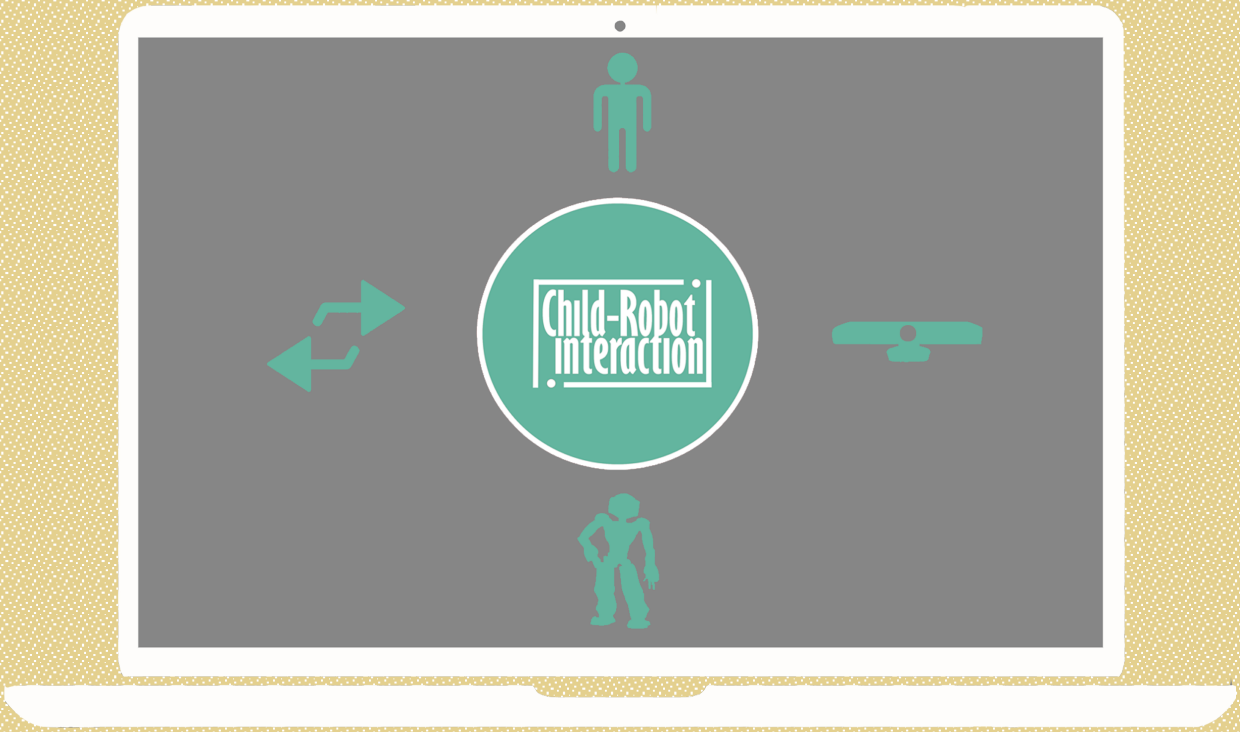


José Pedro da Silva Figueiredo Medeiros

Child-Robot interaction: Learning by Imitation using Motion Generalization.

September 2016

· U C ·

Universidade de Coimbra



Departamento de Engenharia Electrotécnica e de Computadores
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

A Dissertation
for Graduate Study in MSc Program
Master of Science in Electrical and Computer Engineering

Child-Robot interaction: Learning by Imitation using Motion Generalization.

José Pedro da Silva Figueiredo Medeiros

Research Developed Under Supervision of
Prof. Doutor Paulo José Monteiro Peixoto
and Prof. Doutor Diego Resende Faria

Jury
Prof. Doutor Rui Pedro Duarte Cortesão
Prof. Doutor Jorge Nuno de Almeida e Sousa Almada Lobo
Prof. Doutor Paulo José Monteiro Peixoto

September 2016

Work developed in the Institute of Systems and Robotics of the University of Coimbra.



It always seems impossible until it's done.
(Nelson Mandela)

Acknowledgements

In this text, I want to thank all the people without whom this journey would not have come to an end. First and foremost, I thank my whole family for always being there supporting me.

I would like to thank my advisors, Professor Paulo Peixoto, Professor Urbano Nunes and Professor Diego Faria, for everything, for the encouragement and for all the useful lessons.

I thank to my laboratory partners that always supported me in this work. More specifically, I would like to thank my colleague Rui Almeida who started as a colleague and became a close friend.

I would also like to thank ISR for hosting me, providing the necessary resources, conditions and personnel that allowed me to accomplish all the goals I worked for. This work has been supported by Fundação para a Ciência e Tecnologia (FCT), COMPETE and QREN programs, under the project "AMS-HMI12 - Assisted Mobility Supported by shared control and advanced Human Machine Interfaces" with reference RECI/EEI-AUT/0181/2012.

I would like to thank all my friends, I have no words for the support they gave me along this journey. I will thank them personally since I don't want to be unfair and miss anyone.

I have saved the last words to thank my girlfriend Diana that was there for me during the hard times.

