chain refers to a mathematical model consisting in an assembly of rigid bodies (links) connected by joints. Each link is constrained by their connection to other links and they are able to change configuration only thanks to actuators action. Taking in to account this, the translation and orientation of a joint j with respect to an adjacent joint i in the 3D space can be full described using a 4×4 affine transformation matrix T_i^j:

$$T_i^j = \begin{bmatrix} & & & \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \\ & \hat{R} & & \\ & & & \\ \begin{bmatrix} 0 & ... & 0 \end{bmatrix} & & & 1 \end{bmatrix} \tag{1.2}$$

where $\hat{R} \in \mathbb{R}^{3x3}$ and vector $dX(= [dx\ dy\ dz]^T) \in \mathbb{R}^3$. The affine transformation matrix T_i^j provides at the same time the translation dX (related to the link dimensions) and orientation (returned by sub-matrix $\hat{R}$) of the coordinate system j with respect to coordinate system i [20][22].

For each robotic manipulator of N joints, there is an equal number of left-handed Cartesian coordinate systems. Every frame is then linked to the previous one thanks to affine one-by-one transformation. For convenience, in a manipulator the frame enumeration starts from an established base frame, typically a fixed point on the robot's body. In the case of the NAO arm, the shoulder is chosen as starting joint [22]. A point described in frame j can be therefore transformed as a point in another frame coordinates by cascading the transformations for all intermediate frames:

$$T_i^j = T_i^{i+1} T_{i+1}^{i+2} ... T_{j-1}^j \quad and \quad \bar{p}_i = T_i^j \bar{p}_j \tag{1.3}$$

Denavit and Hartenberg [23] established a formalism for describing transformations between two frames adjacent to a joint. The points in one end of a joint are described with respect to a coordinate system that is consistent with the previous frame, as function of the joint state. They concluded

that to fully describe this transformation matrix it was necessary only four parameters, known as Denavit-Hartenberg (DH) parameters: a, α, d, and θ.

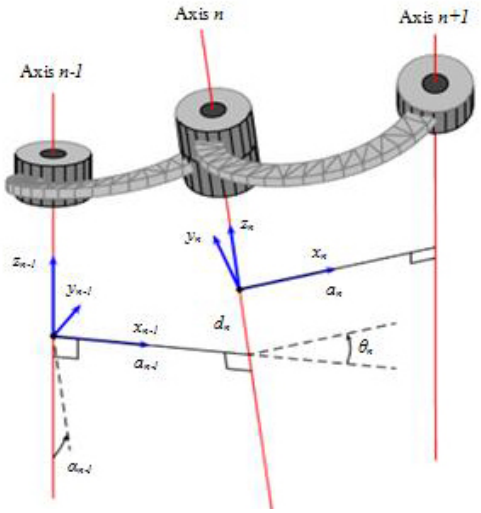


Figure 1.11: Denavit-Hartenberg scheme and notation on two consecutive frame

This convention establishes a specific arrangement between the two consecutive reference frame, displayed in figure 1.11.

1. z_i-axis is aligned to the direction of the joint axis of action
2. x_i-axis is parallel to the common normal between z_i and z_{i-1} ($x_n = z_{n-1} \times z_n$)
3. y_i-axis follows from the x_i and z_i axes to form a right-handed coordinate system.

Once settled this conventions, the DH parameters are defined as:

- $\boldsymbol{a}$: length of the common normal
- α: angle about the common normal, from z_i and z_{i-1}
- $\boldsymbol{d}$: offset along the z_{i-1}-axis to the common normal
- θ: angle about the z_{i-1}-axis, from x_{i-1} and x_i

Thus, the transformation from the (i-1)-th reference frame to the i-th is accomplished using the affine transformation matrix T_{DH}, obtained thanks

the multiplication of four matrix, consisting in the two rotation and in the two translation, represented by the four DH parameters:

$$T_{DH} = R_x(\alpha)T_x(a)R_z(\theta)T_z(d) \tag{1.4}$$

This equation lead to the analytical form:

$$T_{DH} = \begin{bmatrix} cos\theta & -sin\theta & 0 & a \\ sin\theta cos\alpha & cos\theta cos\alpha & -sin\alpha & -dsin\alpha \\ sin\theta cos\alpha & cos\theta sin\alpha & cos\alpha & dcos\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

The kinematic chain for the NAO left arm is composed of five joints. Accordingly, to obtain the expression of the end effector, referred to the reference base frame, five sets of DH parameters are necessary. Table 1.3 shows the parameters that are used for each DH transformation matrix in NAO arm kinematic chain. $A(dx, dy, dy)$ represents a translation matrix of a vector $dX(= [dx\ dy\ dz]^T)$; $R_j(\theta)$ is a transforamtion matrix corresponding to a rotation around the j-axis of θ angle.

Frame (Joint)	**a**	α	**d**	θ
Base Translation	A(0, ShoulderOffsetY, ShoulderOffsetZ)			
LShoulderPitch	0	$-\frac{\pi}{2}$	0	θ_1
LShoulderRoll	0	$\frac{\pi}{2}$	0	$\theta_2 + \frac{\pi}{2}$
LElbowYaw	ElbowOffsetY	$\frac{\pi}{2}$	UpperArm Length	θ_3
LShoulderRoll	0	$-\frac{\pi}{2}$	0	θ_4
LWristYaw	0	$\frac{\pi}{2}$	LowerArm Length	θ_5
End Effector Rotation	$R_x(-\frac{pi}{2})R_z(-\frac{pi}{2})$			
End Effector Translation	A(HandOffsetX, 0, -HandOffsetZ)			

Table 1.3: Denavit-Hartenberg parameters sets related to each frame of the NAO arm kinametic chain; the different $theta_i$ represents the joint angle coordinate-free of the rotatory actuators

Introducing the values reported in Table 1.3 to the analytical solution of Equation 1.5 it is easy to calculate the final transformation matrix as concatenation of the single DH transformations:

$$T_{Base}^{End} = A_{Base}^{0} T_0^1(\theta_1) T_1^2(\theta_2) T_2^3(\theta_3) T_3^4(\theta_4) T_4^5(\theta_5) R_x(-pi/2) R_z(-pi/2) A_5^{End} \tag{1.6}$$

1.5.2 NAOqi

NAO must interpret and move according to the data that it is able to pick up from the environment. This is where the embedded software in its head comes in. NAOqi is the operating system provided by Aldebaran that allows the small humanoid to understand the data received by its sensors.

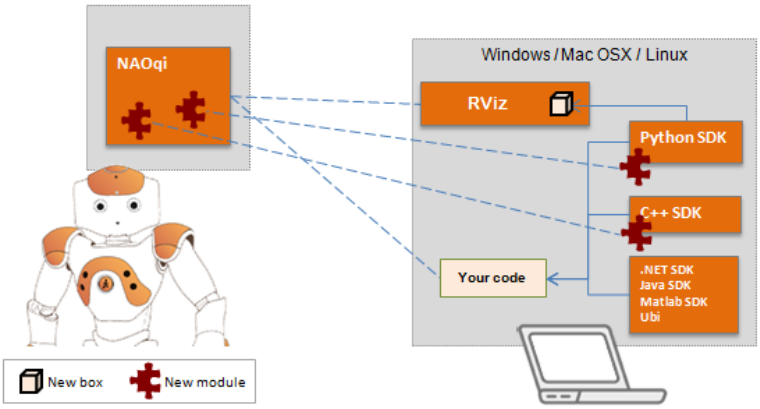


Figure 1.12: Schematic representation of NAOqi remote control

The humanoid robot NAO provides the NAOqi Software Development Kit (SDK). This framework includes a programming interface to develop applications in different programming languages like C++, Java or Python. The NAO operating system lets the developers the possibility to control directly NAO hardware components. The core of the NAOqi SDK represents the main broker which acts as a server running on a Linux kernel inside the robot [19]. Different modules consent the access to joints motion, face recognition and other function incorporated in this framework. The SDK, besides this, grants to developers flexibility in applications development, those could be run remotely from a laptop. The NAOqi, in this study, was directly installed in a Laptop supported by Linux operating system and it is used to

simulate NAO behaviour, later displayed in RViz[5] environment. In Figure 1.12 is represented a short schematic representation of how the connection was set up between NAOqi and the remote control.

1.6 ROS

ROS is a free software framework for robotic purpose that offers many tools for communication between robots and other devices or software through a TCP/IP network.

ROS functionality are split in two parts. The core software contains the different API to create a ROS network and to create the ROS-nodes that can connect to the network. It also contains scripts and command-lines to monitor nodes connections, the exchanged messages on the network and to interface ROS with other software such as RViz. On the other hand, ROS provides a very large packages database referenced in ROS main website [24].

The best service of ROS is its communication network. Communication is based on the TCP/IP protocol with each node connecting with a socket. The server is administrated by a master that handles all the connection and addressing details. The principal components of a ROS network are:

- Nodes: they are process that can perform computation, execute some tasks and communicate through the network. ROS provides libraries to write the nodes with the C++ or the Python languages;
- Messages: they are packages of structured data sent on the network. A message is divided in fields. Different messages are distinguished from the field types;
- Topics: the topics are a transport system with publish and subscribe semantics used to send messages. A node can connect to a topic by its name either as a publisher in order to send data or as a subscriber in order to receive these data;
- Services: they are an alternative to the topics that does not use the publish/subscribe system but instead uses a request/response model. A node that uses service will only receive data in response to the query it made.

[5] RViz (ROS visualization) is a 3D visualizer for displaying sensor data and state information from ROS. Using RViz, it is possible to see the current configuration on a virtual model of the robot. You can also display live representations of sensor values coming over ROS topics including camera data, infrared distance measurements, sonar data, and more.

1.7 Thesis overview

The following chapter provides a overview of the database that was used for analysis and then it briefly outlines the acquisition protocol to better understand the starting point and then interpret the conclusions of this study.

Chapter 3 presents the target-oriented analysis. This part wants to show up how it is possible to classify a multi-channel EMG: multi-class SVM are used to label the motor tasks in different classes, those correspond to the target that as to be reached. Methods are presented and results discussed in way to drag conclusions about the chosen parameters for the classification.

In Chapter 4, the principal part of the project is described. The methods used for the kinematic reconstruction such as MLR and the correlated use of predictive filters are exposed. Then, once the results deriving from the trajectory reconstruction algorithm are discussed, the final part concerning the robot connection is covered.

