Università degli Studi di Padova

Department of Information Engineering

Master Degree in Computer Engineering

Processing of sEMG signals for online motion of a single robot joint through GMM modelization

Advisor:	Prof. Enrico Pagello
Coadvisor:	Dr. Stefano Michieletto

Student: Riccardo Valentini

20th of April 2015 Academic Year 2014/2015

At bottom, robotics is about us.
It is the discipline of emulating our lives,
of wondering how we work.

Rod Grupen

Sommario

Lo scopo di questa tesi è di esplorare la possibilità di utilizzare i segnali elettromiografici (EMG) per allenare un modello di probabilità a componenti gaussiane (Gaussian Mixture Model) che riesca a stimare in tempo reale l'angolo di curvatura di un singolo giunto del corpo umano.

Le informazioni del segnale sono state estratte utlizzando due approcci: il primo facente uso di trasformate nel dominio del tempo e nel dominio della frequenza, mentre il secondo utilizza la trasformata wavelet di cui è stata approfondita la migliore configurazione.

Le funzioni wavelet di Daubechies Db2, Db4, Db6, Db7, Db44, Db45 sono state applicate ai segnali EMG corrispondenti a diversi muscoli della gamba a cui è seguita una aggregazione statistica dei dati da usare come input per la creazione del modello.

Il modello GMM è stato validato su dati sconosciuti e le prestazioni sono state confrontate in base alla funzioni mother wavelet scelte per il calcolo della trasformata ed al numero di ripetizioni usate durante la fase di allenamento.

La procedura migliore è stata utilizzata come base per sviluppare un software scritto in linguaggio C++ in grado di interfacciarsi con un robot umanoide a cui è stato fatto eseguire il movimento corrispondente a quello di una persona. Il software è stato sviluppato sotto forma di modulo ROS separando le varie fasi di elaborazione: generazione del segnale, acquisizione del segnale, analisi nel tempo e nella frequenza, sottomissione al modello e comunicazione del movimento al robot. Ogni fase è stata verificata indipendentemente dalle altre in modo da individuare la configurazione migliore per ciascuna. Il sistema è stato testato su un robot modello Aldebaran NAO.

I risultati ottenuti mostrano come la procedura sia in grado di ottenere alte prestazioni (Normalized Mean Square Error: 0.91, 0.90, 0.81 per i tre soggetti in esame rispettivamente), con l'utilizzo della funzione wavelet Db2 della famiglia Daubechies assieme al calcolo MAV (Media dei Valori Assoluti) dei coefficienti del primo livello di decomposizione della trasformata. I tempi di elaborazione delle varie fasi (500 μS per la trasformata con Db2, 0.845 μS per il calcolo del MAV e 1774 μS per la risposta del modello) risultano compatibili con il controllo in tempo reale di robot umanoidi o esoscheletri.

Abstract

This thesis aims to explore the possibility of using Electromyography to train a Gaussian Mixture Model in order to estimate in real-time the bending angle of a single human joint.

Extraction of features is performed trough two approaches: the first uses of time-domain or frequency-domain transforms while the second makes use of wavelet transform by investigating which configurations perform best when combined with Gaussian Mixture Model (GMM)s.

Daubechies wavelet functions *Db2*, *Db4*, *Db6*, *Db7*, *Db44*, *Db45* are applied to Electromyography (EMG) signals of lower limb muscles and a further statistical aggregation is computed on resulting coefficients. Features obtained are used as input for system modelization.

GMM is validated on new unseen data and the classification performances are compared with respect to the *mother wavelet* chosen for computing wavelet decomposition and the number of collected trials used during the training phase.

Best procedure is used to develop C++ software capable of interfacing with a humanoid robot, which was made to perform a movement corresponding to that of the person. For each phase of the process a different Robot Operating System (ROS) module is developed. A module executes signal generation while others perform signal acquisition, computation of wavelet transform, GMM regression and communication to robot. Every step is independently tested as to detect for each one the configuration which better performs. An Aldebaran NAO robot is used for motion testing.

Achieved results showed that our procedure is able to obtain high performances (Normalized Mean Square Error: 0.91, 0.90, 0.81 for the three examined subjects, respectively), using Daubechies *Db2* wavelet function together with Mean Average Value calculation of first level wavelet decomposition coefficients. Processing times (500 μS for Db2 transform, 0.845 μS for Mean Average Value (MAV) computation and 1774 μS for GMM regression) ensure compatibility with an online use in controlling humanoid robots or exoskeletons.

Contents

List of Tables

Acronyms

AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
CoM	Center of Mass
CWT	Continuous Wavelet Transform
DoF	Degree of Freedom
DoFs	Degrees of Freedom
DWT	Discrete Wavelet Transform
EEG	Electroencephalogram
EM	Expectation-Maximization
EMD	Electro Mechanical Delay
EMG	Electromyography
FFT	Fast Fourier Transform
FNN	Feedforward Neural Network
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
HMM	Hidden Markov Model
HillMM	Hill Muscle Model
HPR	Hierarchical Projected Regression
IAV	Integral Absolute Value
MAV	Mean Average Value

MDL	Minimum Description Length
MML	Minimum Message Length
MoG	Mixture of Gaussians
MPF	Mean Power Frequency
MU	Motor Unit
NASA	National Aeronautics and Space Administration
NMSE	Normalized Mean Square Error
NN	Neural Networks
RLfD	Robot Learning from Demonstration
RMS	Rooted Mean Square
ROS	Robot Operating System
sEMG	Surface Electromyography
YAML	YAML Ain't Markup Language

Chapter 1

Introduction

1.1 Overview on Electromyography

The signal generated from the skeletal muscle activity is called EMG. The functional unit of muscle is the Motor Unit (MU), which consists of an alpha motor neuron and all fibers innervated by it. Muscle fibers are the structural unit of contraction [27].

The number of fibers per motor unit varies by the kind of movement for which it is intended. EMG signal derives from potentials generated through MU activations. It could be extracted by mean of needle electrodes inserted directly in the muscle or in a less invasively way using surface electrodes. The first method ensures a more accurately signal recording, carrying out electrical activity closer to its muscle source. sEMG is instead a result of the summation of nonsynchronous action potentials of a large number of muscle fibers that have been nonlinearly attenuated by body tissue due to the frequency dependent electrical properties of the tissues [26].

sEMG acquisition is done using electrodes that do not exceeds the surface of the skin. In base of the position on the body a different signal is retrieved, that could be affected more or less by different and overlapping muscle activities. Although lower quality of these signals they carry out enough information to be used in a wide variety of applications. Moreover their systems of acquisition are cheaper and so more affordable.

1.2 Applications of sEMG

sEMG signals are widely used in medicine, clinical analysis and diagnostics, and for controlling prosthetic devices. Thanks to advanced techniques is possible to extract from sEMG necessary information able to determine a description of the muscular activity or detect movement diseases.
At the National Aeronautics and Space Administration (NASA), Ames Research Centre has developed a flight control system based on recognition of gestures by mean of sEMG signals. Once retrieved, sEMG signals are classified and then used

as input for neuroelectric joysticks and keyboards. Similar approaches are used in videogames industry to provide a more exciting gaming experience.
In the robotic field many researchers used classification of EMG signals and their corresponding gestures to control prosthesis or exoskeletons. Relevant researches have been done for the upper limb body and in particular for electromechanical hands control [54] [51] [47].

1.3 Extraction of information

Majority of studies in literature proposes different ways identify a user gesture (like closing, flexioning, extensioning and opening hand) by classifying groups of signals ([32], [50]). The objective is perform the corresponding complete movement by means of a prosthesis. These approaches do not estimate the position of every single joint of the hand (about 23 degrees of freedom including the wrist), which is very challenging, what is considered is the entire group of features, from which recognizing the predefined movement. In the hand, the proximity of muscles causes overlapping and interwined signals, for this reason it is easier to recognize the entire gesture instead of identifying every single joint in the tangle of EMGs. The approach to macro movements works well if you move prosthetic devices completely separated from the subject. In the case of exoskeletons, where the movement of the robotic apparatus should be as coordinated as possible with that of the human body, it is important to have specific information for each individual joint; specially, in cases like lower limbs, where joints are distant from each other.

EMG signals are not easy to decode because of, large variance in power spectrum and a random variation in the effectiveness of the body tissues in conducting the signal from the muscle to the surface electrodes. Moreover, there is a time dependency due to MU force production and the actual load, as well as fatigue status of the muscle itself. The features listed above make it necessary to analyze sEMG both in time and frequency domains since ignoring one or the other would gather incomplete or not sufficient information.

Some widespread techniques adopted for pattern recognition are Fourier transform [15], Integral Absolute Value (IAV), variance and zero crossing [30], MAV [31], RMS, Mean Power Frequency (MPF) [28], or as proposed by Sartori et al. [44] full wave rectification, filtering and normalisation. The major drawback of these transformation methods, especially fast and short-term Fourier transform, is that they assume the stationarity of the signal [24].

However, EMG signals are non-stationary, so an alternative approach, based on wavelet transform, is successfully spreading. Through wavelet transform, Daubechies [16] fulfilled analysis of time series that contained non-stationary power at many different frequencies. Early as 1997, Laterza and Olmo [29] found out that wavelet transform is an alternative to other time frequency representations with the advantage of being linear, yielding a multiresolution representation unaffected by crossterms. Guglielminotti and Merletti [21] theorized that wavelet transform exhibits very good energy localization in the time-scale plane when the shape of the EMG signal is matched with wavelet shape. Recent works of Chowdhury et al. [12], Sahin and Sahin [43], and Phinyomark et al. [39] reinforced advantages in using wavelet transform for EMG analysis. Chowdhury et al. [12] concluded that analyzing sEMG signals using Daubechies function returns successful results by investigating and analyzing various research studies on wavelet transform. Some statistical measures are often applied in order to synthesize wavelet information, like exposed by Subasi in [49] and [48].

Significant in EMG analysis is the number of channels considered to predict movement. An increasing number of channel obviously increases the classification's efficieny, but the result obtained by Tsenov et al. [50] using four channels suggest that a small number of channels may be sufficient.

Several studies used features extracted from EMG to feed different machine learning algorithms (i.e., Neural Networks (NN), GMM, Hidden Markov Model (HMM)) to predict and replicate complex human movements [12], [20, 39, 23, 10]. Most of them concern movements of the upper limb [8, 55, 25].

Recently, the focus has been placed on the issue of how to determine the human intent of a continuous motion instead of discrete patterns. Han et al. [22] developed a state space EMG model based on Hill Muscle Model (HillMM) for continuous estimation of elbow joint which however involves many physiological parameters. The computational complexity of the method to predict joint motion states makes it unsuitable for real-time applications. An EMG-driven virtual arm has been developed by Manal, Gonzalez, Lloyd, and Buchanan [35] which reproduces the movement according to HillMM through the recognition of muscles activation . The behavior of a virtual arm is very faithful to a real one in contrast to a robot whose operation requires a specific translation of human movement to mechanical components. An estimation of elbow joint working online has been proposed by Chen et al. [11] who approached EMG patterns recognition through a Hierarchical Projected Regression (HPR) algorithm. It incrementally built a tree-based knowledge library, whose components represent local regression models. A single channel has been used as input signal, but effectiveness of the method has been supported just by some experiments on its own dataset. Looking at the results, it is almost clear that reliability strongly depends on the quality of the selected channel with the risk of suitableness for the specific case.

1.4 Scope of work

The innovation of this study is in estimating a single-joint angle by mean of a GMM trained with EMG features extracted through wavelet transform. A Gaussian Mixture Regression (GMR) technique is then used to retrieve the data from the trained model. This approach enables an autonomous extraction of the task-related information encoded in EMG signals, without loss of generality.

Moreover, such a probabilistic framework based on Mixture of Gaussians (MoG) distributions only require a reduced number of parameters to be kept, resulting in lightweight models. A GMM-GMR probabilistic framework requires low training data to achieve good results and provides fast regression that perfectly match with the use on an online application. Furthermore, it provides a robust mechanism to handle a possible loss of one input channel still providing a good estimation based on other information available.
The procedure has been applied to a humanoid robot, namely a NAO, produced by Aldebaran Robotics[1]. The producer company provides a software called Choregraphe which allows to interact with the robot giving him general instructions but this software could not be used in our research. As field of robotics has been growing in these years a platform of free libraries has been created to provide researchers (but not only) some basic tools to interface with robots and other electro-mechanical devices (i.e. Microsoft Kinect) which comes with the name of ROS. ROS libraries have been used to develop a software written in C++ language which analyzes EMG signals and provides the robot joint angles to assume (in our case the specific joint examined). Robot motion capabilities are not specular to the human ones so the movement should be converted to the mechanical possibilities of the machine.

[1]Aldebaran Robotics - SAS (Limited Company). www.aldebaran.com

Chapter 2

EMG analysis through time-domain and frequency-domain features

Working at the IAS-Lab (Intelligent Autonomous System Laboratory of University of Padua), Michieletto et al. [36] developed a working framework which estimates the knee bending angle. The system has been based on a GMM-GMR framework able to correlate EMG signals to the corresponding joint angles matching them through acquisition time. Starting from their study, in this thesis, we aim to remove the direct dependency from the time and explore EMG analysis as to make the model suitable for real-time purposes. Previous studies exploited good results in associating EMG to a self-built signal that highlights the succession of values over time. Nevertheless between EMG and angle position there is not a bijection since the signal is the sum of potentials of various muscles activation which are conditioned by actual position, velocity, previous movement and fatigue. It is like every piece of signal carries out the history of joint movement and varies accordingly to how much difference there is to the next one. For above reasons shall be analyzed at least a portion of the signal whose width has been investigated by [46] suggesting to use an interval of 150-250 mS.

2.1 Gaussian Mixture Model

For modelization purposes, we exploited a stochastic approach in order to address the high variability of the input EMG signals. Information extracted from EMG was used as input of a Gaussian Mixture Model (GMM) to estimate its correlation with the knee bending angle α.

The aim of GMM is to obtain the weighted sum of K Gaussian components which best approximates the input dataset representing the set of kick trials used for the training. In this particular case, the total number of data samples was $N = nT$, where n is the number of trials used to train the system, and $T = 2000$ is the number of observations acquired during each trial. A single data in input $\zeta_j, 1 \leq j \leq N$ at the framework is described in Equation 2.1.

$$\zeta_j = \{\xi(t), \alpha(t)\} \in \mathbb{R}^D \quad \xi(t) = \{\xi_{c,f}(t)\}_{c=1,\dots,C} \tag{2.1}$$

where:

- $\xi_{c,f}(t) \in \mathbb{R}$ is the value of the feature computed on c^{th} EMG channel at the time instant t and f is one of MAV,RMS or VAR;
- $\xi(t) \in \mathbb{R}^C$ is the set of features assumed from the considered channels, $1 \leq C = |\xi| \leq 8$, at the time instant t;
- $\alpha(t) \in \mathbb{R}$ is the joint bending angle at the time instant t;
- $3 \leq D \leq 10$ is the dimensionality of the problem.

The GMM was trained through the Expectation-Maximization (EM) algorithm [18], resulting in a probability distribution of the train dataset later used to perform the regression of the knee angle. The Expectation-Maximization algorithm iteratively estimates the optimal parameters $\theta = (\pi_k, \mu_k, \Sigma_k)$ that characterizes the K mixtures composing the GMM. The algorithm can be separated in two cyclic phases: Expectation and Maximization. The EM loop stops when the increment of the log-likelihood $\mathcal{L} = \sum_{j=1}^{N} \log\left(p\left(\zeta_j|\theta\right)\right)$ at each iteration becomes smaller than a defined threshold ϵ, given by $\frac{\mathcal{L}(t+1)}{\mathcal{L}(t)} < \epsilon$.

The Expectation step (E-step) is described as follows:

$$p_{k,j}(t+1) = \frac{\pi_k(t)\mathcal{N}(\zeta_j; \mu_k(t), \Sigma_k(t))}{\sum_{i=1}^{K} \pi_i(t)\mathcal{N}(\zeta_j; \mu_i(t), \Sigma_i(t))} \tag{2.2}$$

and the Maximization step (M-step) is computed according to:

$$\pi_k(t+1) = \frac{1}{N}\sum_{i=1}^{N} p_{k,j}(t+1) \tag{2.3}$$

$$\mu_k(t+1) = \frac{\sum_{i=1}^{N} p_{k,j}(t+1)\xi_j}{\sum_{i=1}^{N} p_{k,j}(t+1)}$$

$$\Sigma_k(t+1) = \frac{\sum_{i=1}^{N} p_{k,j}(t+1)(\zeta_j - \mu_k(t+1))(\zeta_j - \mu_k(t+1))^\top}{\sum_{i=1}^{N} p_{k,j}(t+1)}$$

The algorithm optimizes the parameters of the K Gaussian components by maintaining a monotone increasing likelihood during the local search of the maximum. This approach enables an autonomous extraction of the kick characteristic EMG signal while still maintaining an appropriate generalization.