In Chapter 5, finally, we draw some more conclusions about all the experience and from the online testing. Limitations of this study and possible future work are finally discussed.

At the end of this thesis are placed glossary, appendix about part of the code used for classification and bibliography.

Chapter 2

Experimental design

This chapter presents a brief overview of the data set recorded at the Foundation I.R.C.C.S. San Camillo (Venice, Italy)[1]. In the data set a single acquisition is organized in three different recording: EEG, EMG and end effector kinematics (position and velocity). The experiment took place in date 17 December 2013.

2.1 Patients

The experiment involved two patient of the hospital.

The first subject (BV), had a ischemic stroke in the left frontal-parietal of the brain cortex. The brain damage results in the loss of the motor coordination and ability on the right part of the body. Between the stroke and the acquisition had passed approximately ten months.

The second patient (FA), suffered from a massive ischemic stroke in the right cortex and sub-cortex (involving frontal, parietal, insular, temporal and occipital part). Between the stroke and the acquisition had passed two months only. It was not clear from the subject information if many small strokes occurred and loss of motor control was the result of the last one or instead it was only one event. For the second subject was recorded only one run. Since the amount of data for the first patient was fairly higher it was chosen as exemplar for the model.

[1]San Camillo foundation takes care of facilities designed as I.R.C.C.S., i.e. hospitals pursuing research in the bio-medical field and in the organization and management of services.

2.2 Experimental protocol

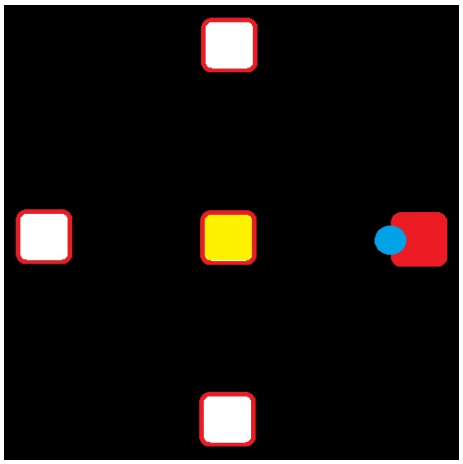


Figure 2.1: Example of possible screen displayed to the subject. The centered square in yellow is the starting point; the external squares represent instead the four possible target, the one filled in red is the goal; last, the blue spot stand for the pointer

On the computer screen was displayed five different squares: one in the middle representing the starting point and the other four at the cardinal points respect the first one representing instead the possible goals, as displayed in Figure 2.1. The patient was asked to repeat four different motor tasks with the superior limb, extending the arm in one of the four direction. One trial started when the pointer is placed correctly inside the starting square. After a time of $t_t = 2000ms$ (time of target delivery) a target square was randomly selected giving a visual output to the patient; the patient had to wait for a second output at $t_{lag} = 4000ms$ (lagging time) from the start before go ahead moving the manipulator towards the target. If the pointer, virtually describing the manipulator on the screen, exited before this second flag, the subject had to start over from the beginning. The trial ended when the pointer entered in the aimed target. Afterwards, it was asked to the patient to reposition the pointer inside the initial square and a new trial started. Each run consists of 80 trials randomly disposed, however for each target is reserved the same number of trials.

2.3 Data acquisition

The acquisition was set up in way that the patient, once arranged with the EEG cap and the EMG electrodes, had his hand on a manipulator with kinematic sensors. One acquisition was divided in three run: each run consist of 80 consecutive trials.

In our analysis, EEG recording was not taken into account. The EMG was captured in a unique acquisition for each run composed by 17 different channels. The first 16 channels are addressed to record a specific muscle activity. The number of channel is chosen in way to record as much more information as possible, avoiding simultaneously problems of cross-talking. To avoid these drawbacks only those muscles that are most influential on the limb motion of interest have been selected. The position of the 16 pairs of electrodes and the list of their respective muscles are shown in Chapter 1.3.2 (particularly attention to Figure 1.6, 1.7a, 1.7b and to Table 1.1). The last channel, labeled as "Trigger", produced a signal which was high when the pointer was inside a square (starter or goal) and instead assume zero value once was out. This signal is useful to us so that it is possible to label the EMG and kinematic data. sEMG is sampled at $1000Hz$. Before the begin of the analysis and the pre-processing of the signals it has been done a check up. Some inconsistencies were found on some of the EMG channels acquisitions. The more obvious was on the 8^{th} channel. The signal results in a constant high intensity noise. Probably the electrode on Theres Major muscle was broken or it have detached immediately after its set-up. Moreover the 16^{th} channel has been excluded from the analysis due to heavy noise alterations.

Velocity and position are actually recorded only when the "Trigger" signal is down. Basically, every kinematic recording starts when the cursor exits from the initial square and ends once the pointer entered in the goal: each run provided a set of 80 acquisitions. This acquisition strategy has proved not to be the best solution since it caused the loss of fundamental data. In fact, this experimental design led to kinematics recordings that actually do not present the starting condition, when the pointer is steady in the center of the plane. This means that the classifier is never fed with data related to stationary condition of the manipulator, probably constituting a weak case in the model. This limitation is going to be discussed later in a following chapter, with illustration of the related problems concerning the modelization. Position and velocity with respect to EMG has a sample rate of $100Hz$. This of course leads to problems of interpolation or down-sampling of the two different recordings.

Thanks to the EMG trigger and to the time samples acquired simultane-

ously to the kinematics data it was possible to coordinate the two acquisition and fit them together.

Chapter 3

Target-oriented classification

This chapter performs a preliminary analysis on the treatment of data in a off-line perspective. The parameters used to implement the classification are, in fact, computed from the entire length of a trial acquisition. This issue makes the following techniques and methods not suitable for a on-line application. Based on what is exposed Chapter 1.4, both the two type of analysis (amplitude and frequency domain related parameters) are adopted for the classification in order to make a comparison between the performance of both the two approach. According to literature [5][6][7][8] the features are extracted based on parameters that more characterize the sEMG evolution in both time and frequency domain:

TIME DOMAIN

- s_mean: mean magnitude of the rectified signal
- IEMG: area under the rectified signal
- ZCR: Zero-Crossing Rate (ZCR) of not-rectified oscillating signal
- in_percent: percentage contribution of each channel to total acquisition
- mean_peak: mean of the highest spikes of rectified signal

FREQUENCY DOMAIN

- f_mean: mean frequency of the filtered signal
- f_median: median frequency of the filtered signal

The label used in the in the previous list are the same used in the Matlab code (Appendix A, Listing A.1).

3.1 Methods

For s_mean, IEMG, in_percent and mean_peak parameters it is necessary a common previous EMG signal pre-processing before they are computed. In fact for amplitude related parameters is usually necessary compute the rectification. The first step is to band-filtering the EMG, according to the scientific finding exposed in Chapter 1.3.1. In this thesis, Butterworth band-pass filters[1] are implemented thanks to the function already present in Matlab main library, adopting a filter order equals to seven, a low-cut frequency of $4Hz$ and as high-cut frequency $200Hz$. To solve signal aleatoriness and to minimize its unpredictable part, instead, a full-wave rectification[2] is applied followed by a simple moving-average smoothing. The band-pass filtering computed before makes sure that signal is clean and hold null mean value in order that baseline does not affect the rectification. The combination of these actions leads to obtain at the end a "linear envelope" signal (Figure 3.1) that outlines the same information of the initial EMG [13].

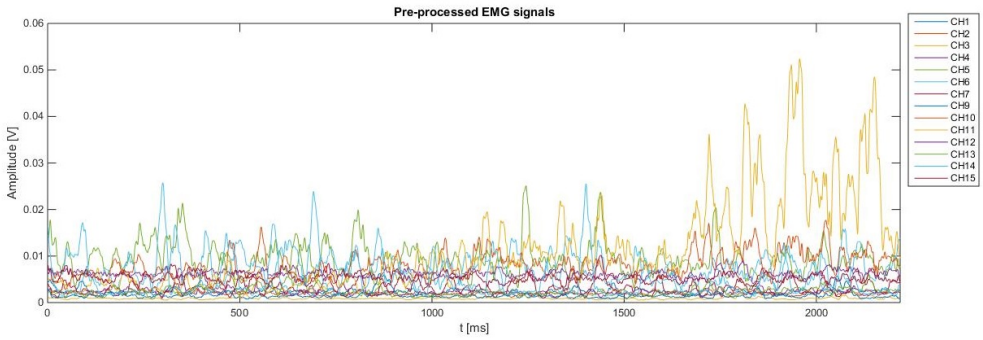


Figure 3.1: Matlab plot of the rectified EMG signals. The figure shows the acquisition of the selected channels of patient BV in trial n° 35

On the other hand, ZCR parameter exploits the fact that contraction gives large changes in potential: higher is the contraction, higher is the rate[5]. For this reason, even if it is a amplitude related parameter, ZCR is instead

[1]The Butterworth filter is a type of signal processing filter designed to have as flat a frequency response as possible in the passband [25].

[2]Full-wave rectifier implies the complete shift to positive of the negative values of the signal. It opposes to half-wave rectifiers those instead cut out from the signal the negative part.

extracted directly on the raw EMG: the signal is not pre-processed since filtering procedure may alter the ZCR trend. Some controls are instead implemented in order to not count noisy zero-crossing and supply to the absence of filters.

At last for the frequency-domain correlated parameters, only the filtering part is applied, in order to extract features related only to the EMG characteristic frequencies.

Appendix A.2 contains also the detailed definition of the chosen parameters and the correlated implementation in Matlab code for their computation.

At the end of the extraction algorithm therefore we should get for each trial and for each of the chosen parameters a features set, one for each channel. One trial, regardless to which parameter, is now marked by the feature vector, in this case of study a vector of 14 elements: basically one trial can be thought as a point in a 14-Dimensional space. Learning algorithms can be applied to this perspective in way to create supervised learning models those are able to use classification and regression analysis to cluster the multi-dimensional space.

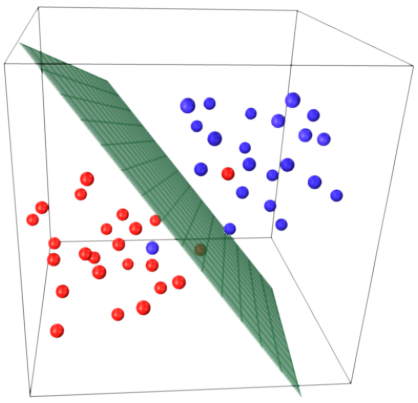


Figure 3.2: Graphic representation of a classification derived from multi-dimensional SVM

Given a set of training examples, each marked for belonging to one of the different class, a SVM training algorithm builds a model that assigns new sample into one class or the other, making it a non-probabilistic binary linear classifier. A SVM model is a representation of the parameters space subdivision, mapped so that the training elements of separate categories are divided by maximizing the distance between the element of different classes. In linear classifier the space subdivision is called modeling class boundaries with hyper-planes, i.e. based on the value of a linear combination of the characteristics. Test samples are then mapped into that same space and predicted to belong to a category based on which part of the space fall on. The Matlab function used in this master thesis project has been download from the MathWorks®web-site in the section of Matlab Central - File Exchange, since the multi-class SVM has been implemented in the main Matlab library only starting from the newly released Matlab R2015a. A multi-class SVM is necessary in our case due to the presence of four different classes represented from the four different motor tasks in the data-set. The function code was written by Cody Neuburger, researcher in the Florida Atlantic University

(Florida, USA) [26].

3.2 Results

Even if the time performance are actually discouraging and not suitable for a real-time application, it was possible at the end to look some interesting results after the testing phase. The classifier is tested and cross-validated using a Leave-One-Out (LOO) technique[3] on all the 240 trials.

Parameters	**Number of Classification Errors**	**Classification Error Frequency**
s_mean	16	6.67 [2.73,9.47]%
IEMG	20	8.33 [4.09,11.50]%
ZCR	87	36.25 [29.82,42.34]%
in_percent	25	10.42 [5.84,13.98]%
mean_peak	12	5.00 [1.43,7.39]%
f_mean	35	14.58 [9.47,18.79]%
f_median	40	16.67 [11.34,21.15]%

Table 3.1: Results of the models cross-validation with LOO on 240 trials (observations). First column contains the name of the parameters; the second one shows the total number of classification errors; the last column presents the classification error frequency with 95% of confidence interval, fitting the the error prediction with a binomial distribution

In Table 3.1 is displayed a comprehensive view of the cross-validation results. LOO cross-validation tests cyclically each observation, i.e. each trial, training the model on the rest of the data-set. It is possible in this way to test the SVM 240 times. The best result is obtained thanks to the *mean_peak* parameter, following a similar outcome with *s_mean* parameter: only 5% around of classification error. The result instead gets worst when we train the

[3]Cross-validation is a model validation technique, mainly used in setting where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. LOO cross-validation in particular involves using only one observations as the validation set and the remaining observations as the training set. This is repeated on all the different observations of the data set.

	Target 1 (Nord)	**Target 2 (East)**	**Target 3 (South)**	**Target 4 (West)**
s_mean	0.25	0.13	0.06	0.56
IEMG	0.15	0.10	0.05	0.70
ZCR	0.01	0.39	0.06	0.54
in_percent	0.20	0.08	0.08	0.64
mean_peak	0.33	0.00	0.17	0.50
f_mean	0.20	0.09	0.17	0.54
f_median	0.17	0.05	0.10	0.68

Table 3.2: The table displays the percentage contribution of each class (embodied by the target and its correspondent motor task) to the total error prediction for each parameter

model with frequency-domain parameters. Both mean and median frequency give back a 15% of class mismatch. The worst outcome is generated however training the classifier with the ZCR: over 35%. In Table 3.2, the attention is focused on the contribution that each class gives to the total error. The class referred to Target 4 happens to be the one that brings larger contribution to class mismatch, greater than 50% of the error for each parameter. Figure 3.3 gives a graphical representation of the Table 3.1 and 3.2.

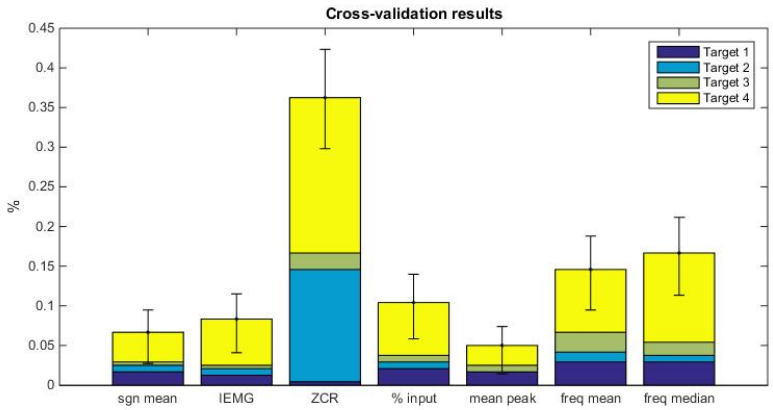


Figure 3.3: Plot of the classifier error prediction results

3.3 Discussion

Classifier performances show a significant improvement in case of using time-domain parameters, but for ZCR. This is probably due to the fact that the Fourier Transform compute the spectrum over all the trial signal, without windowing the signal. The high classification error could be attributed to the fact that some muscles had just a short contraction and the high frequency content of the burst is mitigated from the rest of spectrum when mean and median are extracted. On the contrary, it is possible to see that with just the mean of the signal the SVM is able to classify correctly over the 90% of the trials in the data-set.

The best performance anyway is shown from the mean of the peaks of the pre-processed EMG signal. It is fully understandable thinking about the nature of the peaks in a EMG. Superposition of MUAP increases the amplitude of the EMG, so stronger is the motor unit recruitment, higher are the correspondent burst peaks[13]. The later parameter is probably even less affected from the rest of the signal too. For example EMG mean, the area under the curve or the input per channel percentage are computed taking into account all the samples of the acquisition. Therefore it is possible that even if inside the signal there are significant change of pattern, they are lost during the computation of such parameters. Peaks detection instead counts only restricted number of values, this could explain the better result.

The worst outcome is obtained instead for the ZCR. Explanation to this can be charged to the same reasons displayed for the frequency-domain parameters related, but more likely the implemented controls were not enough to supply the absence of signal filtering. Observing the different run taken by the patient, it is visible a upward trend of the signal amplitude. This drift in the signal acquisition is probably due to muscular fatigue or sensor detachment and it is the reason why the regulative threshold in some trials is not enough to cut out the noisy zero-crossing.

An other remark as to be done to a noteworthy particular. In Figure 3.3 is evident that the major rate of error prediction happen for the class corresponding to Target 4. This is likely related to the choice to not taking into account the corrupted channels' acquisition. Teres Major in fact is involved in arm adduction and backward extension, instead Trapezius (superior fibers) manage the scapula rotation: both of them are involved for the motor task accomplishment.

This kind of analysis does not give back directly a kinematic response, but it is possible once a certain task is encoded with a class to associate that with a prefixed built trajectory. This is not a excellent solution since there is

only a limited set of actions and the paths of the end-effector have every time to be adjusted to the anthropometry of the subject. Moreover the patient can not perform customize movements and it ends to be quite hard to be employed in real-life situations.

The discussion of the obtained results made clear to the necessity of a new kind of approach which could be at the same time fast and not constrained to limited number of class, leaving to the users more freedom and trying to give to him a tool as user-friendly as possible. On the other hand anyway, it was observed that, without spending much effort in algorithm design for parameters extraction, the simple EMG mean value results to be good enough as feature in the target-oriented classification.

Chapter 4

Kinematic reconstruction analysis

The following chapter describs in the main part of the thesis. Using information gathered in the target-oriented analysis as a starting point, we want to show the possibility to use sEMG to decode upper limb trajectory and to control an humanoid robot. The implemented techniques and solutions used to solve problems related to the real-time application are here discussed. The reliability of this solution is in the end presented through performance discussion.

The main challange was the design of an algorithm for the trajectory prediction that would it be at the same time accurate and fast enough to keep up with a real-time application, such as the movement of robotic device. In order to meet this two objectives it was considered to split the kinematic reconstruction in two parts: a first component consisting in the actual prediction model, and a second filtering element able to fix the prediction errors of the first part. Accordingly to this consideration, first, a predictive model is built and developed on MLR and then, instructed on an observed training set. The fitted model is then used to estimate the trajectory of a end effector, only based on the data contained in test set. The best choice for the prediction corrector is here represented by Kalman filter. This type of predictive filters not only permits to overcome the inaccuracy of the MLR predictor, but moreover it perfectly suits the necessity for a low computation time.

4.1 Methods

The analysis is divided in two parts:

- Off-line training
- On-line simulation

Respecting this subdivision, we applied LOO technique where one trial it is randomly selected one trial from the data set, which it will be reserved for the on-line simulation (Test set). The remaining trails are instead employed for the training of the MLR model (Training set).

Filtering, full-wave rectification and digital smoothing is performed on the of the training set EMG signals, using the same procedure of the previous target-oriented analysis for amplitude related parameters.

Afterwards, Principal Component Analysis (PCA) is computed directly on the EMG channel acquisitions. PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This operation aims to obtain a dimensionality reduction of the training set and consequently the reduction of the algorithm computational time. The utilized function is already implemented in the Matlab main library. This function returns the principal component coefficient matrix, also known as weights: each column contains coefficients for one principal component (columns are in order of decreasing component variance). This is directly multiplyed to the training set and results in an uncorrelated orthogonal vector basis set. Of this principal components are kept only those with the largest variance, explaining the 96% of the total training set variance.

After the PCA, the MLR model has to be trained. Multivariate linear regression is an approach for modeling the relationship between a dependent set of correlated variables Y and one or more explanatory variables (or independent variable) denoted from vector X. In this case of study, the vector Y is represented from the kinematics data, velocity and position in the 3D space; X embodies the observations of the channeled EMG after PCA reduction. In linear regression, data are modeled using linear predictor functions. Linear regression refers to a model in which the values of Y are obtained from an affine function of X. The estimation algorithm calculates a series of parameters that are used to compute a linear combination of the PCA vectors, as close as possible to the position and velocity values. Thus, the model is fully characterized within a matrix of dimension dxK, where each column contains the d parameters for the linear combination (number of PCA components plus the constant value) of the K (equals to six) correlated coordinates of position and velocity in a 3D space. Practically, the "mvregress" Matlab function receives as inputs the matrices of the kinematics data and the PCA reduced EMG data and produces in output a single estimated coefficients matrix.