Finally, the resulting probability density function is computed:

$$p\left(\zeta_j\right) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\zeta_j; \mu_k, \Sigma_k\right) \tag{2.4}$$

where:

- π_k are priors probabilities;
- $\mathcal{N}(\zeta_j; \mu_k, \Sigma_k)$ are Gaussian distribution defined by μ_k and Σ_k, respectively mean vector and covariance matrix of the k-th distribution.

The main drawback in the learning process lies in the EM requirement of a prior specification for the model complexity (i.e., the number of components K). On one hand, an overestimation of this parameter might lead to over-fitting and, consequently, to a poor generalization; on the other hand, an underestimation will result to poor regression performances. To deal with this issue we introduced an entropy based selection of the best number of components, K, in the GMM.

Several entropy based model selection techniques has been proposed in literature (e.g., Bayesian Information Criterion (BIC) [45], Akaike Information Criterion (AIC) [7], Minimum Description Length (MDL) [9], and Minimum Message Length (MML) [53]). Although, in [13] the authors proposed a specific criteria to estimate the value of the K parameter in the case of EMG signals, in this work we preferred a more standard approach based on BIC. In our experiments the whole learning process has been repeated with different GMM complexities by using BIC (Equation 2.5) as index of model quality with respect to the number of components K.

$$S_{BIC} = -2\mathcal{L} + n_p \log N \tag{2.5}$$

where:

- $\mathcal{L} = \sum_{j=1}^{N} \log\left(p\left(\zeta_j | \theta\right)\right)$ is the log-likelihood for the considered model θ;
- $n_p = (K-1) + K(D + \frac{1}{2}D(D+1))$ is the number of free parameters required for a mixture of K components with full covariance matrix.

The log-likelihood measures how well the model fits the data, while the second term is introduced to avoid data overfitting and maintain the model general enough. In our experiments the best BIC value was obtained with $K = 15$ components.

The Gaussian Mixture Regression (GMR) has been used to retrieve a smooth generalized version of the signal encoded in the associated GMM. So that, the conditional expectation of knee joint angle $\hat{\alpha}$ is calculated from the consecutive temporal value t and the EMG signals ξ known a priori. As we already said, the k-th Gaussian component is defined by the parameters (π_k, μ_k, Σ_k), where:

$$\mu_k = \{\mu_{p,k}\, \mu_{\alpha,k}\} \qquad \Sigma_k = \begin{bmatrix} \Sigma_{p,k} & \Sigma_{p\alpha,k} \\ \Sigma_{\alpha p,k} & \Sigma_{\alpha,k} \end{bmatrix} \tag{2.6}$$

with μ_p and Σ_p respectively the mean and the covariance of the known a priori information. The conditional expectation and its covariance can be estimated respectively using Equation 2.7 and 2.8.

$$\hat{\alpha} = E\left[\alpha \left| t, \xi\right.\right] = \sum_{k=1}^{K} \beta_k \hat{\alpha}_k \tag{2.7}$$

$$\hat{\Sigma}_s = Cov\left[\alpha \left| t, \xi\right.\right] = \sum_{k=1}^{K} \beta_k^2 \hat{\Sigma}_{\alpha,k} \tag{2.8}$$

where:

- $\beta_k = \frac{\pi_k \mathcal{N}\left(t, \xi_c \left| \mu_{p,k}, \Sigma_{p,k}\right.\right)}{\sum_{j=1}^{K} \mathcal{N}\left(t, \xi_c \left| \mu_{p,j}, \Sigma_{p,j}\right.\right)}$ is the weight of the k-th Gaussian component through the mixture;
- $\hat{\alpha}_k = E\left[\alpha_k \left| t, \xi\right.\right] = \mu_{\alpha,k} + \Sigma_{\alpha p,k}\left(\Sigma_{p,k}\right)^{-1}\left(\{t, \xi\} - \mu_{p,k}\right)$ is the conditional expectation of α_k given $\{t, \xi\}$;
- $\hat{\Sigma}_{\alpha,k} = Cov\left[\alpha_k \left| t, \xi\right.\right] = \Sigma_{\alpha,k} + \Sigma_{\alpha p,k}\left(\Sigma_{p,k}\right)^{-1}\Sigma_{p\alpha,k}$ is the conditional covariance of α_k given $\{t, \xi\}$.

Thus, the generalized form of the motions $\hat{\zeta} = \{t, \xi, \hat{\alpha}\}$ required only the weight, mean and covariance of the Gaussian components calculated through the EM algorithm.

2.1.1 Error estimation

In order to evaluate the effectiveness of the GMM-based system we exploited the Normalized Mean Square Error (NMSE). This function measures the goodness of fit between test and reference data, in our case $\hat{\alpha}$ (the data estimated through the GMR) and α (the angle calculated using the motion capture system):

$$NMSE(t) = 1 - \left\| \frac{\hat{\alpha}(t) - \alpha(t)}{\hat{\alpha}(t) - \mu_t(\alpha)} \right\|^2 \tag{2.9}$$

where:

- t is the temporal instant from the beginning of the trial (mS);
- $\hat{\alpha}(t)$ is the estimated angle at the instant t;
- $\alpha(t)$ is the angle calculated through the motion capture at the instant t;
- $\mu_t(\alpha)$ is the mean along the time of the angles given by the motion capture.

NMSE costs vary between $-\infty$ (bad fit) to 1 (perfect fit). Zero is the value reached from a straight line in fitting the reference.

2.2 Signals acquisition

Electromyography (EMG) signals were acquired with an active 8-channel wireless EMG system at 1000 Hz. The eight EMG electrodes were placed on the left leg of each subject in order to cover the principal muscular groups active during the kick task. In more detail, the following muscle were recorded: *Rectus femoris, Vastus lateralis, Vastus medialis, Tibialis anterior, Gastrocnemius lateralis, Gastrocnemius medialis, Biceps femoris caput longus, Peroneus longus.*

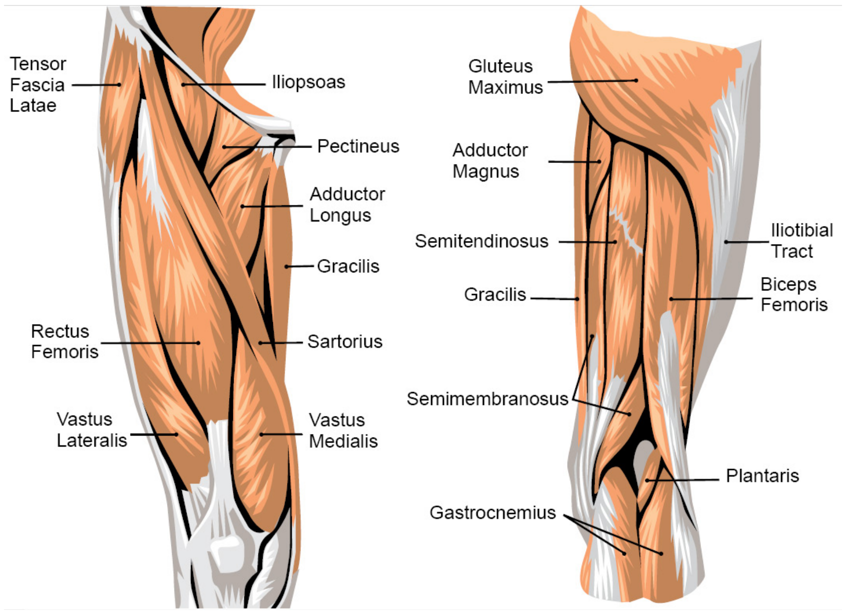


Figure 2.1: Muscles of upper leg.

Synchronously to the EMG signals, we recorded the kinematics of the left leg using an optoelectronic system. Six retro reflecting markers were placed on the subject's leg aka *LGT, LLE, LHF, LLM, LVMH, LVTH.* Six infrared digital cameras recorded the marker positions (namely, the leg kinematic) at 60 Hz during the whole recording session. Due to lower frequency of acquisition, kinematics were interpolated to match the 1000 Hz frequency of acquisition of EMG.

Three healthy volunteers (S1-S3; age 30 $\pm$ 4; one female) participated in the experiment. No motor related problems have been reported. The study was carried out in accordance with the principle of the Declaration of Helsinki[1]. Subjects were asked to naturally kick a ball (diameter 17 cm) from a sitting position. The ball was positioned on the floor at a fixed distance (2 cm) from the foot. After each kick, an operator was in charged of the ball reposition. As additional behavioral and motivational task, we asked the subjects to try to shot the ball in a goal in front of them (distance 350 cm, width 122 cm, height 76.2 cm). It is worth to notice that the task was fully self-paced. Each

[1]WMA Declaration of Helsinki - Ethical Principles for Medical Research Involving Human Subjects

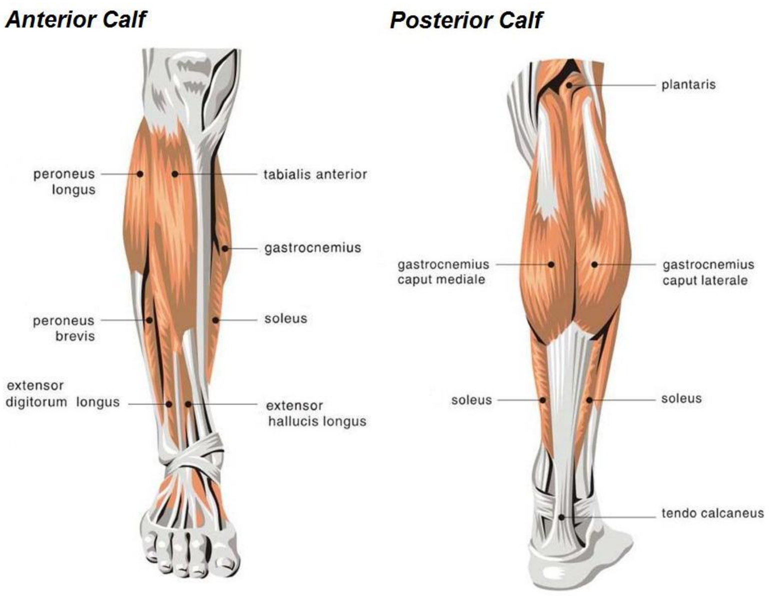


Figure 2.2: Muscles of lower leg.

participant performed several repetitions of the aforementioned task over a single recording session (day). Table 2.1 illustrates the number of trials performed and the duration of the experiment for each subject.

Subject	Kicks	Duration
S1	70	$\sim 8,5m$
S2	67	$\sim 6,9m$
S3	72	$\sim 6,5m$

Table 2.1: Number of kicks performed by each subject and the corresponding experiment duration.

Computation of sEMG signals for feeding the model was made offline using MATLAB[2] tools. Original signals were scanned to individuate and isolate every kick, selecting in our case a number of 2000 samples, since they were sufficient to cover the whole movement.

2.3 Features extraction

Our first analysis approach was to use some well known techniques of features extraction found in literature [42] [37][38] [36]. There are two main types of techniques: time-domain and frequency-domain. These features have been selected thanks to a previous research conducted at IAS-Lab in which they were successfully tested in an offline situation.

[2]MATLAB, The MathWorks, Inc., Natick, Massachusetts, United States

Figure 2.3: Performing kicks.

Time-domain Time domain techniques focus on instantaneous amplitude of signals. EMG was turned to frequency domain in order to apply a filter to remove artifacts and then turned back to time domain to be analyzed. Two variants were considered:

- **FW** - band-pass filtering (zero-phase digital filter) with 4th order Butterworth, [6-40] Hz cut-off frequencies.
- **RFW** - high-pass filtering (zero-phase digital filter, 4th order Butterworth, 40 Hz cut-off frequency) to remove artifacts; then full-wave rectification and low-pass filtering (4th order Butterworth, 6 Hz cut-off frequency)

Frequency-domain Analyzing signals directly in frequecy-domain allows to avoid conversion to time-domain which would save time for other processing. Each feature is based on the signal frequency spectrum which is computed by mean of Fast Fourier Transform (FFT) algorithm. Every window **W** considered was previously filtered with a 4th order Butterworth band-pass filter with [6-40] Hz cut-off frequencies to remove artifacts; then modulus was computed.

- **Frequency Spectrum Area (FSA)** - spectrum area of modulus of frequencies

$$FSA = \sum_{j \in W} |f_j|$$

- **Frequency Bands Area (FBA)** - band is subdivided into 5 sub-bands $([6,8],(8,12],(12-20],(20-29],(29-40])$ and for each one is calculated the FSA feature.

$$FBA = \{FSA_i\}$$
$$FSA_i = \sum_{j \in W_i} |f_j|$$
$$W = \bigcup_i W_i, \qquad i = 1, ..., 5$$

- **Mean of Square Frequencies (MNF)** - sum of product of the EMG power spectrum P_j and the frequency f_j divided by total sum of the spectrum intensity.

$$MNF = \frac{\sum_{j \in W} f_j P_j}{\sum_{j \in W} P_j}$$

FBA technique is the only one producing five features for each signal time which could yield an expensive computation for training using multiple channels.

2.4 Results

Every feature technique has been tested to signals of the three subjects using both one channel that three channels together. In the first case it was chosen the channel carrying out more information according to results showed in [36]. In the second case we chose *Rectus femoris*, *Vastus lateralis*, and *Vastus medialis* channels as the most informative ones. The following tables illustrate results of tests on features described above where the NMSE error (2.9) is used as comparison element.

Type	Subject 1	Subject 2	Subject 3	Sum
FW	0,4319	0,4615	-44,1704	-43,2770
RFW	0,5324	0,7873	-18,8374	-17,5177
FSA	0,5591	0,7915	-5,2848	-3,9342
FBA	0,8279	0,0433	-401,5978	-400,7266
MNF	0,1876	0,2324	-48,9210	-48,5010

Table 2.2: One channel results. Values of NMSE error for GMM training using time-domain features (FW, RFW) and frequency-domain features (FSA, FBA, MNF).

Table 2.2 and Table 2.3 highlight inefficacy of the features used as they yield low NMSE values. Although there are a few good values, e.g. Subject 2 with one channel and RFW or FSA features, generally this approach is not enough effective for our purposes. About half of tests shows an estimation lower than 0.5 or negative which means that randomness could achieve better performances. These bad results are due to signals nature which is supposed to be stationary while

Type	Subject 1	Subject 2	Subject 3	Sum
FW	0,0000	-0,0221	-0,0659	-0,0880
RFW	0,0000	0,4195	0,6831	1,1026
FSA	0,0000	0,4510	0,6750	1,1260
FBA	-2,6448	0,8968	0,5176	-1,2304
MNF	0,0000	0,1448	0,0336	0,0336

Table 2.3: Results for three channels. Values of NMSE error for GMM training using time-domain features (FW, RFW) and frequency-domain features (FSA, FBA, MNF).

EMG is not. Focusing on one side of EMG signals, i.e. frequency or time, leaves out a significant portion of information, so both aspects should be considered together. All types of features extraction based on Fourier Transform, Spectrum analysis or amplitude are not satisfactory and that is the reason why we looked for a different and more complete approach. Laterza and Olmo [29] and so many others (see Chapter 1) in their studies showed that wavelet transform could be a suitable alternative to other time frequency representations with the advantage of being linear, yielding a multiresolution representation and being unaffected by crossterms. Nowadays wavelet transform is a widely adopted approach to EMG analysis whose description follows in the next chapter.

Chapter 3

EMG analysis through wavelet transform

In the previous chapter I have explained way training a Gaussian Mixture Model only with time or frequency features of EMG signals yields to weak results. The major drawback is that both slow and fast changing properties of sEMG signals are relevant to the analyzing task. Slow variations provide information related to the body movements and tissue properties while the fast variations of signal are useful to understand the muscle activity and motor recruitment itself. Many researchers focused on alternative techniques of sEMG analysis coming to experience wavelet transform [12] [43] [39]. Advantages are linearity, multiresolution representation and being unaffected by cross terms [29]. Better results are obtained when wavelet shape matches the shape of the EMG as it achieves a good energy localization in the time-scale plane. Wavelet transform lead us different frequency components getting the most significant information of EMG signal spending less time and computation. Kumar et al. [26] studying the wavelet transform stated that

> The wavelet transform (WT) decomposes a signal into several multiresolution components according to a basis function called the "*wavelet function*". The "wavelet function" is both dilated and translated in time undertaking a two-dimensional cross correlation with the time domain SEMG signal. This method can be seen as a mathematical microscope that provides a tool to detect and characterize a short time component within a nonstationary signal. It is a technique that provides information related to the time-frequency variation of the signal. It has found numerous applications in data compression and feature enhancement for images, speech, and biosignals.

Wavelet transform has been used even in researches involving seismic signals, medical signals (electrocardiograms, electroencephalograms, TAC), audio signals (where Wavelet Packet Transform is preferred), and images (for problems like shape recognition, recognition of the edges, and compression).

Wavelet transform is considered a flexible technique because there is a wide variety of wavelet functions which could be used (Table 3.1). Moreover, anyone can design his own wavelet function according to his needs, provided that some mathematical constraints are satisfied.

3.1 Wavelet Transform

Wavelet transform is similar to Fourier transform with the exception that instead of using a basis composed by sine and cosine it uses particular functions that satisfy certain mathematical rules. Moreover, wavelet analysis allows to locate signal information both in time and in frequency, and can be applied in a useful manner in a wide window of time as well as a closer one. Signal is transformed into a linear composition of scaled and shifted version of a base function called *wavelet function*, *mother wavelet*, or *analyzing wavelet*.
Wavelet transform is subdivided in Discrete Wavelet Transform (DWT) and Continuous Wavelet Transform (CWT). To compute transform we use two functions: $\phi(t)$ called *scaling function* (or *father wavelet*) and $\psi(t)$ called *wavelet function* (*mother wavelet*) which form an orthonormal basis [14]. *Scaling function* allows to inspect low frequency features while *wavelet function* permits high frequency features. Projecting our signal to the basis yields to approximation (low frequency) and detail (high frequency) coefficients which are the real subject of our analysis.

Continuous Wavelet Transform of a time domain signal $x(t)$, as explained in [26], is defined as:

$$W(s,\tau) = \int x(t)\psi^*_{s,\tau}(t)dt \tag{3.1}$$

The *mother wavelet* ψ is a function with zero average

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0 \tag{3.2}$$

which is scaled by parameter s, that measures the degree of compression or scale, and translated by parameter τ, that determines the time location of the wavelet. This provides the information of the signal at different time and scales (i.e. the information of the signal localized in time and frequency). The scaled *mother wavelet* is defined as:

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\Big(\frac{t-\tau}{s}\Big) \tag{3.3}$$

By combining Equation 3.1 and Equation 3.3, the wavelet transform of the signal $x(t)$ can be written as:

$$W(s,\tau) = \frac{1}{\sqrt{s}}\int x(t)\psi\Big(\frac{t-\tau}{s}\Big)dt \tag{3.4}$$

When $|s| > 1$, $\psi_{s,\tau}$ has a larger time-domain than ψ and the wavelet correlates with the low-frequency components of the signal, while when $|s| < 1$ the wavelet is the compressed version (smaller support in time-domain) of the *mother wavelet* and correlates to high-frequency components. The factor $\frac{1}{\sqrt{s}}$ is used to preserve the energy. At large scale, the solution is coarse in the time domain and fine in the frequency domain. As the scale s decreases, the resolution in the time domain becomes finer while that in frequency domain becomes coarser. Like a windowed Fourier transform, a wavelet transform can measure the time-frequency variations of spectral components, but it has a different time-frequency resolution.

Referring to Chun Lin's tutorial ([14]), once the basis are known, we can formalize Discrete Wavelet Transform $f[n]$ of a discrete signal in $l^2(Z)$[1] by:

$$f[n] = \frac{1}{\sqrt{M}} \sum_k W_\phi[j_0, k]\phi_{j_0,k}[n] + \frac{1}{\sqrt{M}} \sum_{j=j_0}^{\infty} \sum_k W_\psi[j, k]\psi_{j,k}[n]. \tag{3.5}$$

where $\phi(t)$ is the scaling function, $\psi(t)$ is the wavelet function, j is the level of decomposition and M is the number of points. Because the sets $\{\phi_{j_0,k}[n]\}_{k\in Z}$ and $\{\psi_{j,k}[n]\}_{(j,k)\in Z^2, j\geq j_0}$ are orthogonal to each other we can take the inner product to obtain the wavelet coefficients

$$W_\phi[j_0, k] = \frac{1}{\sqrt{M}} \sum_n f[n]\phi_{j_0,k}[n]. \tag{3.6}$$

$$W_\psi[j, k] = \frac{1}{\sqrt{M}} \sum_n f[n]\psi_{j,k}[n] \; j \geq j_0. \tag{3.7}$$

which are the approximation coefficients (Equation 3.6) and detail coefficients (Equation 3.7) mentioned above. The DWT can be calculated using matrix multiplication, but a more efficient implementation uses digital filter bank, e.g., Mallat's algorithm [34]. This technique uses filter bank pairs which involve lowpass and bandpass filters and down-sampling (decimation).

> The fundamental idea of multiresolution analysis is to represent a function as a limit of successive approximations, each of which is a "smoother" version of the original function. The successive resolutions correspond to different resolution, which leads to the name multiresolution analysis as a formal approach to construct orthogonal wavelet bases using a definite set of rules and procedures. It also provides the existence of so-called *scaling functions* and *scaling filters* which are then used for constructions of wavelets and fast numerical algorithms. In applications, it is an effective mathematical framework for hierarchical decomposition of a signal or an image into components of different scales represented by a sequence of function spaces on $\mathbb{R}$. [17]

[1] $l^2(Z) = \{f[n] | \sum_{n=-\infty}^{\infty} |f[n]|^2 < \infty\}$

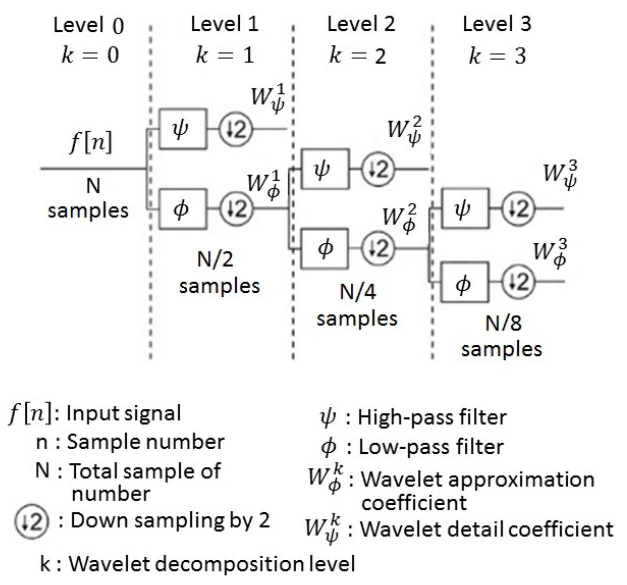


Figure 3.1: Usage of filter bank for signal decomposition.

In Figure 3.1 there is a visual example of how Mallat's algorithm works. It is based on a filter bank decomposition on an orthogonal base realized by a convolution between an original signal f[n] and two filters: an high-pass filter ($\psi_{j,k}[n]$) which computes $W_\psi[j,k]$ (detail coefficients) at decomposition level k; a low-pass filter ($\phi_{j_0,k}$) which computes $W_\phi[j_0,k]$ (approximation coefficients) at decomposition level k. Both approximation and detail wavelet coefficients at level k only depend on the approximation wavelet coefficient at level $(k-1)$. This set of two wavelet filters composes the *mother wavelet*, which shape is adjusted by the coefficients, contrary to the Fourier transform where the convolution function is fixed. [40]
Effectiveness of wavelet transform to extract appropriate features strongly depends on selection of *mother wavelet.* Even when applied to same signal, each function extracts very different features. Some popular families of wavelet basis functions are *Haar*, *Daubechies*, *Coiflet*, *Symmlet*, *Morlet*, and *Mexican Hat.*

Even though there is no well-defined rule for selecting a wavelet basis function some *mother wavelets* are more suitable for a given signal type. Some general guidelines for the choice of wavelets are: *Db4* is more suitable for signals that have "linear approximation" over the support of four samples, while *Db6* is better suited for a signal approximated by a quadratic function over the support of six, *coiflet6* provides better data compression results while *Db4* is more suitable for feature extraction [26][52].

3.1.1 Properties of wavelet transform

The following are some of the important properties of wavelet functions reported in [26]. They are useful to choose the *mother wavelet* in order to obtain a best

Wavelet Family	Wavelet function with orders
Daubechies	db1 or haar, db2, db3, db4, db5, db6, db7, db8, db9, db10
Symlets	sym2, sym3, sym4, sym5, sym6, sym7, sym8
Coiflet	coif1, coif2, coif3, coif4, coif5
BiorSplines	bior1.1, bior1.3, bior1.5, bior2.2, bior2.4, bior2.6, bior2.8, bior3.1, bior3.3, bior3.5, bior3.7, bior3.9, bior4.4, bior5.5, bior6.8
ReverseBior	rbio1.1, rbio1.3, rbio1.5, rbio2.2, rbio2.4, rbio2.6, rbio2.8, rbio3.1, rbio3.3, rbio3.5, rbio3.7, rbio3.9, rbio4.4, rbio5.5, rbio6.8
Discrete Meyer	dmey

Table 3.1: Most used *mother wavelets* or *wavelet functions*

match with the characteristics of the signal examined.

The smoothness is indicated by the regularity of the function. Regularity of r means that the rth derivative exists almost everywhere in the function. The *Haar* wavelet has regularity of zero since it is a discontinuous function.

Time and frequency localization is defined as the ability of wavelet function to localize in time and frequency. This is generally inversely related to the smoothness of the wavelet function.

Zero moments. A wavelet function with N zero moment satisfies:

$$\int_{-\inf}^{\inf} t^n \psi(t) dt = 0 \quad n = 0, 1, 2, ..., (N-1)$$

The zero moment represents the polynomial degree of the wavelet function. In the frequency domain, this may be seen as related to the slope of the bandpass filter characteristics: the higher the *"zero moment"* the sharper the cutoff of the filter is.

Symmetry. A function is defined as a symmetric function about τ if

$$f(t+\tau) = f(-t+\tau)$$

A symmetric wavelet function introduces linear phase when used as impulse response in Mallat's algorithm. The symmetry property of the wavelet function is required when the shape of the signal has to be maintained.

3.2 Experiment on kicks

The elaboration of sEMG signals was made at first offline using MATLAB[2] tools, because they already include implementation of wavelet transform with possibility to set the wavelet function desired. Original signals were scanned to individuate and isolate every kick, selecting in our case a portion of about 2 seconds (2000 samples) for each one, since it was sufficient to cover the whole movement. To extract features corresponding to instant t was considered the portion of sEMG signal going from $t - window$ to t, where *window* was large 200 mS (200 samples). The length of the window was selected as an intermediate value of the interval proposed in [46], where is showed that an optimal window to deal with sEMG should be between 150 and 250 mS. Current window was processed with wavelet transform using a certain *mother wavelet* at first decomposition level as recommended in studies of Mahdavi et al. [33] and Elamvazuthi et al. [19]. To synthesize information carried out by coefficients resulting from wavelet decomposition three statistical measures were chosen: MAV (Equation 3.8), RMS (Equation 3.9) and standard deviation (VAR, Equation 3.10). First two were successfully adopted in [19], [39] and [32], while the last one is a commonly used feature which was chosen to make comparison with first two.

$$MAV = \frac{1}{N} \sum_{k}^{N} |x_k| \tag{3.8}$$

$$RMS = \sqrt{\frac{1}{N} \sum_{k}^{N} x_k^2} \tag{3.9}$$

$$VAR = \sqrt{\frac{\sum_k^N (x_k - \overline{x})^2}{N}} \tag{3.10}$$

where $\{x_k\}$ are wavelet coefficients and $\overline{x}$ is their expectation.

Once retrieved features for time t, window was shifted by one to compute features for $t + 1$, and procedure repeated for the entire signal. For each kick 1800 features were collected since they had a length of 2000 samples. These features were relative to a single channel, so using only one channel we will have one feature per knee bending angle α, while using more channels we will have a feature per channel associated to angle value. If we use three channels, for each time t we will have three features and one angle values. Say $f_{s,j}(t)$ is the feature computed, where s is one of MAV, RMS and VAR; j is the number associated to one of the eight channels; t is time considered. When we used three channels each input point for the GMM is composed of $\{f_{s,1}(t), f_{s,2}(t), f_{s,3}(t), \alpha(t)\}$ where $\alpha(t)$ is the bending angle at time t and f is obtained processing signals corresponding to $[t - window, t]$. When we used one channel, say the first, each input point is composed of $\{f_{s,1}(t), \alpha(t)\}$, and so on.

[2]MATLAB, The MathWorks, Inc., Natick, Massachusetts, United States

The optimal number of channels to consider was investigated to find out the better choice (see Section 3.3).
To investigate the choice of the *mother wavelet* and the number of channels, training was made with ten kick repetitions and testing was made on three kick repetitions. GMM was fed with features collected for each kick having back means and covariances of the k gaussian components of the model. Regression was made to obtain estimated α values and then compared to real ones through NMSE computation.

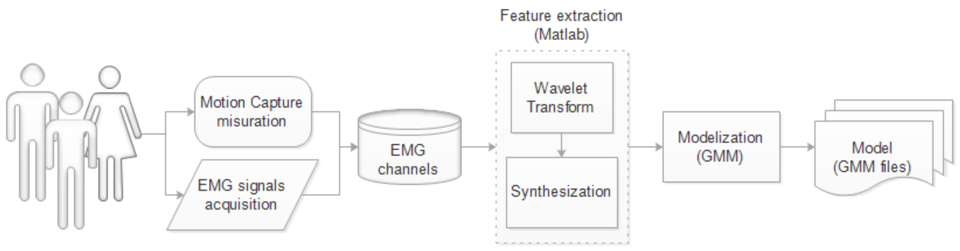


Figure 3.2: Flow of creation of the model

3.3 Selection of number of channels

To obtain better performances the whole information should be considered, this means that all the eight channels available should be included. However, computation would be long and complex, and in a real application may not be available a such large number of input. Thence was studied which is the minimum number of channels necessary to achieve the goal. *Mother wavelet* used was *Daubechies Db2* (as the one more promising and with faster transform computation), and synthesizing measure was MAV as it is the one that in literature gave higher performances.

Firstly it was tested each channel by its own to find out which ones were more informative, producing results of Table 3.2; then the three channels with higher values were used in pair producing results showed on Table 3.3; Finally the three channels were tested together (Table 3.4) and results were compared.

As showed by results reported in Table 3.2, the most informative channels are *Rectus femoris*, *Vastus lateralis* and *Vastus medialis*. Training with these three channels together yields to higher performances than with a lower number, as expected.

Since already with three channels the NMSE reaches good values, training with even more channels was not investigated, keeping it for a future work. In Figure 3.3, Figure 3.4 and Figure 3.5 are shown original sEMG of the three channels selected (a), with signals obtained after computation of MAV (b) of detail coefficients of first level decomposition.

#	Channel	Subject 1	Subject 2	Subject 3	Sum
1	Rectus Femoris	0,5917	0,4523	0,1727	1,2167
2	Vastus Lateralis	0,7401	0,5403	0,5426	1,823
3	Vastus Medialis	0,5917	0,8093	0,6266	2,0276
4	Tibialis anterior	0,0674	-0,2407	0,4706	0,2973
5	Gastrocnemius lateralis	-0,5058	0,3614	-0,3206	-0,465
6	Gastrocnemius medialis	0,5107	0,0233	0,191	0,725
7	Biceps femoris caput longus	-0,0022	0,5074	-0,0281	0,4771
8	Peroneus longus	0,1567	-0,0734	0,0380	0,1213

Table 3.2: NMSE values resulting from tests using only one channel for training GMM. First three channels showed higher values and were investigated more in depth.