The last step, before moving to the on-line test is the implementation of the model used for the Kalman filtering. The Kalman filter is designed to

operate recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state. Two equations define the filter: the update state equation and the measures equation. The algorithm works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, exploiting the update state equation, along with their uncertainties. Once the outcome of the next measurements, corrupted by certain error/noise, is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. Because of the algorithm's recursive nature, it can run in real-time: only the present input measurements and the previously calculated state and its uncertainty matrices are necessary and no additional past information is required. The estimated trajectory out-coming from the MLR predictor is chosen as input for the Kalman filter. The update state equation is derived from a classic kinematic model of a moving point, with the assumption of null acceleration. The measures equation is instead a simply noise adder. More detailed information for filter implementation and related code are displayed in Appendix B.2. At the end of the off-line training, PCA reduction matrix, the MLR predictor and the Kalman filter model should be therefore handed over to the on-line simulation code section.

In a real-time application, the predictor algorithm has to work asynchronously from the acquisition In order to respect this consideration, the virtual acquisition is simulated simply scanning forward the test trial data and the trajectory estimation is applied iteratively on EMG signal buffers composed by 512 samples. The start of consecutive buffers are spaced from one another with offset equals to 32 samples, creating an overlap between successive buffers.

First, pre-processing and rectification is applied on a single buffer. Secondly, like in the off-line part, the dimensionality of the buffer is reduced thanks to PCA and the mean is then computed on the principal vector components. Therefore, an averaged observation is obtained from the single buffer made up of only a single sample. Afterwards, the calculated values are used afterwards as entry for the MLR, that projects them in a point coordinates of a 3D Cartesian space. Finally the position and velocity of the end effector is rectified applying the Kalman filter formerly designed.

The reconstructed kinematics data are sent to the NAO robot by means of to designed ROS network.

4.2 Results

The results presented in this section derive from ten different on-line tests. For each test it is performed the procedure explained in the methods section (off-line training and on-line simulation), choosing every time a different test set.

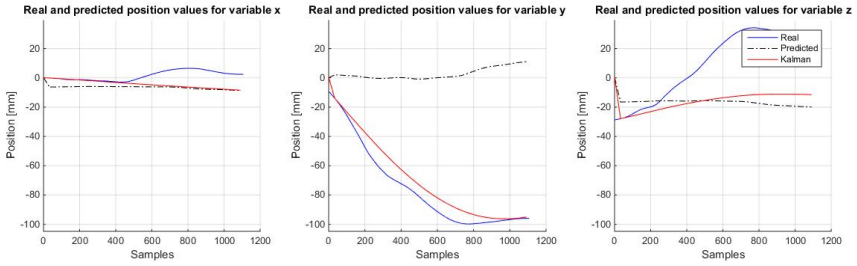
Figure 4.1, 4.2 and 4.1 display the obtained results at the end of three simulations; three representative trials (representing respectively good, bad and mean fit performance) are shown. In each of these figures a graphical representation of both position (on the top) and velocity (on the bottom) prediction is shown. The blue line is the actual kinematics recording, obtained from the acquisition session. In black is plotted the first prediction out-coming from the MLR model. The red curve, instead, represents the final result of the algorithm, after the Kalman filter. The three line are displayed on the same Matlab plot in order to get a better comparison between the estimation and the real kinematics.

Figure 4.4 instead reports the results of the goodness of fit of the Kalman filtered position values and the real trajectory: this parameter shows how good the test data fit the reference one. The Matlab function that was adopted in the code uses a cost function based on Normalized Mean Square Error (NMSE):

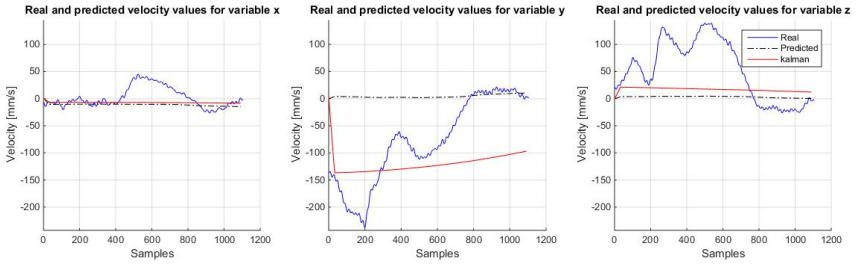
$$GoF = 1 - \left|\left| \frac{\bar{x}_{ref} - \bar{x}}{\bar{x}_{ref} - mean(\bar{x}_{ref})} \right|\right|^2 \tag{4.1}$$

where, || indicates the 2-norm of a vector.

NMSE costs vary between -Inf (bad fit) to 1 (perfect fit). If the cost function is equal to zero, then x is no better than a straight line at matching x_{ref}.

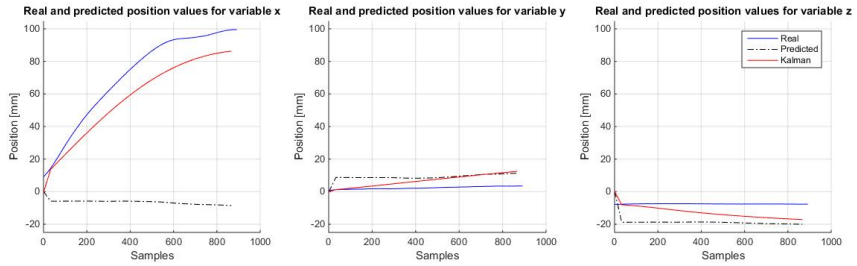


(a) Matlab plot of position prediction results

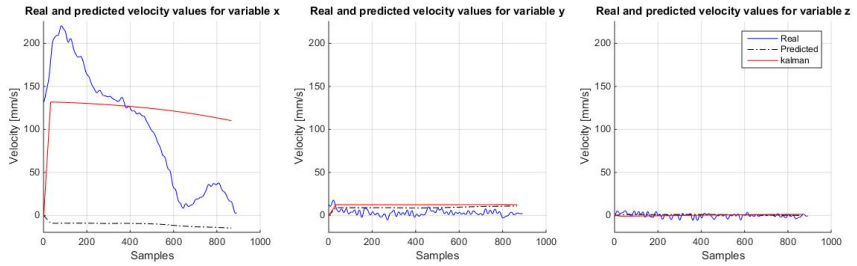


(b) Matlab plot of velocities prediction results

Figure 4.1: Trajectory prediction results on the first on-line simulation. The blue curve represents the real end effector kinematics, the dashed black line stands for the MLR prediction and finally the red line is the final result after Kalman filtering

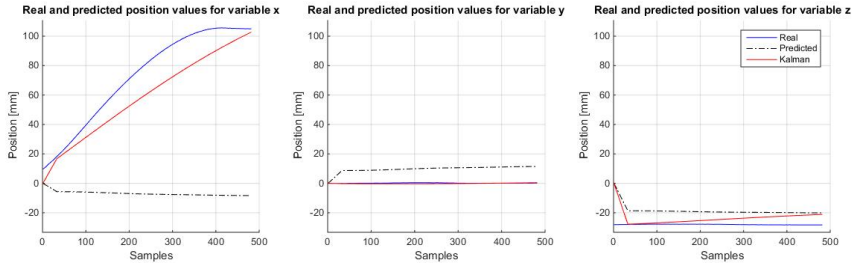


(a) Matlab plot of position prediction results

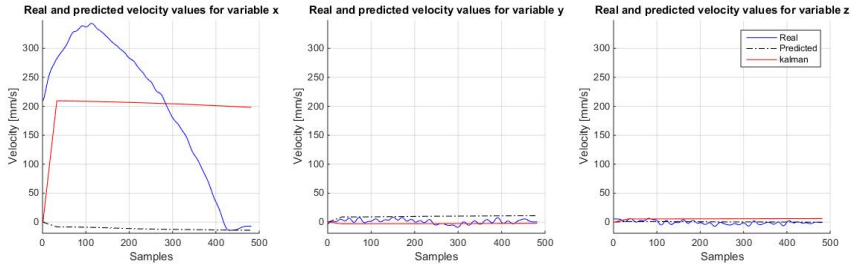


(b) Matlab plot of velocities prediction results

Figure 4.2: Trajectory prediction results on the third on-line simulation. The blue curve represents the real end effector kinematics, the dashed black line stands for the MLR prediction and finally the red line is the final result after Kalman filtering



(a) Matlab plot of position prediction results



(b) Matlab plot of velocities prediction results

Figure 4.3: Trajectory prediction results on the eighth test. The blue curve represents the real end effector kinematics, the dashed black line stands for the MLR prediction and finally the red line is the final result after Kalman filtering

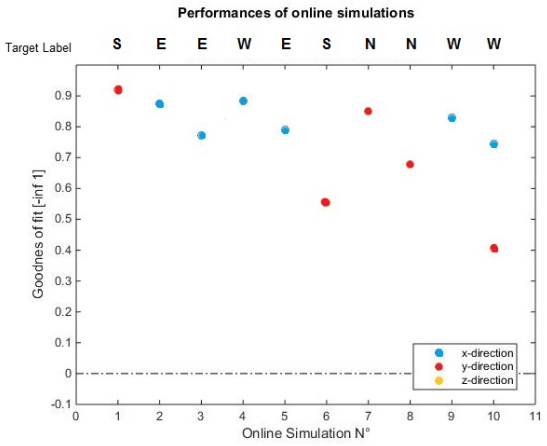


Figure 4.4: Matlab plot of the goodness of fit parameters for each on-line simulation, between the Kalman filtered position values and the real trajectory. The goodness of fit is calculated separately for each direction component. On the Top of the plot are placed the target label: N= North, S=South, W=West and E=East

Finally, in Table 4.1 is presented a comparison between the actual time for the trial execution and the computational time demanded from the trajectory

estimation algorithm for each on-line test.

Simulation N°	Real execution time (ms)	Algorithm execution time (ms)
1	1050	579
2	1502	1102
3	994	527
4	1429	1013
5	1404	985
6	2589	2245
7	1355	1618
8	486	303
9	1328	867
10	1224	767

Table 4.1: Comparison between the algorithm computational time and the actual trial duration. First column contains the label of the on-line simulation, second and third column, respectively, the demanding time for the trial execution and for the trajectory estimation. Time is expressed in milliseconds

4.3 Discussion

The first remark to be done is on the MLR model. It is certainly true that the output of the MLR predictor can not be directly employed as an effective signal for the robot control, but it is anyway interesting to note that prediction results really close to real trajectory. From the plots in Figure 4.1, 4.2 and 4.1, it is actually possible to see that the resulting pattern at least do not totally deviate from real one. If this had not happened, it would have been considerably more difficult obtain such good results after the prediction correction.

The same results, on the other hand, prove the real strength of the Kalman filter. The filter is actually able to rectify the distorted prediction based on a simply kinematic model of a moving point in a 3D space and prior statistical knowledge. In the Matlab plots it is evident that the goal is always met, at least in the leading direction. For the remaining two components, instead, there are often more evident fitting problems. This is particularly appreciable when the latter two trajectory components are stationary: the filter produces a noticeable drift toward the MLR estimation values. The reason probably lies on the fact that, for stationary signals, the Kalman filter action on the prediction step is not strong enough to overcome the error

introduced from MLR. This last observation is proved from Figure 4.4. The goodness of fit reaches always, at least for one direction, a particularly high value.

Anyway, Figure 4.4 seems to display worst result with respect to the graphical representation in Figure 4.1, 4.2 and 4.1, since for some trial the goodness of fit value go down under -10. This can be explained within the choice of the cost function used in the goodness of fit calculation. NMSE, as its name states, normalizes the fit error and for this reason in stationary curve even a small deviation can lead to a big NMSE values. Using the same logic, this explain why curves with significant variations in the trend and presenting the same amount of fit error, result actually in better performance.

The velocity prediction on the contrary do not provide great results, but in this study these values have an active role only in the Kalman filter, where they are employed for the estimation of the new end effector position. In fact, this thesis do not utilize velocity directly in the robot device control, but only position.

Talking about performance, it is noteworthy to take a look at Table 4.1. The evident conclusion that can be deduced from the table is that the computational time is significantly lower to duration of the trial. The buffer off-set acts directly on the computational time. The computational load can be further lowered, raising the spacing between consecutive buffers. Higher is the buffer off-set, lower is the output frequency of the predictive algorithm and less time is therefore computed the iterative estimation. At the same time lowering the output frequency could lead to less accuracy in the trajectory estimation values.

Anyway, employing a buffer offset of 36 samples ($\sim$28 buffer per second), this study was able to meet the real-time constraint and an excellent fit at once.

4.4 NAO robot control

Until now was only presented and discussed methods and techniques regarding the implementation of an algorithm for the EMG-driven movement decoding, but no reference has been addressed to the control of a robotic device. As stated in Chapter 1, once the part of trajectory prediction is successfully concluded, the out-coming data have to be used for the control of a robotic manipulator. The last issue that has to be solve is the creation of a comunication path between the output data from the Matlab program and the robotic element.

As presented in Chapter 1, a ROS network is the perfect tool to achieve

the realization of a communication channel with a robot device. Figure 4.5 displays the ROS computation graph of the implemented network for the NAO robot control.

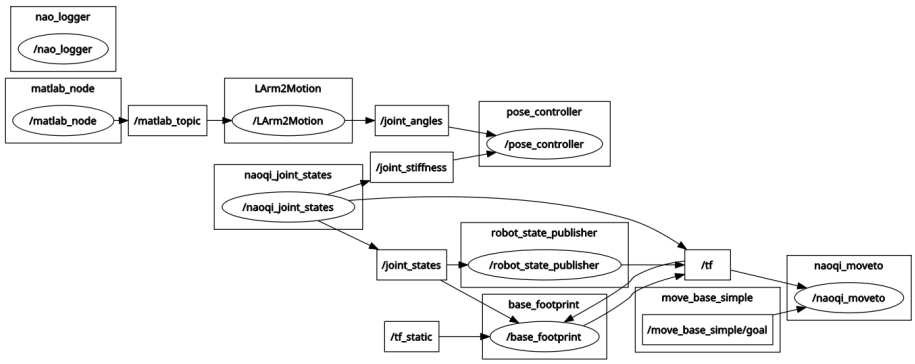


Figure 4.5: rqt_graph GUI plugin display of the ROS network created for the NAO robot control

The main part of the analyzed ROS network was actually already implemented from libraries which can be easily download from the ROS online database. This network section is already designed and it manages the communication with the NAOqi operating system, joint control and motion regulation, displaying the state information of the NAO robot in a RViz 3D model.

This thesis developed only a small portion of the displayed network: *matlab_node* and *LArm2Motion* ROS nodes. This two nodes are simply addressed to the exchange of topic between the Matlab process and the NAOqi operating system.

The *matlab_node* consists of a simply publisher node that works parallel with the trajectory prediction Matlab code.

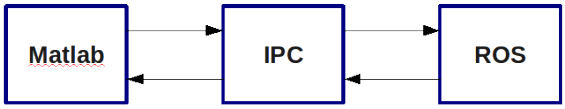


Figure 4.6: Schematic representation of the Matlab-IPC-ROS conectivity

The node is implented using IPC (Inter Process Communication)[1]. It is a software package for iter-process comunication employed for the creation of

[1]IPC provides flexible, efficient message passing between processes (Figure 4.6). It can transparently send and receive complex data structures, including lists and variable length arrays, using both anonymous "publish/subscribe" and "client/server" message-

bridge communication between ROS and Matlab. ROS cannot communicate directly with Matlab due to incompatible dependencies.

Following the data stream, once Matlab estimates the 3D trajectory point starting from the EMG data, the inverse kinematic has in first to be applied. As stated in Chapter 1.5.1 the control of the NAO's arm, in fact, is performed directly on the joint angle values. The code implementation of the Jacobian approximation method for inverse kinematic is taken from the repository attached to Kofinas Nikolaos Master Thesis project [20]. This numerical method provide finally the joint angles to be applied on the actuators in order to reach a particular position in the 3D Cartesian space with NAO's arm end effector.

The obtained free coordinates are then published on the *matlab_topic* toward the second node. The *LArm2Motion* ROS node is created instead in c++ language. The class implemented by the embodied c++ code performs two main operation. The first is the creation of a subscriber that gathers the messages coming from the Matlab code and the second is the creation of another publisher that reorganizes the structured information from the *matlab_topic* and in another topic that is acknowledged from the NAOqi API.

Finally, The *joint_angle* topic sends to the arms joints the updated angle values permitting the NAO to move the arm.

During the on-line simulation, it is simply necessary to start up the ROS network running the launchers from the terminal and once the the Matlab process is started, the virtual model in RViz starts to move accordingly to the prediction algorithm (Figure 4.7).

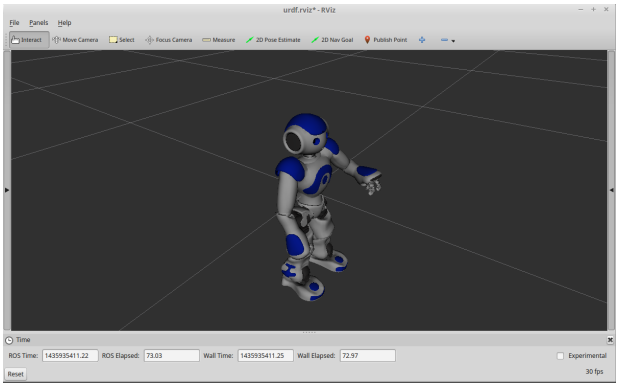


Figure 4.7: RViz program display of the NAO virtual model replicating a subject movement

passing paradigms. A wide variety of languages and operating systems are supported. (http://www.cs.cmu.edu/ IPC/).

Chapter 5

General discussion and outlooks

In conclusion, this study manages to accomplish the two main goals established at the opening of this thesis. The designed EMG-driven movement decoding algorithm proved a computational time fast enough to be actually employed in a real-time application and at the same time a satisfying performance in the trajectory fit. Finally, the control of an humanoid robot is implemented on the data out-coming from end effector trajectory estimator, by means of a ROS network set up. The Cartesian coordinates of the trajectory are then transposed in the joint space coordinates generating a fluently motion in the robotic limb, but above all, generating a movement that replicates the subject task.

5.1 Limitation of the case of study

The first drawback that can be made on the obtained results is that the algorithm is only tested on one subject. The choice to implement the study only on one patient was accordingly to the fact that only a small amount of trials were available for the second subject. The absence of an adequate number of trials made impossible to create a reliable predictive model.

An other clear limitation is that all the implemented models are trained on a data set that involves a low number of task classes. The fact that the patient performed only four kind of motor task actually could have helped the positive performance of the classifiers. The model has not been tested for customized movements, but given the results obtained so far we are confident that it can respond optimally even then.

For what concern directly the first analysis of motor task target-oriented classification, it is made clear that its main limitation lies in the issue that no kinematics information are actually produced in output. This actually related to the type of analysis that was undertaken. If a robotic device as

to be controlled subsequently a set of pre-selected trajectories have to be provide in advanced to the algorithm. Each of this trajectory is then labeled and assigned to the corresponding target class. This solution, as explained in Chapter 3.3, appear to be actually to much constraining for a real-time application, since the only freedom left to the patient is the choice of the goal.

Far less evident deficiencies can be found instead in the kinematic reconstruction analysis. The prediction performance are really satisfying and promising but it is nevertheless true that the on-line simulation initial condition is not the actual real trail start. In fact the kinematic recording of a single trial starts only when the virtual pointer leaves the initial square, as described in Chapter 2. This implies that the manipulator is already moving with a certain velocity when the data starts to be acquired. The lack of null velocity and position as a starting condition could have influenced somehow the performance of the predictive algorithm: the velocity and position values at the beginning of the acquisition may have "helped" the estimation toward the right direction.

A second shortcoming in this analysis is the total absence of a velocity control. The NAO robot is purely controlled only with joint angle values, without taking in to account the velocity of task accomplishment. The motion of the robotic end effector from one point to the consecutive in the trajectory is performed always with the same velocity. Therefore, when the subject accomplishes the same path with different velocities, NAO robot moves the arm with the same kinematics.

5.2 Future directions

Future works based on this study should first of all try to overcome the aforementioned limitations. Therefore, the algorithm should be tested employing different data sets on different patients and different kinds of task, making particular attention to the starting condition analysis.

Once the efficacy of this predictive model has been fully proved, the study can be further developed in new perspectives. The thesis aim to replicate the same movement of a subject into a robotic device. Even if the subject was affected from a neuro-muscolar dysfunction the employed EMG data were relative to the healthy limb. The EMG signals under analysis then are not corrupted respecting the standard parameters characterization. It is clear that, as the conclusions of this study are fascinating, and as limitless could be its applications in the robotic field, no benefit can be brought on a patient with motor dysfunctions. For this reason a further step would be

to improve the predictive algorithm in order to estimate the trajectory of a end effector employing EMG recordings of a non-healthy person. Therefore, the algorithm will have to be strong enough to estimate the trajectory even from corrupted EMG signals.