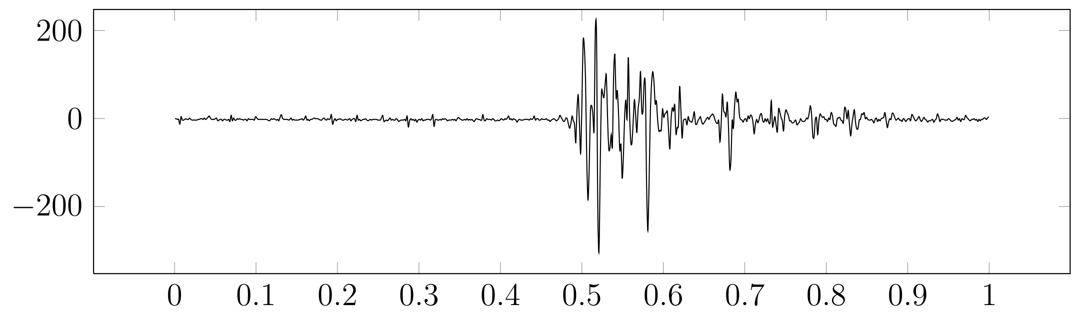
# of channels	Subject 1	Subject 2	Subject 3	Sum
1 - 2	0,8401	0,4427	0,7123	1,9951
2 - 3	0,8410	0,8657	0,5367	2,2434
1 - 3	0,6393	0,8525	0,7986	2,2904

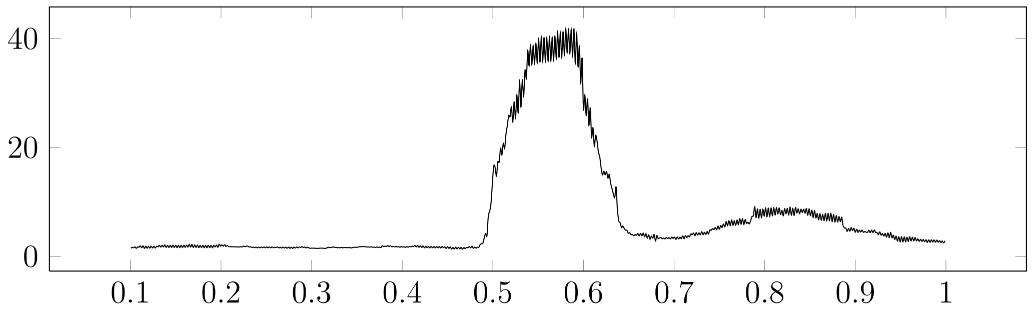
Table 3.3: NMSE values resulting from tests using two channels of the best three of Table 3.2, *dB2* and MAV, for training GMM. Couples of channels exploited similar performances.

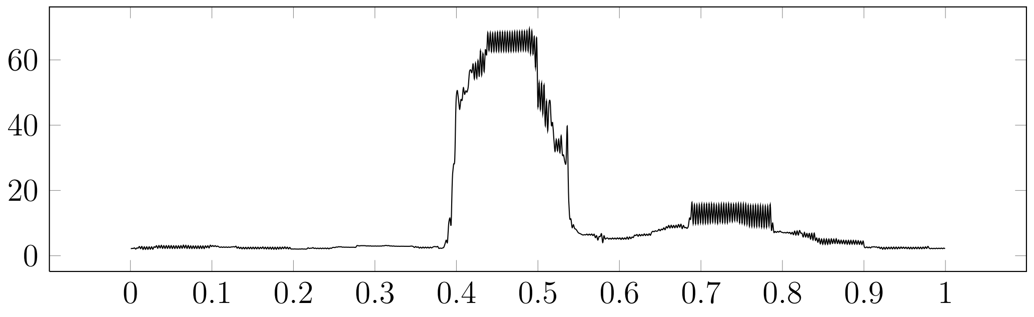
# of channels	Subject 1	Subject 2	Subject 3	Sum
1 - 2 - 3	0,8256	0,8484	0,7539	2,4279

Table 3.4: NMSE values resulting from tests using best three channels of Table 3.2, *dB2* and MAV, for training GMM.

Figure 3.3: Original sEMG of first channel (rectus femoris) of subject 1 (a) with results of elaboration: wavelet transform and MAV (b),RMS (c) and variance (d) of first level decomposition coefficients.


(a) original sEMG


(b) MAV of wavelet coefficients


(c) RMS of wavelet coefficients

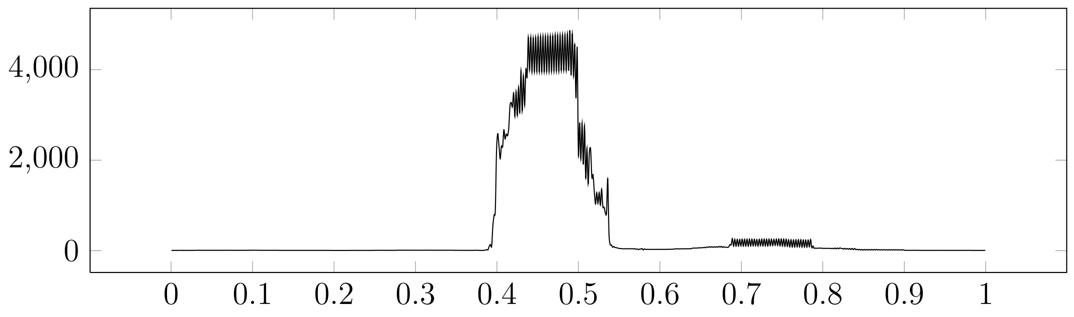
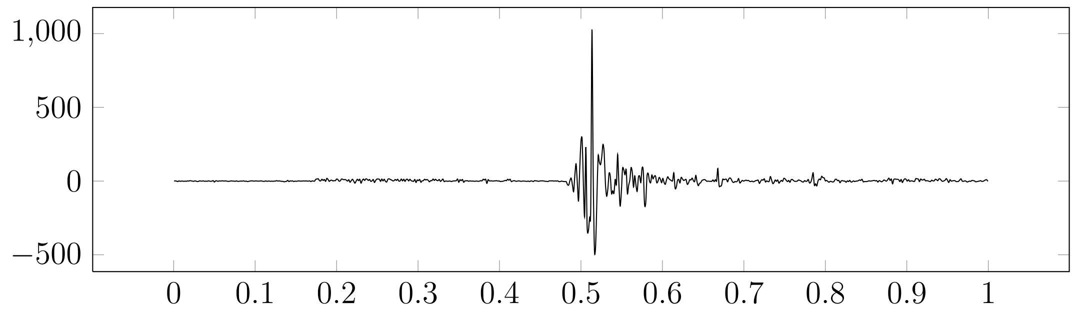
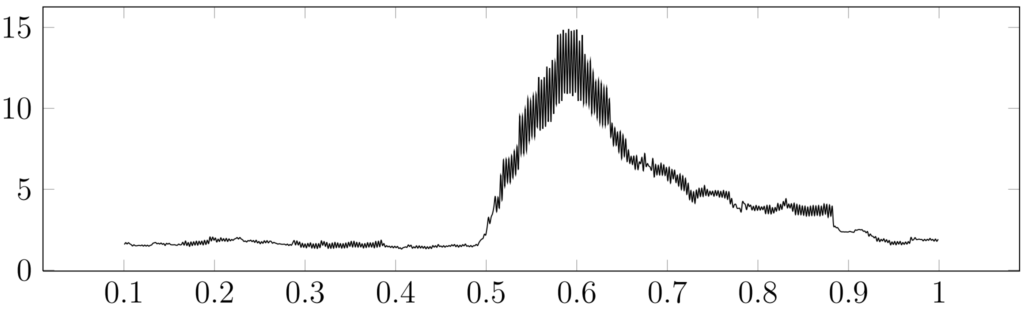
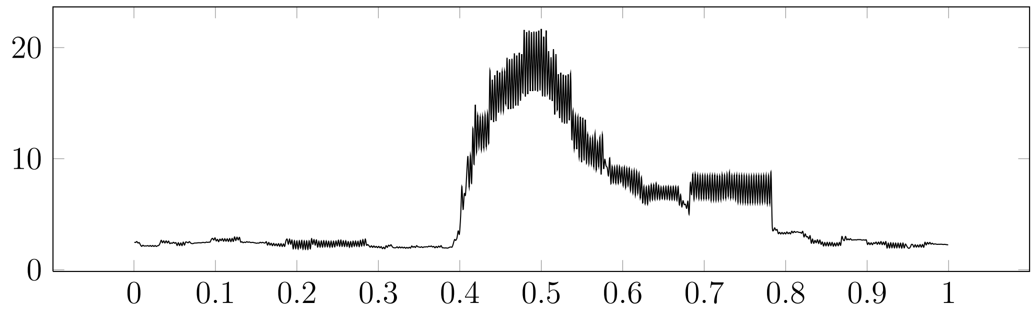

(d) Variance of wavelet coefficients

Figure 3.4: Original sEMG of second channel (vastus lateralis) of subject 1 (a) with results of elaboration: wavelet transform and MAV (b),RMS (c) and variance (d) of first level decomposition coefficients.


(a) original sEMG


(b) MAV of wavelet coefficients


(c) RMS of wavelet coefficients

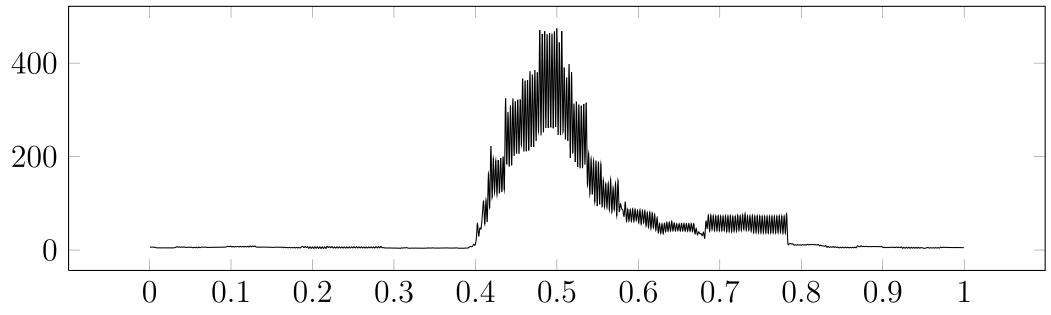
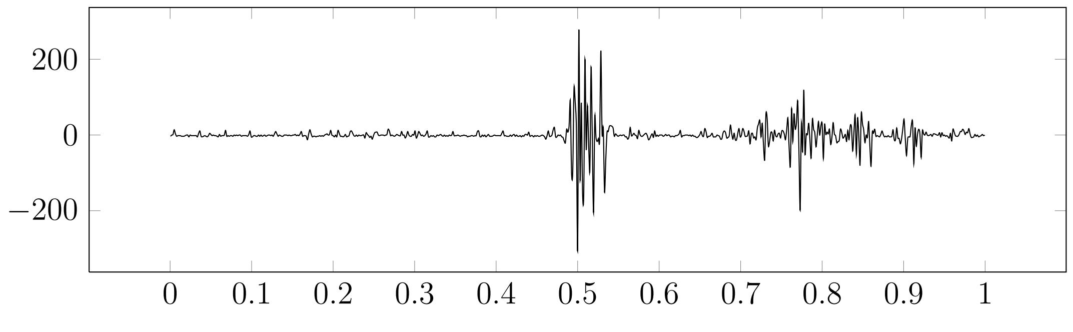
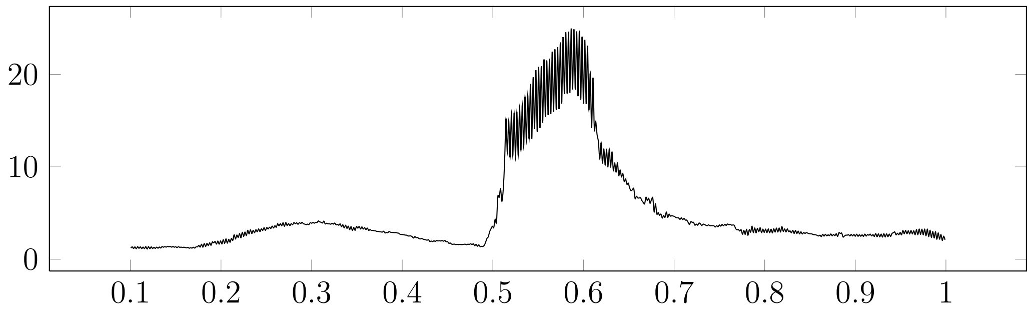

(d) Variance of wavelet coefficients

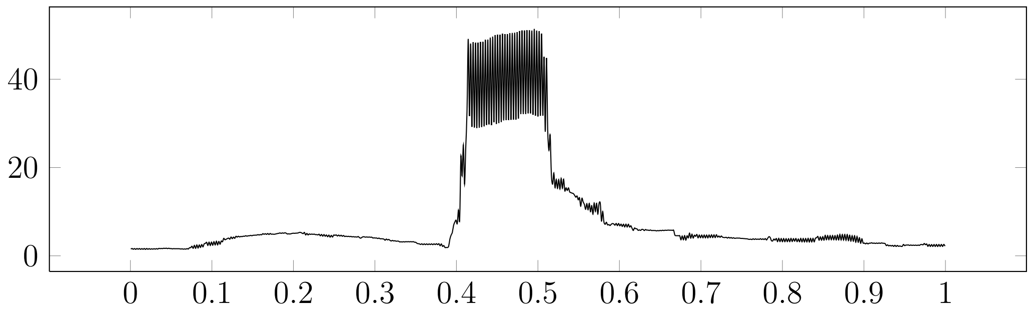
Figure 3.5: Original sEMG of third channel (vastus medialis) of subject 1 (a) with results of elaboration: wavelet transform and MAV (b),RMS (c) and variance (d) of first level decomposition coefficients.



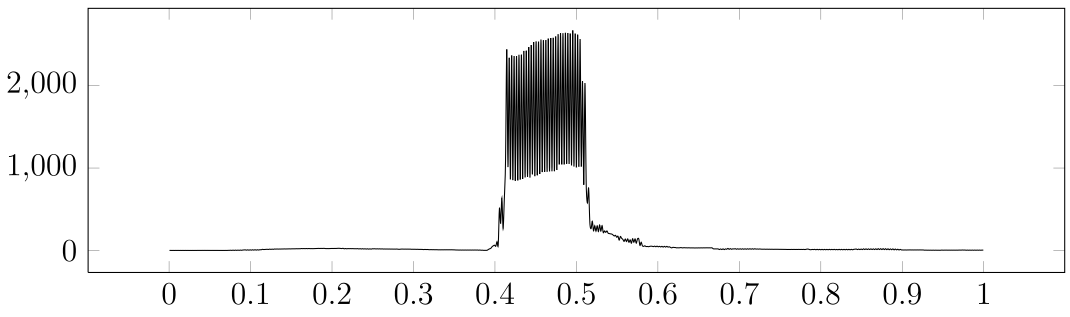
(a) original sEMG



(b) MAV of wavelet coefficients



(c) RMS of wavelet coefficients



(d) Variance of wavelet coefficients

3.4 Selection of *mother wavelet*

In their research on wavelet transform, Chowdhury et al. [12] showed that analyzing sEMG signals using Daubechies function renders better features extraction. For obtaining good results from a sEMG analysis on different applications, they recommend to use Daubechies family, in particular functions *Db2*, *Db4*, *Db6*, *Db44* and *Db45*, at decomposition level 1 according to [33]. Other successful works in using Daubechies are of Elamvazuthi et al. [19], Reaz et al. [41] and Phinyomark et al. [39], which include also *Db7* to list of candidate *mother wavelet*s. Each function mentioned above was tested to discover the one that leads to better results in this case study. Table 3.6, Table 3.5 and Table 3.7 show values of NMSE exploited by testing every *mother wavelet* with three channels and with different synthesization measures. An example of signals resulting from elaboration can be seen in Figure 3.3, Figure 3.4 and Figure 3.5 where *mother wavelet* used is *Db2*.

RMS					
Mother Wavelet	Subject 1	Subject 2	Subject 3	Sum	%
db2	0,5701	0,7409	0,8583	2,1693	72,31
db4	0,5806	0,8008	0,5825	1,9639	65,46
db6	0,5455	0,6945	0,8531	2,0931	69,77
db7	0,7177	0,8092	0,8678	2,3947	79,82
db44	0,6306	0,7931	0,6455	2,0692	68,97
db45	0,7258	0,8345	0,7572	2,3175	77,25

Table 3.5: NMSE values of tests to find best *mother wavelet* within MAV of first level decomposition coefficients.

MAV					
Mother Wavelet	Subject 1	Subject 2	Subject 3	Sum	%
db2	0,8256	0,8484	0,7539	2,4279	80,93
db4	0,8021	0,7779	0,4415	2,0215	67,38
db6	0,8051	0,8397	0,3710	2,0158	67,19
db7	0,7732	0,8544	0,6568	2,2844	76,15
db44	0,7574	0,5983	0,6234	1,9791	65,97
db45	0,5426	0,7828	0,3928	1,7182	57,27

Table 3.6: NMSE values of tests to find best *mother wavelet* within RMS of first level decomposition coefficients.

Best performance was achieved with *mother wavelet Db2* and MAV features with an 80,93 % of correct estimation, like can be seen in Table 3.6, although *Db7* and RMS exploited values slightly lower with its 79,82%, see Table 3.5. Estimation using variance as feature showed lower performances of at least ten percent

VAR					
Mother Wavelet	Subject 1	Subject 2	Subject 3	Sum	%
db2	0,7252	0,7171	0,6139	2,0562	68,54
db4	0,5967	0,8694	0,6563	2,1224	70,75
db6	0,4321	0,7946	0,5203	1,7470	58,23
db7	0,6760	0,8494	0,5909	2,1163	70,54
db44	0,6150	0,7642	0,6279	2,0071	66,90
db45	0,3603	0,7446	0,5189	1,6238	54,13

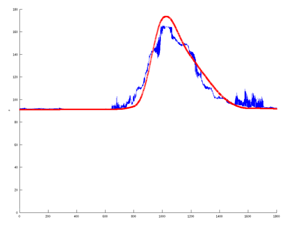
Table 3.7: NMSE values of tests to find best *mother wavelet* within variance of first level decomposition coefficients.