This new direction of the study could lead to novel applications in health care and develop a new concept of performing rehabilitation. Nowadays, modern physiotherapy has already employed the use of robotic devices, such as manipulators, which are used to help the patients in their rehabilitation process. The repetitive movement of the offended limb, even if passively, is a typical therapy strategy in order to restore motor abilities. In this contest the use of of exoskeleton combined with a EMG-driven decoding algorithm would create a new device able in first place to help the patient in carry on daily life motor tasks and simultaneously improve his/her pathological condition. Patients will not need anymore stressful rehabilitation sessions in the hospital but they will be able to carry them on simply wearing a robotic support.

On another front, this algorithm can be integrated with other systems able to decode movements from other biological signals, for example eye-movement or EEG. BMI is another promising field that in the past years have led to numerous accomplishments [27][28]. The combination of more systems for decoding patient movements could give to the patient a tool that will be, primarily, incredible reliable, but moreover a tool that will guarantee great liberty of action, thanks to shared-control. This prospective gives confidence that this study is only the start for series of other future accomplishments that would lead in medical environment to great innovations. Neuro-muscular pathologies that now make the patient's live condition really challenging and exasperating, they will be treated in a future with simple wearable devices. Patient in locked-in state will be able to interact with the world thanks to robotic manipulators. New rehabilitation strategy and protocol will be developed. Essentially the applications of this study are limitless.

Even if the obtained results are far from being defined "perfect", what is actually important is the concept and the conclusion that underlie them: it is really possible control a machine with physiological signals.

Acronyms

ACH Acetylcholine. 5, 6, 45

BMI Brain-Machine Interface. 2, 45

DFT Discrete Fourier Transform. 11, 45

DH Denavit-Hartenberg. 17, 18, 45

DOF Degrees of Freedom. 3, 13, 14, 16, 45

DWT Descrete Wavelet Transform. 45

ECG Electrocardiography. 45

EEG Electroencefalography. 2, 3, 23, 24, 45

EMG Electromiography. 2–4, 6–9, 11, 12, 21, 23, 24, 28, 29, 32, 33, 35–37, 42, 45, 51, 53

EPP End Plate Potential. 6, 45

fEMG fine-wire Electromiography. 6, 45

FFT Fast Fourier Transform. 11, 45, 54, 55

IAS-Lab Intelligent Autonomous System Laboratory. 45

LOO Leave-One-Out. 30, 45

MLR Multivariate Linear Regression. 21, 35–39, 41, 45, 57

MUAP Motor Unit Action Potential. 7, 12, 32, 45

NMSE Normalized Mean Square Error. 38, 40, 41, 45

PCA Principal Component Analysis. 36, 37, 45, 57

ROM Range of Motion. 14, 15, 45

ROS Robotic Operating System. 4, 19, 20, 37, 42, 43, 45, 57

SDK Software Development Kit. 19, 45

sEMG surface Electromiography. 7–9, 12, 24, 27, 35, 45, 53–55

STFT Short-time Fourier Transform. 12, 45

SVM Support Vector Machine. 3, 21, 29, 30, 32, 45, 51, 53

WT Wavelet Transform. 12, 45

ZCR Zero-Crossing Rate. 27–29, 31, 32, 45, 54

Appendices

Chapter A

Target-oriented classification code

A.1 Matlab main

The following code displays the program main for target-oriented classification, comprehensive of signal extraction from data set files, filtering and EMG rectification, parameter extraction, SVM modeling and error prediction calculation.

```
1
%% Data loading and store variables
3 disp('[user] - Loading data');

5 [emg, pos, vel, TargetLab, TrialId, LTrials] = ...
                                  offlineDataExtraction(path, limb);
7
%% General settings
9 disp('[user] - General settings');
EmgNumChannels  = size(emg, 2);
11 EmgNumSamples   = size(emg, 1);
EmgChanLabels   = {'CH1', 'CH2', 'CH3', 'CH4', 'CH5', 'CH6', 'CH7', ...
13                    'CH8', 'CH9', 'CH10', 'CH11', 'CH12', 'CH13', ...
                      'CH14', 'CH15', 'CH16'};
15 EmgChanId       = 1:EmgNumChannels;
EmgChanLabels   = EmgChanLabels(EmgChanId);
17 KinNumVariables = size(pos, 2);
KinLbVariables  = {'x', 'y', 'z'};
19 NumTrials       = length(LTrials);
NumTargets      = 4;
21 param           = {'s_mean', 'IEMG', 'ZCR', 'in_percent', 'mean_peak', ...
    'f_mean', 'f_median'};
23 NumParam        = length(param);

25 % Taking off from analysis the corrupted channels
EmgExChanId     = [8 16];
27 EmgChanId       = setdiff(EmgChanId, EmgExChanId);
EmgChanLabels   = EmgChanLabels(EmgChanId);
29 emg = emg(:, EmgChanId);

31 %% Emg signal rectification
% The emg acquired signal is rectified before bulding up the model
```

```
% according litterature the parameter of the filters are set up

disp('[proc] - Computing rectification of the emg signals');
s = emg_smoothing(proc_rectification(emg, 7, 4, 200, 1000),20);

%% Amplitude Parameters Extraction
% Extraction of the most significative EMG signal parameters related to
% amplitude for each channel.

disp('[proc] - Extraction of amplitude related parameters');
[s_mean, IEMG, ZCR, in_percent, mean_peak] = emg_amparam(emg, s, TrialId);

%% Frequence Parameters Extraction
% Extraction of the most significative EMG signal parameters related to
% frequency domani for each channel.

disp('[proc] - Extraction of frequency related parameters');
[f_mean, f_median] = emg_fparam(emg, TrialId);

%% Multi-class SVM prediction with cross-convalidation Leave-One-Out

% Definition of the group test: it conteins the target label of each trial
for idTrial = 1:NumTrials
    tmp = TargetLab(TrialId == idTrial);
    tLab(idTrial,1) = tmp(1);
end

% Creation of NumTrials different partition of the trial sets acording with
% the leave-one-out criterion
LO_partitons = cvpartition(NumTrials,'LeaveOut');

% Run of the multi-class support vector machine

disp(['[proc] - multi-class SVM prediction with ' ...
                        'cross-convalidation Leave-One-Out']);
for idParam = 1:NumParam
    disp(['         ...for ' cell2mat(param(idParam)) ' parameter']);
    for idTrial = 1:NumTrials
        ind_training=training(LO_partitons,idTrial);
        ind_test=test(LO_partitons,idTrial);
        eval(['result_class(ind_test,idParam) = multisvm(' ...
            cell2mat(param(idParam)) '(ind_training,:),' ...
            'tLab(ind_training),' cell2mat(param(idParam)) ...
            '(ind_test,:));']);
    end
end

% Calculation of the error class prediction
disp('[proc] - Error prediction computation');
check_class = zeros(NumTrials, length(param));

% check_class variable represents a table containing the missmatch between
% the predicted class and the real class
for idParam = 1:NumParam
    for idTrial = 1:NumTrials
        if result_class(idTrial,idParam) ~= tLab(idTrial)
            check_class(idTrial,idParam) = 1;
        end
    end
end

% total number of error prediction over all the cross-convalidation
```

```
95 % datasets divided by parameters
   NumErrPred = sum(check_class,1);
97
   % total error prediction frequency estimation over all the
99 % cross-convalidation datasets divided by parameters
   fErrPred = NumErrPred/NumTrials;
101
   % estimation of the real error frequency range with 95% confidence
103 % interval, fitting the error prediction with a binomial distribution
   for idParam = 1:NumParam
105     pd = fitdist(check_class(:,idParam),'Binomial');
        int_tmp = paramci(pd);
107     intConf(1:2,idParam) = abs(int_tmp(:,2)-fErrPred(idParam));
   end
109
   % analysis of the total error prediction frequency related to the target
111 for idTarget = 1:NumTargets
        ind = tLab == idTarget;
113     errTarget(idTarget,1:NumParam) = sum(check_class(ind,:),1)./NumErrPred;
   end
```

Listing A.1: Main section of the target-oriented classification code

A.2 Parameters extraction

This appendix section defines the parameters used to feed the SVM model and how they are extracted. In the following formulas, the variable s identifies the rectified signal. The letter i represents the pointer to the acquisition sample (N is the length of the trial acquisition vector), the letter j instead stands for the identification number of the channel.

- ***s_mean*** is defined as the mean value of the EMG rectified signal of a single channel. On line 38 in Listing A.2 the value is computed simply calling out the matlab function *mean()*.

$$s\_mean_j = \frac{1}{N}\sum_{i=1}^{N} s_j(i) \quad i,j \in \mathbb{N} \tag{A.1}$$

- ***IEMG*** represents the integral under the curve of a rectified sEMG channel acquisition. Since the signal after the pre-processing is converted in positive values there is no integration problem. In Matlab code is used Trapezoidal Numerical Integration (Listing A.2, line 41).

$$IEMG_j = \frac{t(N)-t(1)}{2N}\sum_{i=1}^{N-1}[s_j(i)+s_j(i+1)] \quad i,j \in \mathbb{N} \tag{A.2}$$

- ***ZCR*** is simply a counter of the sEMG zero-crossing. The difficult part is to get rid of the noisy crossing. For this particular parameter the same procedure of Chang et al.[5] is kept. First, on the raw sEMG a base-line correction is done so that the signal has null mean; second step is to set up a control able to identify the zero-crossing only related to the muscle contraction. This was done selecting overshoots and undershoot in the signal (using *findpeaks()* function) and counting only the consecutive pair which respects threshold condition and positive-negative value reversal (Listing A.2, line 48-56).

- ***in_percent*** wants to express which channel is more active than the others. It is possible to see each channel as just a portion of the total information carried out from the sEMG: basically each channel gives a percent contribution to the totality of the acquisition. The channel activation is represented by the mean value of its recording; the mean values are then normalized with the value obtained from their summation (Listing A.2, line 39,45). The sum of *in_percent* along j return 100%.

$$in\_percent_j = \frac{s\_mean_j}{s\_100\%} \tag{A.3}$$

$$where\ s\_100\% = \sum_{j=1}^{M} s\_mean_j \quad j \in \mathbb{N} \tag{A.4}$$

- ***mean_peak*** is derived from the mean of the highest spikes value for each of the sEMG channel. The Matlab function *findpeaks()* was set up in way to get the 20 highest peak values (Listing A.2, line 42-44).