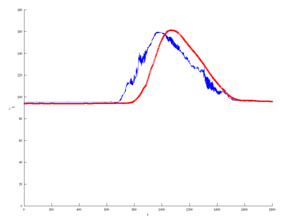
respect to the other ones, so it appears not to be suitable for this kind of analysis (Table 3.7). Obtained results indicate that is not possible to consider one *mother wavelet* better than the others since further computations have equal importance as choice of *wavelet function*. Same wavelet transform could be suitable or not depending on how information extracted is treated. An example is *Db45* which exploits a 77,25 % of correct estimation when used with RMS (Table 3.5) while exploits only a 54,15 % when used with standard deviation (Table 3.7). Combinations with higher values are the ones which yield a similar (and good) estimation for all the three subjects, while the others may fit very well for one subject but very bad for the other two. *Db2* and *Db7* have been selected to further investigations. In Figure 3.6 and Figure 3.7 it is possible to see curves of estimated angles (in blue) against originals (in red). Pictures which show better alignment are those with higher NMSE values.

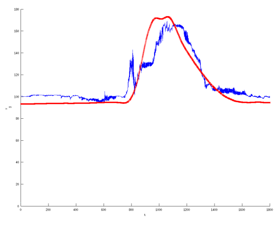
3.5 Tests on number of repetitions

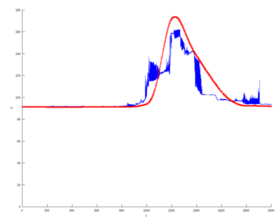
The dataset of kicks includes 63 repetitions of the movement for each subject. Thence *Db2* and *Db7* were tested on complete data, searching for the minimum number of kicks which could achieve good performances. In Table 3.9 and Table 3.8 you can see NMSE values on training with incremental number of repetitions stepped by ten.

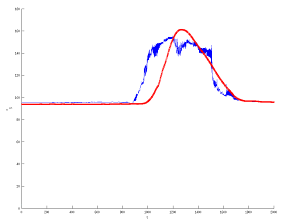
Values for *Db7* seems not to have a precise trend, while with *Db2* is observable a linear proportion within number of kicks and NMSE. Increase in kicks yields to higher effectiveness as it was expected. For Subject 1 and Subject 2 good results are obtained even with a few repetitions given that with only 5 kicks we have a correct estimation in 80% of times. Anyway a small training set shall not be used in order to avoid overfitting problems. According to Table 3.9, to achieve a good modeling also for Subject 3 almost 50 repetitions are needed.

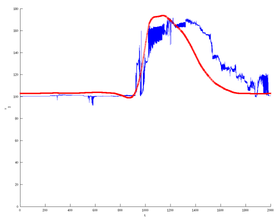

(e) S1_Db2_MAV

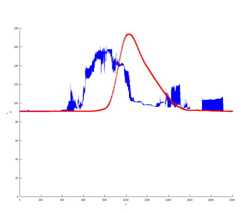

(f) S2_Db2_MAV

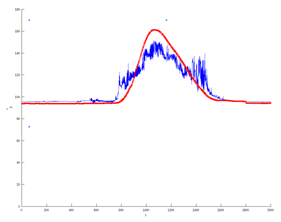

(g) S3_Db2_MAV


(h) S1_Db2_RMS


(i) S2_Db2_RMS


(j) S3_Db2_RMS


(k) S1_Db2_VAR


(l) S2_Db2_VAR

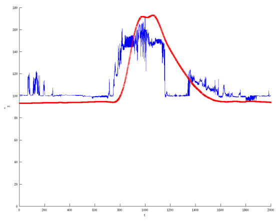
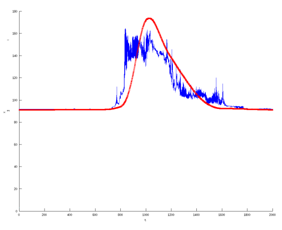
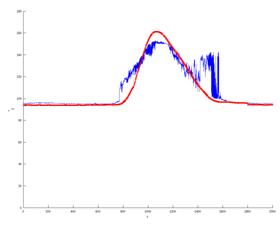
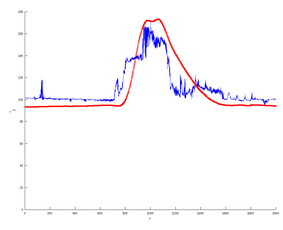
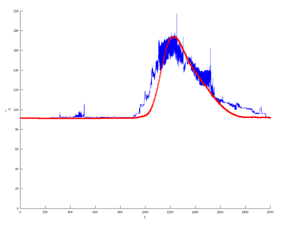

(m) S3_Db2_VAR

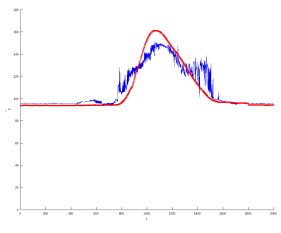
Figure 3.6: Angles of knee joint estimated (in blue) against angles to predict (in red) of subjects when used wavelet Db2.

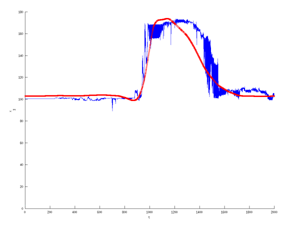

(a) S1_Db7_MAV

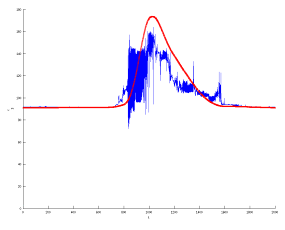

(b) S2_Db7_MAV

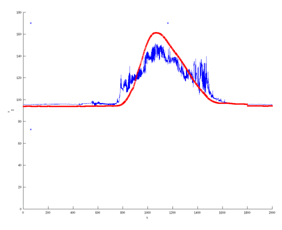

(c) S3_Db7_MAV


(d) S1_Db7_RMS


(e) S2_Db7_RMS


(f) S3_Db7_RMS


(g) S1_Db7_VAR


(h) S2_Db7_VAR

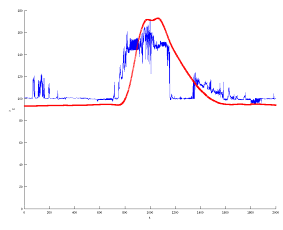
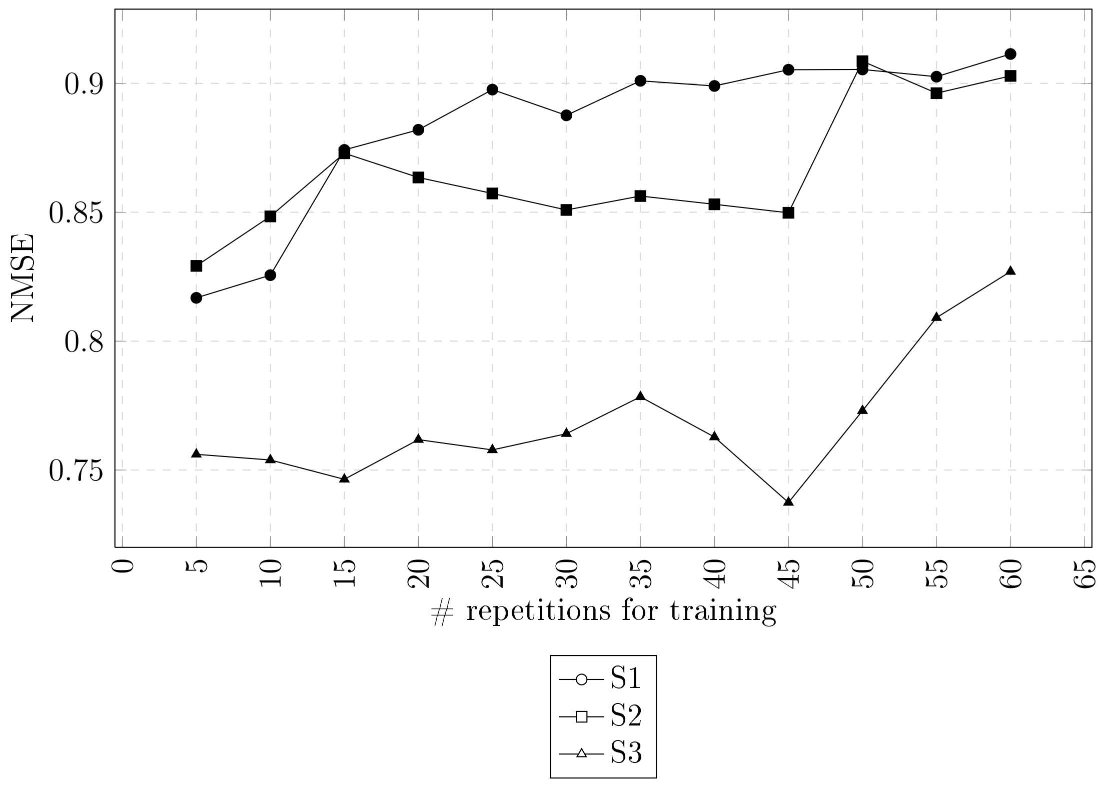

(i) S3_Db7_VAR

Figure 3.7: Angles of knee joint estimated (in blue) against angles to predict (in red) of subjects when used wavelet Db7.

# kicks	Subject 1	Subject 2	Subject 3	%
5	0,4873	0,8711	0,8668	74,17
10	0,6230	0,7987	0,8678	76,32
15	0,7782	0,7753	0,8594	80,43
20	0,8622	0,8021	0,8863	85,02
25	0,7445	0,7183	0,8831	78,20
30	0,8824	0,7836	0,8858	85,06
35	0,8764	0,7689	0,8838	84,30
40	0,7505	0,8392	0,8844	82,47
45	0,9071	0,8579	0,8814	88,21
50	0,7258	0,8831	0,8802	82,97
55	0,8771	0,8889	0,8624	87,61
60	0,8882	0,8702	0,8584	87,23

Table 3.8: Results of tests with incremental number of repetitions with *Db7* and RMS.

Figure 3.8: Results of tests with incremental number of repetitions with *Db2* and MAV.



3.6 Experiment on step

This second experiments was made to confirm the results obtained from the first one. A more challenging but common movement has been considered in

# kicks	Subject 1	Subject 2	Subject 3	%
5	0,8168	0,8292	0,7561	80,07
10	0,8256	0,8484	0,7539	80,93
15	0,8742	0,8729	0,7464	83,12
20	0,8820	0,8635	0,7618	83,58
25	0,8976	0,8573	0,7578	83,76
30	0,8876	0,8509	0,7641	83,42
35	0,9010	0,8563	0,7784	84,52
40	0,8990	0,8531	0,7628	83,83
45	0,9053	0,8498	0,7374	83,08
50	0,9054	0,9086	0,7730	86,23
55	0,9026	0,8962	0,8091	86,93
60	0,9114	0,9029	0,8270	88,04

Table 3.9: Results of tests with incremental number of repetitions with *Db2* and MAV.

order to evaluate framework performances in a different scenario. Three subjects (S4-S6) were asked to walk at self selected speed on a 8 m walkway. Several gait trials were acquired and, out of them, 8 (train) + 2 (test) stance phases of gait were extracted for each subject. In this case a greater amount of muscles participate to the movement and therefore are activated which means that signals could be more crossed each other. Focus has been set on a different joint to be sure that the model does not rely on some peculiar characteristics of the knee. We decided to use the three most informative channels from a biological point of view, skipping intentionally a survey on others. The results of the previous experiment helpd us to choose the *mother wavelet* and synthesis measure which led to better performances. All the available training steps have been used to compute the GMM representing the motion for each subject. It is worth to notice that the speed of the gait influence directly the number of samples considered for the subjects. The more rapidly they walked, the less samples we obtained for analyzing the movement. Again, NMSE has been used to evaluate the framework performances (Table 3.10).

ID subject	samples	NMSE
S4	651	0.9052
S5	550	0.8344
S6	701	0.8170

Table 3.10: NMSE of ankle motion estimation during the stance phase of gait.

Despite the lower number of repetitions and the lower number of samples due to shorter movement, the results are consistent with respect to the previous dataset emphasizing the performances of the proposed framework when facing a small amount of input data.

3.7 Stretched features

The model is based on three features computed on different channels but reflect the past history of the signal only implicitly. A point of the model consists of three features corresponding to a determined position, but does not give information on the phase of the movement. During the kick there are an ascending phase and a descending one where the leg reaches the common positions from angle point of view, but not from energy transmitted by the muscles and therefore the speed associated with the movement. From the analysis of EMG signals was observed that frequency and amplitude variations correspond to variations in speed and strength in the leg. To ensure that this information is considered by the model, a possible option is to expand the number of features corresponding to a given instant involving some of the previous values. The use of greater size points increases the creation cost of the model by the GMM, and also increase the number of parameters affecting the lightness and usability of the model, especially in case of whole body extension. All these factors deserve to be studied in depth, but here they do not find adequate space. It was therefore decided to do some exploration testing to check if this might be a viable route to be pursued in the future.

Instead of using a point with four dimension (one point per channel plus one for target) is used a point of 16 components: five for each channel and one for target. In first experiment (Exp 1) the five components per channel contain last five features computed, while in the second experiment (Exp 2) features are more stretched and correspond to intervals spaced ten. Recalling notation of 2.1 we have:

$$\zeta_{j,Exp1} = \{\xi(t), \xi(t-1), \xi(t-2), \xi(t-3), \xi(t-4)\alpha(t)\} \in \mathbb{R}^{5C+1} \tag{3.11}$$

$$\zeta_{j,Exp1} = \{\xi(t), \xi(t-10), \xi(t-20), \xi(t-30), \xi(t-40)\alpha(t)\} \in \mathbb{R}^{5C+1} \tag{3.12}$$

The number of components and the distances between the previous features have been arbitrarily chosen. Features were collected using *Db2* and MAV. The model was created with GMM and effectiveness evaluated with NMSE as in the previous experiments. In Table 3.11 are reported results obtained.

Comparing result of Table 3.11 to ones of Table 3.9 you can see a general improvement of the model, more pronounced in the case of the subject 2 and more evident in experiment 2 in which the improvement is about ten percent. These results give hope that an improvement in the model is possible using the tools adopted so far and show that this path should be more explored in the future to get a better modelization.

ID subject	# train	Exp 1	Exp 2
S1	30	0.9367	0.9337
S2	30	0.9510	0.9404
S3	30	0.7912	0.9573

Table 3.11: NMSE of kick motion with stretched features (*Db2*,MAV) corresponding to $t, t-1, ..., t-4$ (Exp 1), and to $t, t-10, ..., t-40$ (Exp 2).

Chapter 4

Interfacing with robot

The framework described in Chapter 2 and Chapter 3 has been tested in a real situation controlling the humanoid robot NAO produced by Aldebaran Robotics. In order to use open source libraries a ROS based system has been developed. The software is able to simulate generation of EMG, to compute signal analysis and to tell robot the position to set. For each step there is a dedicated ROS node which performs its task through reception and dispatch of ROS messages. Figure 4.1 shows blocks which form the framework, starting from EMG signals, passing through features extraction, until robot motion. The chart shows how it is possible to modify and separate each step according to needs; for example if we want to change the humanoid robot, or if we want to use a different kind of model. Due to mean processing time, software can generate messages for the robot at about 240 Hz, which is quite well for general real-time applications, but very high for NAO that can work properly with a maximum of 50 Hz. When robots are used to replicate human movements it should be carefully checked whether motors and joints allow to perform motion the same way. In the case of knee joint there are not particular problems since NAO limits almost match ones of a human being, but could arise some if for example are considered also hip and ankle joint.