After the FFT the signal is mapped from time domain to a frequencies preserving the same number of samples. The Furier Transform of a real signal is however a periodic signal of 2π and symmetric to the π axis: which is why only the first half of the domain is kept for the analysis. The FFT of the signal in the following formulas is represented from the parameter $\mathcal{S}$ and the letter k is the pointer to the frequency components, included between 1 and $N/2$.

- ***f_mean*** is the mean frequency content of the signal. (Listing A.3, line 31).

$$f\_mean_j = \frac{\sum_{k=1}^{N/2} f(k) \cdot \mathcal{S}(k)}{\sum_{k=1}^{N/2} \mathcal{S}(k)} \quad k \in \mathbb{N} \tag{A.5}$$

- ***f_median*** represents the median frequency of the signal spectrum. (Listing A.3, line 32-37).

```
1
function [s_mean, IEMG, ZCR, in_percent, mean_peak] = ...
3                                                   emg_amparam(emg, s, TrialId)
    % The function emg_amparam is designed for the extraction of the more
5   % important and relevant parameters, according with litterature, for
    % the caraterization of emg signals in amplitude domain.
7   % INPUT
    %       - emg        : raw emg signal matrix; columns contain single
9   %                      channel acquisition, rows contain the ith sample
    %                      for each channel
11  %       - s          : pre-processed emg signal acquisition matrix
    %       - TrialId    : vector containing the trial label for each emg
13  %                      sample
    % OUTPUT
15  %       - s_mean     : s signal matrix mean for each channel
    %                      acquisition
17  %       - IEMG       : s signal matrix integration for each channel
    %                      acquisition
19  %       - ZCR        : Zero-crossing counter of the emg signal matrix
    %                      for each channel acquisition
21  %       - in_percent : percentual input of each channel acquisition in
    %                      s signal matrix
23  %       - mean_peak  : mean of the more relevant peak in s signal
    %                      matrix for each channel acquisition
25  % Each output parameter is a matrix which contains along rows the
    % parameter of each channel and along columns the realisation of each
27  % trial.

29  % Initial settings
    ZCR = zeros(TrialId(end), size(s,2));
31  s_mean = zeros(TrialId(end), size(s,2));

33  for idTrial = 1 : TrialId(end)
        ind = TrialId == idTrial;
35      cemg = emg(ind,:);
        cs = s(ind,:);
37      s_mean(idTrial,:) = mean(cs, 1);
        emg_mean = mean(cemg,1);
39      s_100perc = sum(cs, 2);
        for idCh = 1 : size(s,2)
41          IEMG(idTrial, idCh) = trapz((1:size(cs,1))/1000, cs(:,idCh));
            [pks pkpos] = findpeaks(cs(:,idCh), 'npeaks',20,'sortstr', ...
43                                                          'descend');
            mean_peak(idTrial, idCh) = mean(pks);
45          in_percent(idTrial, idCh) = mean(cs(:,idCh)./s_100perc)*100;
            tmp=cemg(:,idCh)-emg_mean(idCh);
47          tmp = spline_smth(tmp);
            [over over_pos] = findpeaks(tmp, 'MinPeakDistance', 10);
49          [under under_pos] = findpeaks(-tmp, 'MinPeakDistance', 10);
            pks_pos = sort([over_pos; under_pos]);
51          for idZcr = 1: length(pks_pos)-1
                if abs(tmp(pks_pos(idZcr))-tmp(pks_pos(idZcr+1)))>0.008 ...
53                      && tmp(pks_pos(idZcr))*tmp(pks_pos(idZcr+1))<0
                    ZCR(idTrial, idCh) = ZCR(idTrial, idCh)+1;
55              end
            end
57
```

```
        end
    end
end
```

Listing A.2: *emg_amparam* function: used for the extraction of time-domain related parameters

```
function [f_mean, f_median]=emg_fparam(emg, TrialId)

    % The function emg_fparam is designed for the extraction of the more
    % important and relevant parameters, according with litterature, for
    % the caraterization of emg signals in frequency domain.
    % INPUT
    %       - emg        : raw emg signal matrix; columns contain single
    %                      channel acquisition, rows contain the ith sample
    %                      for each channel
    %       - TrialId    : vector containing the trial label for each emg
    %                      sample
    % OUTPUT
    %       - f_mean     : emg signal matrix frequency mean for each
    %                      channel acquisition
    %       - f_median   : emg signal matrix frequency median for each
    %                      channel acquisition
    % Each output parameter is a matrix which contains along rows the
    % parameter of each channel and along columns the realisation of each
    % trial.

for idTrial = 1 : TrialId(end)
    ind = TrialId == idTrial;
    cemg = emg(ind,:);
    for idCh = 1 : size(emg,2)
        hp_emg = filt_highlow(cemg(:,idCh), 7, 4, 1000, 'high');
        % Fourier fast transform: frequency domain shift
        fft_tmp = abs(fft(hp_emg));
        fft_emg = fft_tmp(1:ceil(length(fft_tmp)/2));
        freq = linspace(1/1000,500,size(fft_emg,1))';
        f_mean(idTrial, idCh) = sum(freq.*fft_emg)/(sum(fft_emg));
        median_threshold=sum(fft_emg)/2;
        ind_median = 2;
        while sum(fft_emg(1:ind_median))<median_threshold
            f_median(idTrial, idCh)=freq(ind_median);
            ind_median=ind_median+1;
        end
    end
end
```

Listing A.3: *emg_fparam* function: used for the extraction of frequency-domain related parameters

Chapter B

Kinematic reconstruction analysis code

B.1 Matlab main

The following code displays the program main for the kinematic prediction. MLR modeling, PCA data reduction and Kalman filtering are included in the following code (Listing B.1). The Matlab code presents, as described in Chapter 4, two primary section: off-line training and on-line simulation.

The code contains, moreover, the implementation of the Matlab Publisher used to communicate with the ROS network and send the messages to NAO robot.

```
%% Load initialized Data-base
disp('[user] - Loading Reshaped DataBase');
load('OnlineDB.mat');
addpath(genpath(path));

%% Section for initialization the comunication with NAO robot

send2Nao = true;

if (send2Nao)
    % Human anthropometry initialization

    global shoulderOffsetX
    global shoulderOffsetY
    global elbowOffsetY
    global upperArmLength
    global shoulderOffsetZ
    global LowerArmLength
    global HandOffsetX
    global HandOffsetZ

    % INITIALIZATION ROS PUBBLISHER

```

```
    % add the ipc_bridge_matlab binaries to your path
    [a, p] = system('rospack find ipc_geometry_msgs');
    addpath(strcat(p, '/bin'));

    % create a publisher that publishes a geometry_msgs/Twist message
    pid=geometry_msgs_Twist('connect','publisher','matlab_module','twist');

    % create an empty geometry_msgs/Twist message structure
    msg=geometry_msgs_Twist('empty');

    % Starting free-coordinates initialization

    thetaST(1) = 1.2;   % Shoulder pitch: with negative values --->
                                                    % forward rotation
    thetaST(2) = 0.40;  % Shoulder roll: with positive values  --->
                                                    % abdution
    thetaST(3) = -1.13; % Elbow yaw: with negative values      --->
                                                    % intrarotation
    thetaST(4) = -1.40; % Elbow pitch: with negative values    --->
                                                    % flexion
    thetaST(5) = 1.00;  % Wrist yaw: wth positive values      --->
                                                    % intrarotation
    % only one trial is tested in NAO robot
    NumTest = 2;
else
    NumTest = 10;
end

% initializazion of result variable
gdf = zeros(NumTest, KinNumVariables);

for h=1:NumTest
    if send2Nao
        % Set Nao to the starting position
        initNao(thetaST);
        rosPublisher(pid,msg,thetaST);
        theta = thetaST';
    end

    disp(['         ONLINE SIMULATION N' num2str(h) ' :']);
    Ep = s(TrainIndex(:,h), :);
    Et = s(TestIndex(:,h), :);
    Pp = pos(TrainIndex(:,h), :);
    Pt = pos(TestIndex(:,h), :);
    Vp = vel(TrainIndex(:,h), :);
    Vt = vel(TestIndex(:,h), :);

    % PCA and LRM
    % Computing PCA on the emg signals and from these results building up
    % the linear regression model for predicting the position and velocity
    % acquisition

    PCAExplained = 96;

    % The following function gives back position and velocity regression
    % model and PCA parameters

    Kp = [Pp, Vp];
    [coeff, PCACmpId, M] = emgModelRegress(Ep, Kp, PCAExplained);

    % ONLINE SIMULATION
    % The online simulation is run on the test set consisting of a single
```

```
% trail. Since the online simulation has to be accomplished in
% real-time, the test trial is segmented in a overlapped buffer series;
% for each buffer is computed:
% - rectification;
% - PCA decomposition;
% - buffer mean calculation;
% - Multivariate linear regression model for the mean vector;
% - Kalman filtering of the regression pre1diction.


testEmg = emg(TestIndex(:,h), :);
Fs = 1000;
FsB = 32;        % Sample distant between buffers

BufferSize = 512;
FrameShift = FsB;
testEmg = [testEmg; repmat(testEmg(end,:),BufferSize,1)];
StartPos = 1:FrameShift:length(testEmg) - BufferSize;
StopPos = StartPos + BufferSize - 1;

Pos = zeros(length(StartPos), KinNumVariables);
Vel = zeros(length(StartPos), KinNumVariables);
KPos = zeros(length(StartPos), KinNumVariables);
KVel = zeros(length(StartPos), KinNumVariables);

% Kalman filter matrixes creation
% Auxiliary matrix
A = eye(3);
B = FsB/Fs*A;
O = zeros(3);
% Model matrix
F = [A,B;
     O,A];
% Measurement matrix
H = [A,O;
     O,A];

% Covariance Matrix
kappa = 2.9;             % Modulation value

%R = eye(6);                  % Measure covariance matrix
tmp = Ep*coeff;
tmp1 = [ones(size(tmp(:,PCACmpId),1),1),tmp(:,PCACmpId)]*M-Kp;
R = cov(tmp1);
Q = kappa*eye(6);           % Model covariance matrix

% Initial condition setting
Pstart = F*Q*F';            % Intial error extimation covariance matrix
Xstart = [Pt(1,:),Vt(1,:)]'; % Initial kinematic values extimation

% BUFFER CYCLING
pnew = Pstart;
xnew = Xstart;

display('[proc] - Online Simulation (Buffer cycling)');
tic;
for sId = 2:length(StartPos)
    % Buffer initialization
    cstart = StartPos(sId);
    cstop  = StopPos(sId);
    cbuff = testEmg(cstart:cstop, :);
    % Buffer rectification
```

```
        crect = proc_rectification(cbuff, 7, 200, 4, 1000);
        % Buffer PCA reduction
        ctmp  = crect*coeff;
        cpca  = ctmp(:, PCACmpId);
        % Buffer mean
        cval = mean(cpca, 1);
        % Kinematic values prediction
        cpred = [ones(size(cval,1),1),cval]*M;
        cpos = cpred(:,1:size(KinLbVariables,2));
        cvel = cpred(:, (1:size(KinLbVariables,2))+size(KinLbVariables,2));
        % KALMAN FILTERING
        [ckin, pnew, xnew] = fdk([cpos cvel]', pnew, xnew, F, H, Q, R);
        Pos(sId, :) = cpos;
        Vel(sId, :) = cvel;
        KPos(sId, :) = ckin(1:size(KinLbVariables,2));
        KVel(sId, :) = ckin((1:size(KinLbVariables,2)) + ...
                                                size(KinLbVariables,2));
        if send2Nao
            theta = ToNao(KPos(sId,:),theta);
            rosPublisher(pid,msg,theta);
        end
    end
    tSim = toc;


    disp('[proc] - RESULTS: performance of the prediction algorithm');
    for vId = 1:KinNumVariables
        gdf(h,vId) = goodnessOfFit(KPos(:, vId), Pt(StartPos, vId), ...
                                   'NMSE');
        disp(['          Goodness of fit for ' KinLbVariables{vId} ...
                    ' position [-inf 1]: ' num2str(gdf(h,vId))]);
    end
    disp(['          Simulation runing time: ' num2str(tSim) 's vs. ' ...
                   'real test time ' num2str(size(testEmg,1)/1000) 's'])
end
```