4.1 NAO robot[1]

NAO is a 58 cm tall humanoid robot. It is small, cute and round. NAO is intended to be a friendly companion around the house. It moves, recognizes you, hears you and even talks to you. Since his birth in 2006, it has been constantly evolving to please, amuse, understand and love you. In short, to one day become your friend. Aldebaran created NAO to be a true daily companion. It is the little creature who helps you be your best. His humanoid form and extreme interactivity make him really endearing and loveable. While waiting to be ready for home use, NAO became a star in the world of education. In more than 70 countries, it was used in

[1]Description of robot is retrieved from owner site [1]

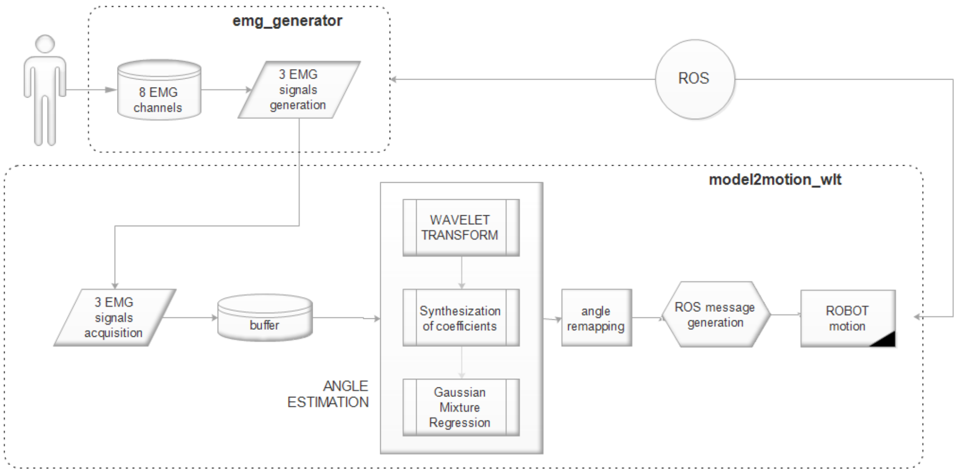


Figure 4.1: Blocks which form the framework that are independent each other.

computer and science classes, from primary school through to university. Thanks to NAO, students can learn programming in a fun and practical way. They can program him to walk, catch small objects and even dance.

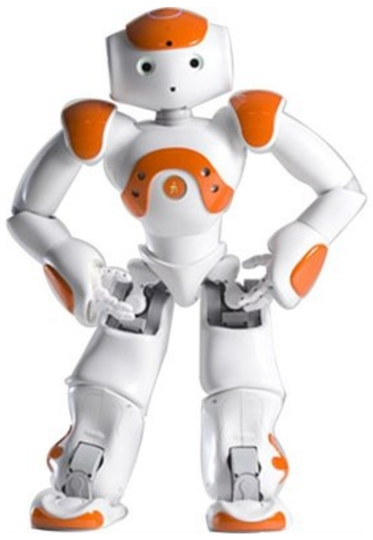


Figure 4.2: NAO robot

NAO has:

- Body with 25 Degree of Freedom (DoF) whose key elements are electric motors and actuators (Figure 4.4)
- Sensor network: two cameras, four directional microphones, sonar rangefinder, two IR emitters and receivers, one inertial board, nine tactile sensors and eight pressure sensors
- Various communication devices, including voice synthesizer, LED lights, and 2 high-fidelity speakers

- Intel ATOM 1,6 GHz CPU (located in the head) that runs a Linux kernel and supports Aldebaran's proprietary middleware (NAOqi)
- Second CPU (located in the torso)
- 48.6-watt-hour battery that provides NAO with 1.5 or more hours of autonomy, depending on usage

This combination of technologies (and many others) gives NAO the ability to detect its surroundings. Now it must interpret what it detected. This is where the embedded software in NAO's head comes in. Aldebaran created a dedicated operating system, *NAOqi*, allowing the small humanoid to understand and interpret the data received by its sensors. Beyond that, it's all programming and imagination.

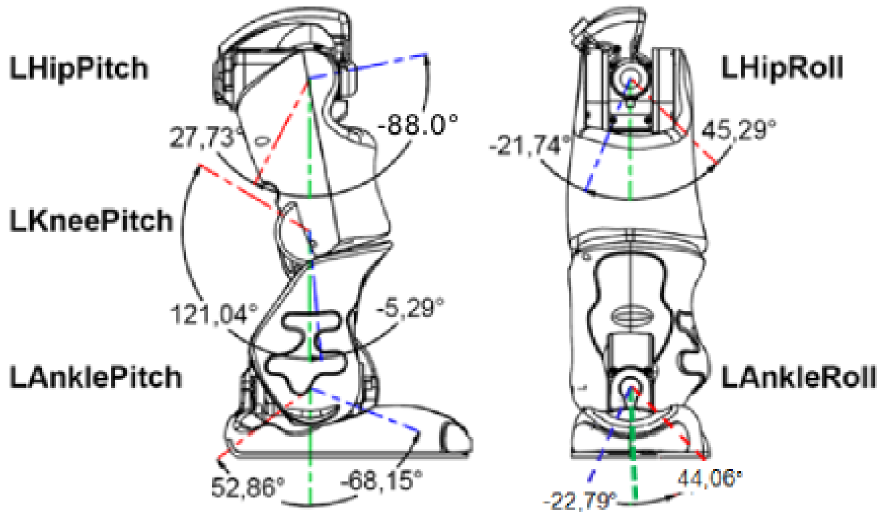


Figure 4.3: NAO left leg angle limits. Picture is taken from [2]

NAO's motion module is based on generalized inverse kinematics, which handles Cartesian coordinates, joint control, balance, redundancy, and task priority. This means that when asking NAO to extend its arm, it bends over because its arms and leg joints are taken into account. NAO will stop its movement to maintain balance.

The *Fall Manager* protects NAO when it falls. Its main function is to detect when NAO's Center of Mass (CoM) shifts outside the support polygon. The support polygon is determined by the position of the foot or feet in contact with the ground. When a fall is detected, all motion tasks are killed and, depending on the direction, NAO's arms assume protective positioning, the CoM is lowered, and robot stiffness is reduced to zero.

NAO currently supports Wi-Fi (bgn) and Ethernet, which are currently the most widespread network communication protocols. In addition, infrared transceivers in his eyes allow connection to objects in the environment. NAO is compatible with the IEE 802.11b/g/n Wi-Fi standard and can be used on both WPA and

WEP networks, making it possible to connect him to most home and office networks. NAO's OS supports both Ethernet and Wi-Fi connections and requires no Wi-Fi setup other than entering the password.

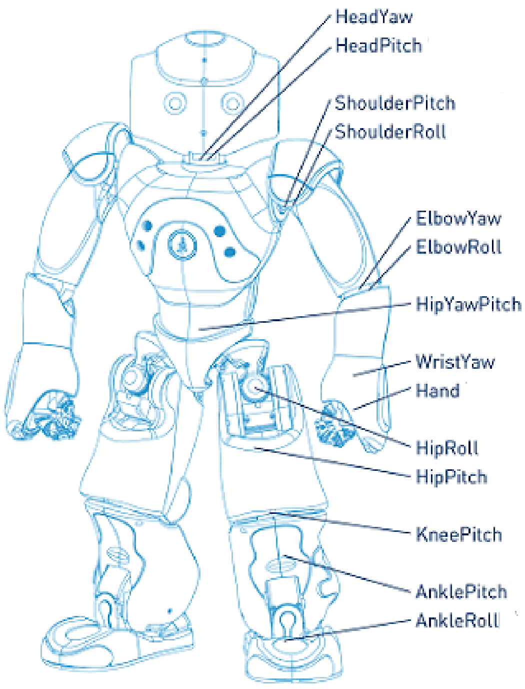


Figure 4.4: Position of motors. Picture is taken from [2]

In Figure 4.3 are shown angle limits of NAO left leg which were considered for implementation of controlling software. Human joint angles are adapted to robot capabilities making a proportion between ranges. Even if in this case only knee joint is used, the same approach has been used for each joint of robot in case of extension to whole body applications.

4.2 ROS

The Robot Operating System is a flexible framework for writing robot software. It was born at Stanford University in the mid-2000s thanks to researchers and students involved in various robotic programs. The idea was to develop a platform to share tools, libraries and competencies to simplify the task of creating robust and complex robot behavior through a wide kind of environments. The robotics research community has to face to a lot of challenges and issues that a single Laboratory or University could not deal with by its own. Open-ended collaboration among groups, laboratories and peoples, allow to unify strengths of each one providing to users the latest solutions. In 2007, Willow Garage, a nearby visionary robotics incubator, provided significant resources to extend these concepts much further and created well-tested implementations. Successively of the creation of a ROS core, researchers started to share their software packages through the permissive BSD open-source license. With the "federated" model, source packages are not placed all on the same servers. Developers keep projects on their servers maintaining full ownership and control on them until they choose to make code publicly available. ROS is widely used in universities, industry and by enthusiasts. It is available for various Linux distributions and for OSX; although it could run on Windows operating system it has not yet been implemented a dedicated version. On July 2014 has been released ROS Indigo Igloo, the eight release of the framework.

4.2.1 ROS concepts

As the name suggests ROS is a real operating system which provides hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. Here are presented some basic structures of the framework constituting one of its strengths [4].

Packages Packages are the main unit for organizing software in ROS. A package may contain ROS runtime processes (nodes), a ROS-dependent library, datasets, configuration files, or anything else that is usefully organized together. Packages are the most atomic build item and release item in ROS. Meaning that the most granular thing you can build and release is a package.

Nodes Nodes are processes that perform computation. ROS is designed to be modular at a fine-grained scale; a robot control system usually comprises many nodes. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one Node provides a graphical view of the system, and so on.

Master The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services. The Parameter Server allows data to be stored by key in a central location. It is currently part of the Master.

Messages Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).

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

There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. In general, publishers and subscribers are not aware of each others' existence. The idea is to decouple the production of information from its consumption. Logically, one can think of a topic as a strongly typed message bus. Each bus has a name, and anyone can connect to the bus to send or receive messages as long as they are the right type.

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

4.2.2 NAO packages

Even if NAO is a commercial product with its native software, there have been developed libraries and packages to connect to the robot under ROS. Some stacks are specific for this robot, while other are generally for humanoid robots. The software has been developed at the Humanoid Robot Lab[2] of Friburgo University and is composed of *nao_robot*, *nao_common*, *humanoid_msg* and *humanoid_navigation*, where the first two allow to control and simulate the NAO, while the last ones regard all humanoid robots.

4.2.3 Starting NAO

In this research it has been used the distribution for Linux, specifically Ubuntu version 12.04 LTS. Commands that must be run through a shell are listed below. For a complete description of set-up see the wiki [5].

Communication is done through TCP/IP so the IP address of the robot must be set. After switching on the robot tells the address when it is ready. Here is used a random example.

```
export NAO_IP=192.168.20.220
```

Start operating system of NAO and connect to it. $NAOQIPATH is the path of where Aldebaran's drivers are installed.

```
$ $NAOQIPATH/naoqi
```

Start drivers of robot and activate sensors.

[2]http://hrl.informatik.uni-freiburg.de/

```
$ roslaunch nao_driver nao_driver.launch
```

Make all the joints stiff, i.e. you won't be able to move the robot joints by force of your hands.

```
$ rosservice call /body_stiffness/enable "{}"
```

Launch node that allow to send to robot positions of a pre-built library.

```
$ roslaunch nao_pose pose_manager.launch
```

Make robot hold a certain position. In this case, is called the pose "init".

```
$ rosrun nao_pose execute_pose.py init
```

Now NAO is ready to perform tasks.

4.3 Simulating generation of signals

EMG signals were available in a file format since they were used in the offline processing described on Chapter 2 and 3. The scope of the thesis is to develop a framework which works in a real-time context therefore has been developed a ROS node which simulate generation of signals as if they were instantly acquired. The node, named *emg_generator*, reads from a configuration file (see Listing 4.5) the path where the file containing signals is stored and publishes a message with EMG values at a frequency rate equal to the acquisition one. Since for the wavelet transform only three channels were used, here the message contains only three variables, one per channel.

Listing 4.1: emg.msg

```
#value of emg signal read from a person to be read
and computed by GMM
#joint_name is the name of the joint which we want
to be corresponding to the emg signal
#value is the motion capture value used for
correctness tests.

string joint_name
float64 value
float64 ch1
float64 ch2
float64 ch3
```

When the node starts, it reads values of parameters and then publishes a message for all the values in file.

```
ros::init(argc, argv, "emg_generator");
ros::NodeHandle n;
n.param(ros::this_node::getName() + "/generator/rate", rate, 1000.0);
n.param(ros::this_node::getName() + "/generator/signalPath", path, std::string("
    "));
n.param(ros::this_node::getName() + "/generator/n_repeat", n_repeat, 1);
std::cout<< "path: "<<path;
ros::Publisher emg_pub_ = n.advertise<nao_kick::emg>("emg_value", 10);

Eigen::MatrixXd emgSignal;
magic_folder::MatrixOperations::loadFromFile(path,emgSignal);
int i = 0;
ros::Rate r(rate);
while(ros::ok() && i < std::min((int) emgSignal.cols(),2000*n_repeat))
{
        nao_kick::emg emg_msg;
        emg_msg.joint_name = "RKneePitch";
        emg_msg.ch1 = emgSignal(1,i);
        emg_msg.ch2 = emgSignal(2,i);
        emg_msg.ch3 = emgSignal(3,i);

    //for test using, add effective angle value
    emg_msg.value = emgSignal(9,i);
        emg_pub_.publish(emg_msg);
        ROS_INFO("Published");
        r.sleep();
        ++i;
}
```

When there are no more values to be published it sends a message with fake values to tell listening node that the signal is finished.

```
nao_kick::emg emg_msg;
emg_msg.joint_name = "RKneePitch";
emg_msg.ch1 = 99999;
emg_msg.ch2 = 99999;
emg_msg.ch3 = 99999;
emg_pub_.publish(emg_msg);

ROS_INFO("Finished");
ros::shutdown();
return 0;
```

4.4 Motion of the robot

Node which listen to EMG messages and connect to robot is named *model2motion_wlt*. It implements two functionalities:

- load predefined trajectories from files and publish messages to robot making it performing tasks;
- listen to EMG messages, compute wavelet transform and then publish to robot bending angle of joint selected.

First functionality is used principally to test motion of the robot and to find any issue in connection and communication. Second functionality is the one to focus on as it allows to command the NAO according to sEMG signals.

Listing 4.2: model2motion class definition

```
/** ***********************************************************************
 ** Type definitions
 ***********************************************************************/
typedef gml::GaussianComponent<Eigen::Matrix<double, DIM_POINTS, 1> > Component;
typedef gml::GaussianMixture<Component, NUM_COMPONENTS> Mixture;

typedef Mixture::ComponentPtr ComponentPtr;
typedef std::shared_ptr<Mixture> MixturePtr;

typedef Component::Point Point;
typedef Component::CovarianceMatrix CovarianceMatrix;

typedef gml::GaussianMixture<Component,Eigen::Dynamic> DynamicMixture;
typedef std::shared_ptr<DynamicMixture> DynamicMixturePtr;



/** ***********************************************************************
 ** Class definition
 ***********************************************************************/
class Model2Motion_wlt {
public:

    /**
      * @brief create an instance of the class
      * @param[in] path_dir - Path to which load trajectories
      * @param[in] path_model - Path to which load GMM mixture
      * @param[in] task_name - Name of the process
      * @param[in] n_repeat - Number of times which shall be repat trajectories
      * @param[in] start_index - Index from which start trajectories
      * @param[in] end_index - Index to which end trajectories
      * @param[in] rate - Rate of execution
      * @param[in] joint_names - Joint names of which positions shall be
        comunicated to robot
      * @param[in] num_channels - number of EMG channels used for motion
      */
    Model2Motion_wlt(std::string path_dir = "",
                                        std::string path_model = "",
                                        std::string task_name = "",
                                        std::string wavelet = "db2",
                                        std::string synth = "MAV",
                                        int n_repeat = 1,
                                        int num_train = 10,
                                        int start_index = 0,
                                        int end_index = -1,
                                        double rate = 30.0,
                                        std::vector<std::string> joint_names =
                                            std::vector<std::string>(),
                                        int num_channels = 3);

    /**
        * @brief Computes wavelet transform on emg signals and then calculate
          synthesis measure
        * @param[in] signal - emg signal
        * @param[in] level - level of wavelet decomposition
        * @param[in] wavelet_function - type of wavelet function on which is
          based wavelet trasform
        * @param[in] synthesis_measure - type of sysnthesis measure to use
          between "MAV" (Mean Average Value), "RMS" (Rooted Mean Square), "VAR
          " (Standard deviation)
        * @returns Knee bending angle in radiants.
        */
    double computeFeature(std::vector<double> signal, int level);

    /**
        * @brief Uses GMR to compute knee bending angle
        * @returns Knee bending angle in radiants.
        */
    double computeRegression();

        /**
        * @brief project angles according to robot capabilities
        * @param[in] joints - vector of joint names to move
        * @param[in] angles - vector of angles corresponding to joints to move
        */
    void fitLimits(std::vector<std::string>& joints, std::vector<float>& angles)
        ;

    /**
      * @brief load precomputed trajectories to be executed
      */
    int loadTrajectories();

    /**
      * @brief load limits of NAO joint and human motion possibilities
      */
        int loadLimits(ros::NodeHandle& node);

    /**
      * @brief load means, covariances and wieghts to assemble mixture of the
        Gaussian Mixture Model
      */
        int loadGMMModel();

    /**
      * @brief set pose of robot from which start movement
      * @param[in] first_time - set true if it is called for the first time in
        the application.
      * Setting to false allows to force only knee joints position leaving the
        others as they are
      */
    int setInitialPose(bool first_time = true);

    /**
      * @brief set a default pose of robot from which start movement
      */
        int setStartingPose();

    /**
      * @brief make robot to compute all the trajectories loaded
      */
        int drawTrajectories();

    /**
      * @brief make robot to compute the i-th trajectory
      * @param[in] i - index of trajectory to execute
      */
        int drawTrajectory(uint i);

    /**
      * @brief Set wether or not to project human joint angles to robot ones
      * @param[in] value - Set true to do projection.
      */
        void setProjectAngle(bool value);

    /**
      * @returns value of parameter which tells wether to project human joint
        angles to robot ones or not
      */
        bool getProjectAngle();

    /**
      * @brief Execute robot motion according to EMG values of messages from
        EmgGenerator.
```

```
     * Reads from buffer values of EMGs, calls computeRegression to estimate
       joint angle,
     * then sends to robot the position.
     * Position is computed as result of weighted moving average of last 20
       values
     */
      void modelToMotion();


private:
        std::string
        /** @brief Path of precomputed trajectories */
                path_dir_,
        /** @brief Path containing Gaussian Mixture Model parameters*/
                path_model_,
        /** @brief Name of application */
                task_name_,
                /** @brief Synthesis measure for decomposition coefficients */
                synth_measure_,
                /** @brief Mother wavelet name to use for wavelet transform */
                wavelet_func_;

        int
        /** @brief Number of times to repeat each trajectory*/
                n_repeat_,
                /** @brief Number of repetitions used for training. Needed for
                    model loading*/
                num_train_,
        /** @brief  Index from which start trajectories */
                start_index_,
        /** @brief  Index from which end trajectories */
                end_index_,
                /** @brief  number of channels used for motion */
                num_channels_;

        double
        /** @brief Rate of execution */
                rate_,
        /** @brief Calculated time between two consecutive changes of position
            of trajectories
          * t = 1 / rate
          */
                time_interval_;

        bool
        /** @brief parameter which tells wether to project human joint angles to
             robot ones or not */
                project_angle;

        std::vector<std::string>
        /** @brief Name of joints to move */
                joint_names_;

        std::vector<Eigen::MatrixXd>
        /** @brief Loaded trajectories to perform */
                trajectory_set_;

        std::vector<Eigen::Vector4d>
        /** @brief Loaded limits of human and robot motion possibilities
          * For each joint firss two values of Eigen vector are human limits and
               other two are ones of robot.
          * Order of joints is: HeadPitch, HeadYaw, LShoulderPitch,
              LShoulderRoll, LElbowRoll, LElbowYaw, LWristYaw,
          * LHand, LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch,
              LAnkleRoll, RHipYawPitch, RHipRoll, RHipPitch,
          * RKneePitch, RAnklePitch, RAnkleRoll, RShoulderPitch, RShoulderRoll,
              RElbowRoll, RElbowYaw, RWristYaw, RHand.
          */
```

```
                    limits_;

        MixturePtr
                    /** @brief mixture of gaussians for regression*/
                    mixture;

        gml::GaussianRegression<Component, NUM_COMPONENTS>
                    /** @brief class for computation of refression*/
                    gmr;

        ros::Publisher
                    /** @brief ROS publisher for robot messages*/
                    naoPublisher;
};
```

Since in these trials the performed actions are kicks or ankle movements, the robot is made to assume a proper and stable pose. Function *setInitialPose* calls first the default pose *"sit relax"* and then bends legs. Position of arms hold up the robot while moving, like in Figure 4.5.

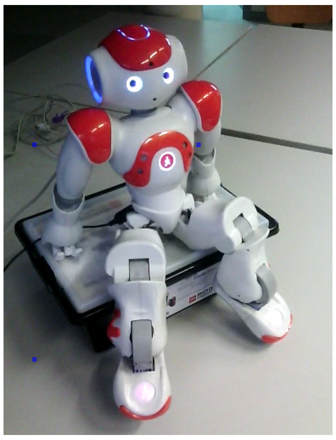


Figure 4.5: Stable pose during motion.

Hence, GMM model is loaded, a thread to listen for EMG signal is created and core function *modelToMotion* called. Every time a new input value is received by the callback the new portion of signal is elaborated through function *computeRegression* which returns the corresponding angle. Due to imperfections of estimation, values resulting from regression must be managed moreover in order to eliminate noise and oscillations. The position sent to the robot is the mean of the last twenty angles computed. If the parameter *projects_angle* is set to true the angle value is proportioned basing on joint limits (see Section 4.5 and 4.6). Bending angle is sent to robot with a ROS message of type *JointAnglesWithSpeed* of which an example is showed in listing 4.4. Speed is set to maximum value to avoid wasteful delays.

Listing 4.3: JointAnglesWithSpeed message definition

```
Header header

# A list of joint names, corresponding to their names in the Nao docs.
# This must be either the same lenght of joint_angles or 1 if it's a
# keyword such as 'Body' (for all angles)
string[] joint_names
float32[] joint_angles

#fraction of max joint velocity [0:1]
float32 speed

# Absolute angle(=0, default) or relative change
uint8 relative
```

Listing 4.4: ROS message to send NAO joints values

```
//publish message to NAO
nao_msgs::JointAnglesWithSpeed nao_trajectory;
nao_trajectory.joint_angles.clear();
nao_trajectory.joint_names.clear();
std::vector<std::string> jns;
jns.push_back("LKneePitch");
nao_trajectory.joint_names = jns;
std::vector<float> tmp;
tmp.push_back(angle_mean);
nao_trajectory.joint_angles = tmp;
nao_trajectory.speed = 1;
naoPublisher.publish(nao_trajectory);
```

computeRegression function uses the open source library wavelib-0.4.0.0 to perform wavelet decomposition which can be downloaded at [6].

4.5 Projection of angles

Humanoid robots are built to resemble human beings but mechanical components like motors and materials cause some limits to behavior replication. Human and NAO capabilities are different more or less depending on the joint considered. For example the motion of the knee joint is almost identical and the only different is about the minimum and maximum reachable angles of degrees. For the motion of the arm instead there is not a precise correspondence since gears do not replicate perfectly human fluency. While the second case is a problem of kinematics the first one can be easily solved with a set of parameters that let know the model which are values allowed. When the specific parameter tells to (see Section 4.6) a YAML Ain't Markup Language (YAML) file that contains NAO and humans joint limits is loaded. Below the portion of file about left leg.

```
LHipRoll:
   Min: -0.379472  #deg -21.7
   Max: 0.790477   #deg 45.3
   H_Min: -0.46632 #deg -26.8
   H_Max: 0.46632  #deg 26.8

LHipPitch:
   Min: -1.773910  #deg -88.0
   Max: 0.484090   #deg 27.7
   H_Min: 0.000001 #deg 0.0
   H_Max: 2.02710  #deg 116.5

LKneePitch:
   Min: -0.092346  #deg -5.3
   Max: 2.112530   #deg 121.0
   H_Min: 0.000001 #deg 0.0
```

```
    H_Max: 2.06016   #deg 118.4

LAnklePitch:
    Min: −1.189520  #deg −68.1
    Max: 0.922747   #deg 52.8
    H_Min: −0.14094 #deg −8.1
    H_Max: 0.62814  #deg 36.1

LAnkleRoll:
    Min: −0.397880  #deg −22.8
    Max: 0.769000   #deg 44.0
    H_Min: −0.397880 #deg −22.8
    H_Max: 0.769000 #deg 44.0
```

Bounds for NAO were retrieved from Aldebaran documentation [2] while description of human capabilities was retrieved at NASA site section on anthropometry and biomechanics [3]. Figure 4.4 shows portion about left leg taken from documentation. File contains joint limits of the whole body even if they are not all used. In the configuration file (Section 4.6) is possible to set which joints are used and then which limits are necessary. *Min* and *Max* are bounds relative to he robot, while *H_Min* and *H_Max* are relative to humans. Values are in radians. Once computed the angle to send to robot it is reported to humans limits, namely if it is lower than *H_Min* it is set to *H_Min* and viceversa: if it is greater than *H_Max* it is set to *H_Max*. In any case angle computed by the regression step is proportioned as explained in Equation 4.1.

$$angle' = \frac{angle}{H\_Min(H\_Max)} * Min(Max) \tag{4.1}$$

4.6 Configuration file

Parameters for both nodes *model2motion* and *emg_generator* can be set through the YAML file *test.yaml* 4.5. First part regards the first node while parameters preceded by the term "generator" concern simulation of generation of EMG signals. Meaning of parameters is:

task_name name of the task performed;

path_dir path of the directory of the trajectories to be loaded and performed (for usage without EMG);

path_model path of the directory of GMM model description files to be loaded (means and covariances);

n_train number of repetitions used for training the GMM model

save_path path of the directory which to save estimated time of processing (filename included);

n_repeat number of times to repeat trajectories (for usage without EMG);

n_channels number of EMG channels used in the model;

start_index number of first trajectory to perform (for usage without EMG);

end_index number of last trajectory to perform (for usage without EMG);

rate frequency rate to read EMG messages;

joint_names list of joint names involved in the trajectories (for usage without EMG);

projects_angle set whether to project or not human bending angle to robot capabilities;

wavelet_function mother wavelet to be used for computing wavelet transform;

synthesis_measure synthesis measure to be applied to wavelet coefficients (one of: "MAV","RMS","VAR");

generator: rate rate to which publish EMG values

generator: signalPath path of file containing EMG signals (filename included);

generator: n_repeat number of kicks performed in the EMG

Listing 4.5: example of test.yaml

```
# Task name
task_name: "emg_driven_knee_motion"

# Path directory of the trajectories to be loaded
path_dir: "/home/riccardo/Workspace/ros/catkin_ws/src/nao_unipd/nao_kick/task/"

# Path directory of the GMM model description to be loaded
path_model: "/home/riccardo/Dropbox/Matlab/EMG files/data_extraction/
    data_processed/Wavelet_db2_3_canali/MAV/gs/"
# Number of train of model
n_train: 60

#path and filename to which save delays
save_path: "/home/riccardo/Workspace/ros/catkin_ws/src/nao_unipd/nao_kick/times/
    gs_db2_MAV_times.txt"

# Number of repetitions
n_repeat: 1

# Number of channels used
n_channels: 3

# Start and end indexes
start_index: 0
end_index: 3

# Framerate
rate: 1000

# Names of joints used
#joint_names: ["HeadPitch", "HeadYaw", "LShoulderPitch", "LShoulderRoll", "
    LElbowRoll", "LElbowYaw", "LWristYaw", "LHand", "LHipYawPitch", "LHipRoll",
    "LHipPitch" , "LKneePitch", "LAnklePitch", "LAnkleRoll", "RHipYawPitch", "
    RHipRoll", "RHipPitch" , "RKneePitch", "RAnklePitch", "RAnkleRoll", "
    RShoulderPitch", "RShoulderRoll", "RElbowRoll", "RElbowYaw", "RWristYaw", "
    RHand"]
```

```
joint_names: ["LKneePitch"]

#when task correspond to human being movement, set true if adapt angle to robot
    capabilities
projects_angle: true

#parameters of wavelet transform
wavelet_function: "db2"
synthesis_measure: "MAV"

#generation of emg signal from file
generator:
   rate: 1000
   signalPath: "/home/riccardo/Dropbox/Matlab/EMG files/data_extraction/
       Original_kicks/gs_original_63.txt"
   n_repeat: 1
```

4.7 Launch files

Once NAO is ready to perform actions (commands of Section 4.2.3) node *model2motion_wlt* can be run through the following ROS command:

```
$ roslaunch nao_kick model2motion.launch
```

When robot is in position can be started simulation of EMG with:

```
$ roslaunch nao_kick emg_generator.launch
```

First *.launch* file runs the *main* contained in *model2motion_wlt.cpp* while the second runs *main* of *emgGenerator.cpp*. Both contains instruction to load test.yaml as it has needed values of parameters.

Listing 4.6: model2motion.launch

```
<launch>
  <node pkg="nao_kick" type="model2motion_wlt" name="model2motion_wlt"  output="
      screen">
    <rosparam command="load" file="$(find nao_kick)/config/test.yaml" />$
    <rosparam command="load" file="$(find nao_kick)/config/limits.yaml" />$
  </node>
</launch>
```

Listing 4.7: emgGenerator.launch

```
<launch>
  <node pkg="nao_kick" type="emgGenerator" name="emgGenerator"  output="screen">
        <rosparam command="load" file="$(find nao_kick)/config/test.yaml" />$
  </node>
</launch>
```

Figure 4.7 shows the chart produced with ROS *rqtgraph* in which are drawn all node involved during execution of the framework and the interactions between them (arrows).

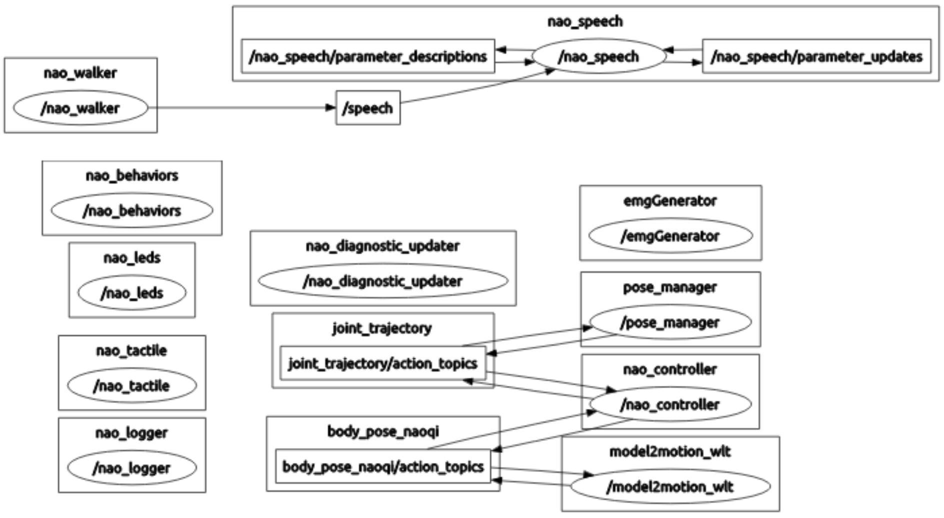


Figure 4.6: Graph of active nodes during execution of the framework.

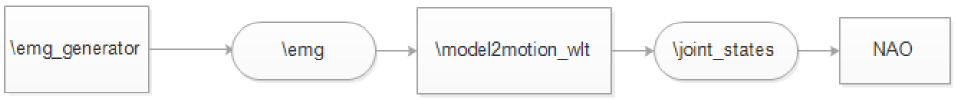


Figure 4.7: Simplified structure of main nodes and messages involved.

4.8 Performances

Elaboration was made with a Intel® 64-bit computer with i3 quad core CPU of 2.13 GHz and 4 GB of RAM. Computation of wavelet transform, feature extraction and regression have been tested for each subject.

As highlighted by the collected data on Table 4.1, the software is able to compute the pose messages for the robot in about 2.4 mS, which means a theoretical rate of 416 Hz. Anyway *emg_generator* publishes signal values at a rate slightly slower than 1000 Hz, and ROS framework uses a little portion of computing power for internal tasks, so effective rate measured by ROS is 240 Hz. However, even if considering a maximum threshold of 3 mS between one message and the successive, elaboration grants large space for other stuff satisfying the threshold of 100 mS indicated by Arvetti et al. [8]. In their study Arvetti et al. set threshold of 100 mS because they had to handle a fixed time of 200 mS to retrieve signals due to hardware constraints, but their empirical threshold for a tolerable delay by the user was 300 mS. In our case there was not a relevant delay due to signal acquisition since it was simulated, but even if it was, only set-up time would have been affected. Message sending rate shall be adapted to destination device, so for the NAO robot, the rate has been lowered to 50 Hz. The resulting robot movement is smooth and quite consistent with respect to the human motion.

Step	Method	Time(μs)
Wavelet Transform	db2	581.3955
	db4	1827.9829
	db6	1596.3840
	db7	2322.7949
Feature extraction	MAV	0.8448
	RMS	2.2616
	VAR	2.7085
Regression	-	1774.6145
Angle remapping	-	7.4357
Minimum sum	db2 + MAV	2408.9559
Maximum sum	db7 + VAR	4100.1179

Table 4.1: Analysis of the computational time (μs) needed from the framework at each phase.

Channel loss. The loss of one or more channels due to malfunctions is a concrete possibility so the framework was tested in such a situation. It is observed that, performing kicks, the loss of first channel do not affect general motion but only reduces fluency. Same behavior occurs with the loss of the third channel, meanwhile when the loss regards the second channel the motion heavily worsens. If loss of channels grows to two the framework is not able to reconstruct the movement and robot behavior is not reasonable. These considerations are purely empirical and we did no extensive study on the amount of information lost in each case.