Listing B.1: Main section of the kinematic reconstraction code

B.2 Kalman Filter

This section of the appendix is dedicated to Kalman filtering. Here it is explained the formulation of the Kalman filter and the implementation in the Matlab code.

The idea underlying the Kalman filter is to think at filtering as a problem of system state estimation. In this perspective our measurements can be thought of as noisy output from a certain system. If the model of the system is known, it is possible to compare the noisy measurements with the output of this model in order to estimate the better the best solution.

$$\begin{cases} x(t+1) = Fx(t) + w(t) \\ y(t) = Hx(t) + v(t) \end{cases} \tag{B.1}$$

In the system, the first line embodies the *Model equation* the second is called instead the *Measure equation*. x is the state vector, and represents what actually we want to estimate, y is the vector of the measurements, F is the matrix of the transition state and H is the input-output matrix. w and v in the end represent respectively the error of model and the noise we want to get rid of on the measurements data. In the equation system, w and v are aleatory vectors between them uncorrelated, with covariance matrix $cov(w) = Q$ and $cov(v) = R$.

The Kalman filter is designed to operate recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state. Two equations define the filter: the update state equation and the measures equation. The algorithm works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, exploiting the transition state equation, along with their uncertainties. Where the estimation and the error prediction are have the following representation:

$$\begin{aligned} \hat{x}(t+1|t) &= F\hat{x}(t|t) \\ P(t+1|t) &= FP(t)F^T + Q \end{aligned} \tag{B.2}$$

What it is estimate here represents the prior knowledge of the next event at $t+1$, independently from what is going to actually happen. Once the outcome of the next measurements, corrupted by certain noise, is observed, these estimates are updated refining the estimation of the new state. The obtain formulation is the following:

$$\begin{aligned} \hat{x}(t+1|t+1) &= \hat{X}(t+1|t) + P(t+1|t) \\ &\qquad H^T[HP(t+1|t)H^T + R]^{-1}[y(t+1) - H\hat{x}(t+1|t)] \\ P(t+1) &= P(t+1|t)H^T[HP(t+1|t)H^T + R]^{-1}HP(t+1|t) \end{aligned} \tag{B.3}$$

where $x(t+1|t+1)$ is the finally estimation of the system state and $P(t+1)$ is its uncertainty. Listing B.2 displays the code of Kalman filter predictive filter; definition of the input matrices are instead in Listing B.1 from between lines 112 ans 139.

```
2  function [y,Pnew,Xnew]=fdk(ys,Pold,Xold,F,H,Q,R)

4  %-------------------------------------------------------------------------%
   % KALMAN FIELTER %
6  %-------------------------------------------------------------------------%
   % ### INPUT ###
8  % ys = value of the istant measurment
   % Pold = error extimation covariance matrix at the prior iteration
```

```
% Xold = state vector at the prior iteration
% F = state transition matrix
% H = Input/Output trasfer matrix
% Q = model error covariance matrix
% R = measure error covariance matrix
% ### OUTPUT ###
% y = ys filtered value at the current iteration
% Pnew = error extimation covariance matrix at the current iteration
% Xnew = state vector at the current iteration
%--------------------------------------------------------------------------%

% predictive step of Kalman filter
xpred=F*Xold; % calculation of x(t|t-1)
Ppred=F*Pold*F'+Q; % calculation of P(t|t-1)
K=Ppred*H'*inv(H*Ppred*H'+R); % calculation of the filter gain currection

% correction step using ys measurments
e=ys-H*xpred; % prediction error
Xnew=xpred+K*e; % calculation of x(t|t)
I=eye(size(Q));
Pnew=(I-K*H)*Ppred; % calculation of P(t|t)

% calculation of the filtered value
y=H*Xnew;
```

Listing B.2: *fdk* function: it implements the Kalman filtering. The matrices used by the model are passed to the algorithm as input variables along with the measure

Bibliography

[1] Jeremy Norman's HistoryofInformation.com, "The first industrial robot (1954 – 1961)," May 2015.

[2] R. Bortoletto, "Simulating a Flexible Robotic System based on Musculoskeletal Modeling," Master's thesis, Università degli Studi di Padova, 2000.

[3] Johns Hopkins University, Applied Phisics Laboratory, "Amputee makes history with apl's modular prosthetic limb," Dec. 2014.

[4] da Vinci Surgery, "The da vinci®surgery experience," 2014.

[5] G.-C. Chang, W.-J. Kang, J.-J. Luh, C.-K. Cheng, J.-S. Lai, J.-J. Chen, and T.-S. Kuo, "Real-time implementation of electromyogram pattern recognition as a control command of man-machine interface," *Medical Engineering & Physics*, 1995.

[6] M. B. I. Reaz, M. S. Hussian, and F. Mohd-Yasin, "Emg analysis using wavelet functions to determine muscle contraction," *e-Health Networking, Applications and Services*, 2006.

[7] K. Englehart, B. S. Hudgins, P. A. Parker, and M. Stevenson, "Classification of the myoelectric signal using time-frequency based representations," *Medical Engineering & Physics*, Feb. 1999.

[8] B. S. Hudgins, "A new strategy for multifunction myoelectric control," *IEEE Trans Biomedical Engineering*, 1993.

[9] BackYardBrains, "Experiment: Getting started with the emg spikerbox."

[10] W. Durfee, *Neuromuscular Junction*. University of Minnesota, 2011.

[11] B. S. Rajaratnam, J. C. Goh, and V. P. Kumar, "A comparison of emg signals from surface and fine-wire electrodes during shoulder abduction," *International Journal of Physical Medicine & Rehabilitation*, June 2014.

[12] J. V. Basmajian and C. J. De Luca, *Muscles Alive*. Baltimore: Williams Wilkins, 1985.

[13] P. Konrad, *The ABC of EMG*, 1.4 ed., Mar. 2006.

[14] S. Kumar and A. Mital, *Electromyography in Ergonomics*. London: Taylor&Francis, 1996.

[15] F. Sandbrink, "Motor unit recruitment in emg definition of motor unit recruitment and overview," May 2014.

[16] J. R. Cram and G. Kasman, *Introduction to Surface Electromyography*. Gaithersburg: Aspen Publishers, 1998.

[17] L. Vodovnik, J. Kreifeldt, R. Caldwell, L. Green, E. Silgails, and P. Craig, "Some topics on myoelectric control of orthotic/prostetic systems," *Case Western Reserve Univerisity*, 1967.

[18] C. Wei-ting, W. Zhi-zhong, and R. Xiao-mei, "Characterization of surface emg signals using improved approximate entropy," *Journal of Zhejiang University SCIENCE B*, Apr. 2006.

[19] F. Johannßen, "Nao robots in the cloud - an interface to transform and execute abstract plans," Master's thesis, Hamburg University of Applied Sciences, Faculty of Engineering and Computer Science, 2014.

[20] N. Kofinas, "Forward and inverse kinematics for the nao humanoid robot," Master's thesis, Technical University of Crete, Department of Electronic and Computer Engineering, 2012.

[21] J. M. Soucie, C. Wang, A. Forsyth, S. Funk, M. Denney, K. E. Roach, D. Boone, and the Hemophilia Treatment Center Network, "Range of motion measurements: reference values and a database for comparison studies," *Haemophilia*, Nov. 2010.

[22] N. Kofinas, E. Orfanoudakis, and M. G. Lagoudakis, "Complete analytical inverse kinematics for nao," in *Autonomous Robot Systems (Robotica), 2013 13th International Conference on*, Intelligent Systems Laboratory, Department of Electronic and Computer Engineering Technical University of Crete, April 2013.

[23] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans ASME*, 1995.

[24] F. Mondada, "Ros interface and urdf parser for webots," Master's thesis, École polytechnique fédérale de Lausanne, 2014.

[25] S. Butterworth, "On the theory of filters amplifiers," *Experimental wireless & the wireless engineer*, 1930.

[26] C. Neuburger, "Multi class svm."

[27] L. Tonin, R. Leeb, M. Tavella, S. Perdikis, and J. R. del Millan, "The role of shared-control in bci-based telepresence," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, Center for Neuroprosthetics, Ecole Polytecnique Fed. de Lausanne, October 2010.

[28] D. J. McFarland and J. R. Wolpaw, "Brain-computer interface operation of robotic and prosthetic devices," *Computer Journal Impact Factor & Information*, 2008.

Dedicated to my mother, who made me a believer.
To my father, who turned me into a fighter.
To my sisters, who taught to me what means to grow up.
To my friends, who constantly remind me what means to be young.