Chapter 5

Conclusions

5.1 Discussion

This study proposed a method to estimate a single-joint angle by means of Surface Electromyography signals for online purposes. Classical time-domain and frequency-domain techniques of features extraction have been tested with poor results, due to non-stationary nature of signals. A method which can locate information both in time and frequency was necessary. Thence wavelet transform has been chosen as technique for signal analysis, thanks to wide success in similar applications reported in literature. Chowdhury et al. , Elamvazuthi et al. and Phinyomark et al. showed that, when working with EMG signals, the wavelet function should be selected from ones of Daubechies family. Available channels of sEMGs were investigated to find their factual participation to motion. Muscles which gave more contribution in performing kicks were *Rectus femoris*, *Vastus lateralis* and *Vastus medialis*. This result could be expected since those muscles are the main executors of the movement, while others are somehow passive. If another movement was performed, or just even if subjects were standing the role played by each muscle would had been kinda different. Anyway the fact that these muscles are all located above the knee makes this approach attractive even in the case of design of a prosthetic leg from the knee down.

The most informative channels were used to discover the most suitable *mother wavelet* and as Section 3.4 highlights we recommend to extract features applying wavelet transform with *Db2* function and to apply MAV computation of first level decomposition coefficients. Results show that it is possible to obtain a good model even using different combinations, for example with mother wavelet *Db7* and RMS; this ambiguity is due to the fact that the wavelet transform highlights specific properties of the signal depending on the wavelet function considered. Anyway we suggest to exclude use of variance, as it achieves generally a 10% lower effectiveness, respect to both other two measures.

This study therefore does not give us a final answer for EMG signals analysis but indicates a well workable approach to a problem that can be tackled in multiple ways.

Information extracted was summarized and compressed as much as possible in order to lighten the computation for use in real-time, but estimation could be improved by considering more levels of decomposition or taking in account old features when training the model.

Results reported on Table 4.1 enforce adoption of *Db2* mother wavelet whose computational time is lower than 600 μS while for others is more than 1500 μS. Similarly, MAV measure has a computational time of about 0.845 μS which is significantly lower respect to 2.261 μS needed by the others. Therefore, this combination is better than others because yields higher estimation performances and lower processing times.

Physiological information has been encoded through a Gaussian Mixture Model, while the joint angle related to a new sequence of unseen EMG data has been estimated from the model by using Gaussian Mixture Regression. Results exhibited an high accuracy in joint angle estimation (NMSE$> 0,8$) for all the subjects even if a margin for improvement still exists.

This study showed a possible way to command a single joint of a robot by means of sEMG which require a great effort for set-up due to GMM training phase but that turns out very fast at execution time making it suitable for real-time.

Thanks to the projection of human movement on the robot structure you get a good emulation of the behavior of the person, whose goodness, however, remains strongly dependent on mechanical properties.

Application of the framework to the robot showed a good level of modeling since the NAO was able to faithfully reproduce the movement corresponding to the EMG signals. Furthermore, in some cases of channel loss (see Section 4.8) the framework still works well confirming its robustness and showing a fairly good resistance to unexpected events (as might be the malfunction of a sensor).

5.2 Future work

Further work will be to improve robustness of the framework with an higher number of people and test it on other joints such to validate compatibility to whole body. This improvement could be achieved in various ways. One possibility is to increase the number of channels as to include all the players of the motion. To refine estimation could be analyzed greater levels of signal decomposition obtained through the wavelet transform as to include more information details.

In our study for each bending angle only one feature per channel was considered, but a better modeling could derive from the association of some past features in addition to the ones computed. Results obtained with experiments described in Section 3.7 encourage this approach. Attention should be paid to how many past features to use and how much in the past they should be. In this way we would implicitly treat a larger portion of signal enhancing the information examined.

Another aspect to consider is about the delay between the start of EMG signals and the start of actual human motion which is called Electro Mechanical

Delay (EMD). Existence of EMD is confirmed by various studies in literature but it is not quantified to a precise value. It seems to vary depending on the physical condition and age of the subjects, as well as the type of action, so there is still no general rule on how to handle it. Such improvements will make the framework proper for online controlling of whole humanoid robots and exoskeletons, as well as prosthesis or other biomechanical devices, by means of EMG signals.

Bibliography

[1] Aldebaran robotics. http://www.aldebaran.com/, .

[2] Aldebaran robotics documentation. doc.aldebaran.com, .

[3] Nasa: anthropometry and biomechanics. http://msis.jsc.nasa.gov/sections/section03.htm. Accessed: 2015-02-16.

[4] Ros wiki: concepts. http://wiki.ros.org/ROS/Concepts, . Accessed: 2015-03-06.

[5] Ros wiki: Nao tutorials: Getting-started. http://wiki.ros.org/nao/Tutorials/Getting-Started, . Accessed: 2015-03-06.

[6] Sourceforge wavelet2d. http://sourceforge.net/projects/wavelet2d/files/wavelib-0.4.0.0/. Accessed: 2015-03-06.

[7] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *2nd International Symposium on Information Theory*, pages 267–281, 1973.

[8] M. Arvetti, G. Gini, and M. Folgheraiter. Classification of EMG signals through wavelet analysis and neural networks for controlling an active hand prosthesis. In *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference on*, pages 531–536. doi: 10.1109/ICORR.2007.4428476.

[9] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6): 2743–2760, 1998.

[10] Adrian DC Chan and Kevin B Englehart. Continuous myoelectric control for powered prostheses using hidden markov models. *Biomedical Engineering, IEEE Transactions on*, 52(1):121–124, 2005.

[11] Yang Chen, Xingang Zhao, and Jianda Han. Hierarchical projection regression for online estimation of elbow joint angle using EMG signals. *Neural Comput and Applic*, 23:1129–1138, 2013.

[12] Rubana H. Chowdhury, Mamun B. I. Reaz, Mohd Alauddin Bin Mohd Ali, Ashrif A. A. Bakar, Kalaivani Chellappan, and Tae G. Chang. Surface Electromyography Signal Processing and Classification Techniques. *Sensors*, 13 (9):12431–12466, 2013. ISSN 1424-8220. doi: 10.3390/s130912431.

[13] J. Chu and Y. J. Lee. Conjugate prior penalized learning of Gaussian mixture models for EMG pattern recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1093–1098, 2007.

[14] Liu Chun Lin. A Tutorial of the Wavelet Transform.

[15] M.V. Costa, L.A. Pereira, R.S. Oliveira, R.E. Pedro, T.V. Camata, T. Abrao, M.A.O.C. Brunetto, and L.R. Altimari. Fourier and wavelet spectral analysis of EMG signals in maximal constant load dynamic exercise. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 4622–4625, Aug 2010. doi: 10.1109/IEMBS.2010.5626474.

[16] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *Information Theory, IEEE Transactions on*, 36(5):961–1005, Sep 1990.

[17] Lokenath Debnath. *Wavelet transforms and their applications.* Springer, 2002.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[19] I. Elamvazuthi, G.A. Ling, K.A.R.K. Nurhanim, P. Vasant, and S. Parasuraman. Surface electromyography (sEMG) feature extraction based on Daubechies wavelets. In *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on*, pages 1492–1495, June 2013. doi: 10.1109/ICIEA.2013.6566603.

[20] Kevin. Englehart, B. Hudgins, P. A. Parker, and M. Stevenson. Classification of the Myoelectric Signal using Time-Frequency Based Representations. pages 431–438, Jul-Sep 1999.

[21] P Guglielminotti and R Merletti. Effect of electrode location on surface myoelectric signal variables: a simulation study. In *9th Int. Congress of ISEK*, volume 188, 1992.

[22] Jianda Han, Qichuan Ding, Anbin Xiong, and Xingang Zhao. A state space emg model for the estimation of continuous joint movements. 2014.

[23] Y Huang, KB Englehart, B Hudgins, and ADC Chan. Optimized gaussian mixture models for upper limb motion classification. In *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, volume 1, pages 72–75. IEEE, 2004.

[24] A.R. Ismail and S.S. Asfour. Continuous wavelet transform application to EMG signals during human gait. In *Signals, Systems amp; Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on*, volume 1, pages 325–329 vol.1, Nov 1998.

[25] NING JIANG and Dario Farina. Myoelectric control of upper limb prosthesis: current status, challenges and recent advances.

[26] Dinesh Kant Kumar, Nemuel D. Pah, and Alan Bradley. Wavelet Analysis of Surface Electromyography to Determine Muscle Fatigue. *IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING*, 11(4):400–6, Dec 2003.

[27] Heloyse Uliam Kuriki, Emanuelle Moraes Mello, Fabio Micolis De Azevedo, Luciana Sanae Ota Takahashi, Neri Alves, and Ruben De Faria Negrao Filho. *The relationship between electromyography and muscle force.* INTECH Open Access Publisher, 2012.

[28] T.D. Lalitharatne, Y. Hayashi, K. Teramoto, and K. Kiguchi. A study on effects of muscle fatigue on EMG-based control for human upper-limb power-assist. In *Information and Automation for Sustainability (ICIAfS), 2012 IEEE 6th International Conference on*, pages 124–128, Sept 2012.

[29] F. Laterza and G. Olmo. Analysis of EMG signals by means of the matched wavelet transform. *Electronics Letters*, 33(5):357–359, Feb 1997.

[30] Sukhan Lee and G.N. Saridis. The control of a prosthetic arm by EMG pattern recognition. *Automatic Control, IEEE Transactions on*, 29(4):290–302, Apr 1984.

[31] Claudio Loconsole, Stefano Dettori, Antonio Frisoli, Carlo Alberto Avizzano, and Massimo Bergamasco. An EMG-based approach for on-line predicted torque control in robotic-assisted rehabilitation. In *Haptics Symposium (HAPTICS), 2014 IEEE*, pages 181–186. IEEE, 2014.

[32] K. Mahaphonchaikul, D. Sueaseenak, C. Pintavirooj, M. Sangworasil, and S. Tungjitkusolmun. EMG signal feature extraction based on wavelet transform. In *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference on*, pages 327–331, May 2010.

[33] F. A. Mahdavi, S. A. Ahmad, M. H. Marhaban, R. Mohammad, and Akbarzadeh T. Surface Electromyography Feature Extraction Based on Wavelet Transform. *International Journal of Integrated Engineering*, 4(3):1–7, 2012.

[34] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):674–693, 1989.

[35] Kurt Manal, Roger V. Gonzalez, David G. Lloyd, and Thomas S. Buchanan. A real-time EMG-driven virtual arm. *Computers in Biology and Medicine*, 32(1):25 – 36, 2002.

[36] Stefano Michieletto, Luca Tonin1 Mauro Antonello, Roberto Bortoletto, Fabiola Spolaor, Enrico Pagello, and Emanuele Menegatti. Gmm-based single-joint angle estimation using emg signals.

[37] Mohammadreza Asghari Oskoei and Huosheng Hu. Myoelectric control systems—a survey. *Biomedical Signal Processing and Control*, 2(4):275–294, 2007.

[38] P Parker, K Englehart, and B Hudgins. Myoelectric signal processing for control of powered limb prostheses. *Journal of electromyography and kinesiology*, 16(6):541–548, 2006.

[39] A. Phinyomark, C. Limsakul, and P. Phukpattaranont. Application of Wavelet Analysis in EMG Feature Extraction for Pattern Classification. *Measurement Science Review*, 11(2):45–52, 2011. doi: 10.2478/v10048-011-0009-y.

[40] Adam Quotb, Yannick Bornat, and Sylvie Renaud. Wavelet transform for real-time detection of action potentials in neural signals. *Frontiers in neuroengineering*, 4, 2011.

[41] M.B.I. Reaz, M.S. Hussain, and F. Mohd-Yasin. EMG analysis using wavelet functions to determine muscle contraction. In *e-Health Networking, Applications and Services, 2006. HEALTHCOM 2006. 8th International Conference on*, pages 132–134, Aug 2006. doi: 10.1109/HEALTH.2006.246433.

[42] MBI Reaz, MS Hussain, and Faisal Mohd-Yasin. Techniques of emg signal analysis: detection, processing, classification and applications. *Biological procedures online*, 8(1):11–35, 2006.

[43] U. Sahin and F. Sahin. Pattern recognition with surface EMG signal based wavelet transformation. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 295–300, Oct 2012. doi: 10.1109/ICSMC.2012.6377717.

[44] M. Sartori, M. Reggiani, E. Pagello, and D.G. Lloyd. Modeling the Human Knee for Assistive Technologies. *Biomedical Engineering, IEEE Transactions on*, 59(9):2642–2649, Sept 2012.

[45] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[46] Lauren H Smith, Levi J Hargrove, Blair A Lock, and Todd A Kuiken. Determining the optimal window length for pattern recognition-based myoelectric control: balancing the competing effects of classification error and controller

delay. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 19(2):186–192, 2011.

[47] Yu Su, Mark H Fisher, Andrzej Wolczowski, Geoffrey Duncan Bell, David J Burn, and Robert X Gao. Towards an emg-controlled prosthetic hand using a 3-d electromagnetic positioning system. *Instrumentation and Measurement, IEEE Transactions on*, 56(1):178–186, 2007.

[48] Abdulhamit Subasi. Application of adaptive neuro-fuzzy inference system for epileptic seizure detection using wavelet feature extraction. *Computers in Biology and Medicine*, 37(2):227–244, 2007.

[49] Abdulhamit Subasi. EEG signal classification using wavelet feature extraction and a mixture of expert model. *Expert Systems with Applications*, 32 (4):1084–1093, 2007.

[50] G. Tsenov, A.H. Zeghbib, F. Palis, N. Shoylev, and V. Mladenov. Neural networks for online classification of hand and finger movements using surface emg signals. In *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*, pages 167–171. IEEE, 2006.

[51] JJ Villarejo, A Frizera, TF Bastos, and JF Sarmiento. Pattern recognition of hand movements with low density semg for prosthesis control purposes. In *Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.

[52] James S Walker. *A primer on wavelets and their scientific applications*. CRC press, 2008.

[53] C. S. Wallace and D. L. Dowe. Minimum message length and kolmogorov complexity. *The Computer Journal*, 42(4):270–283, 1999.

[54] Kevin R Wheeler and Charles C Jorgensen. Gestures as input: Neuroelectric joysticks and keyboards. *IEEE pervasive computing*, 2(2):56–61, 2003.

[55] Xu Yong, Xiaobei Jing, Yinlai Jiang, Hiroshi Yokoi, and Ryu Kato. Tendon drive finger mechanisms for an emg prosthetic hand with two motors. In *Biomedical Engineering and Informatics (BMEI), 2014 7th International Conference on*, pages 568–572. IEEE, 2014.

Ringraziamenti

Questi anni di studio all'università sono stati ricchi di scoperte, di approfondimenti, di sfide e di obbiettivi da raggiungere. Essi mi hanno permesso di acquisire nuove conoscenze, non solo in campo scientifico ma anche su me stesso. Tutto questo non sarebbe stato possibile senza le persone che hanno condiviso con me questo percorso, chi per un pezzo più lungo, chi per un tratto più breve.

Innanzitutto ringrazio la mia famiglia, i miei genitori e mia sorella, senza il cui continuo sostegno ed incoraggiamento non avrei potuto trovare le energie per portare a termine i miei studi.

Ringrazio il dott. Stefano Michieletto che mi ha permesso di approfondire un campo dell'ingegneria affascinante e stimolante, e per la pazienza, la disponibilità e l'aiuto fornitomi in questo lavoro. Ringrazio il prof. Pagello e il prof. Menegatti per avermi appassionato alle loro discipline facendomi scegliere di svolgere la tesi allo IAS-Lab. Grazie anche a tutti i dottorandi e i ricercatori del laboratorio di robotica che con il loro contributo mi hanno aiutato a superare i piccoli scogli, ed in particolar modo a Marco e Morris per il sostegno e la compagnia.

Questa tesi arriva al culmine di cinque anni di impegni e di fatiche che non sarebbero stati altrettanto proficui e spensierati senza i miei fidati compagni di gruppo Nicola e Simone che sono stati con me dall'inizio alla fine; Federico per l'allegria nelle lunghe giornate in laboratorio; e gli amici con cui ho condiviso tante lezioni e viaggi in treno: Alvise, Gianluca, Silvio e Martino. Un grande ringraziamento va anche a Francesco, Giulia, Martina, Elisa e tutti gli altri amici che dal di fuori con semplici gesti hanno manifestato il loro appoggio ed hanno creduto in me.

Infine, un ringraziamento speciale a Silvia, per avermi incoraggiato in tutti i miei sforzi, aver sostenuto le mie scelte ed essere sempre stata al mio fianco sia nei momenti brillanti che in quelli meno luminosi.