Thank you all.

Abstract

The work developed here is included in a partnership between Instituto de Sistemas e Robótica (ISR) and Associação de Paralesia Cerebral de Coimbra (APCC). This partnership intends to develop several works in the child-robot interaction area, focusing children from the APCC association which suffer from cerebral palsy.

For robots to engage with human users in social interactions, the setup where they are included must be prepared with a set of different capabilities. In this work a system setup is proposed where some of this capabilities were implemented with the help of NAO robot. To assist the robot a framework composed by a computer and the kinect RGB-D depth sensor was used. The system was developed under Robot Operating System (ROS) which provided the necessary communications between the parts of the system.

The work intends to make the robot suited for playing didactic games with the children from APCC. To accomplish this, different skills were developed in the proposed system. The system is capable of detecting users, it is capable of learning movements taught by the user (from the arms), it is also able to reproduce the user arms movement in the robot, it can recognize movements and it can communicate trough voice commands.

The system skills were tested with adults and children users interacting with NAO robot. In the first part of the work each skill was developed independently. After all the skills being tested they were included together in a module. This module was tested on-the-fly were all the skills are used for a continuously interaction with the user. This module allows the user to switch between the imitation mode and the learning mode. In the imitation mode the robot will imitate the user movement. In the learning mode the user can teach new movements to the robot. In this mode if the system recognizes the user movement NAO will inform the user and reproduce the saved movement after. In case a movement is not recognized the robot will ask the user to repeat it some times so it can be learned and generalized.

This type of system is a growing system where other skills can always be added. It also works the

other way around, the skills of the work proposed here have the potential to be included in future works. Hopefully it will come to a point where it exists a system advanced enough suited to help children in this situations. Small steps like the proposed system are the way to accomplish it.

Keywords: Kinect Motion Learning, Child-Robot interaction, Kinect, NAO, ROS.

Resumo

Para que seja possível a interação entre robôs e humanos em contextos sociais, a criação de uma estrutura com capacidades sociais deve servir de suporte ao robô.

Neste trabalho uma estrutura do género é proposta. Nesta estrutura algumas destas capacidades sociais foram implementadas com a ajuda do robô NAO. A estrutura é composta ainda pelo sensor kinect RGB-De um computador.

Este trabalho está incluído numa parceria entre o Instituto de Sistemas e Robótica (ISR) e a Associação de Paralesia Cerebral de Coimbra (APCC). Esta parceria pretende desenvolver vários trabalhos na área de interação entr robôs e crianças, com um foco nas crianças da associação APCC que sofrem de paralisia cerebral.

As diferentes funcionalidades da estrutura foram testadas por utilizadores adultos e por uma criança. O sistema foi desenvolvido na plataforma *robot operating system* (ROS) que efectua as comunicações necessárias entre todos os componentes do sistema. O sensor kinect foi utilizado devido às suas capacidades de reconhecimento humano e rastreamento de esqueleto.

O sistema está preparado para realizar a aprendizagem e generalização de movimentos recorrendo ao sensor kinect na captura da informação do esqueleto do utilizador. Depois de o movimento ser generalizado o sistema está preparado para mapeá-lo no robô NAO, ficando depois preparado reproduzi-lo. Um módulo de classificação também foi implementado para detecção de movimento, no caso de um movimento ser reconhecido o robô poderá reproduzi-lo.

Foi desenvolvido ainda um módulo de imitação onde a parte superior do corpo poderá ser imitada pelo robô.

A fim de realizar as habilidades descritas, o sistema também está equipado com o reconhecimento de voz e, ao mesmo tempo, o robô NAO está equipado com um altifalante permitindo assim comunicar com o utilizador.

Este tipo de sistema é um sistema expansível onde outras habilidades podem sempre vir a ser adicionadas. O mesmo acontece no sentido contrário, as habilidades aqui desenvolvidas poderão ser incluídas em trabalhos futuros. É de esperar que seja alcançado o ponto em que exista um sistema avançado o suficiente para ajudar as crianças nessas situações, para alcançar esse objetivo é necessário pequenos passos como o sistema proposto.

Palavras-chave: Kinect, NAO, ROS

Contents

Acknowledgements **i**

Abstract **iii**

Resumo **v**

List of Figures **xi**

List of Tables **xiii**

List of Abbreviations **xv**

1 Introduction **1**

1.1 Motivation 1

1.2 Objectives 2

1.3 Implementations and main contributions 2

2 Background and State of the Art **5**

2.1 Background 5

2.1.1 Movement Generalization 5

2.1.1.1 Principal Component Analysis 5

2.1.1.2 Dynamic time warping 6

2.1.1.3 Gaussian Mixture Model with EM 8

2.1.1.4 Classification Algorithms 9
2.1.1.5 Support Vector Machine 9
2.1.1.6 Naive Bayes 13
2.1.1.7 Dynamic Bayesian Mixture Model 14
2.2 State of the Art 15
2.2.1 Robot Learning by Imitation 15
2.2.2 Child-Robot iteraction 16

3 Child-robot Social Interaction System Setup **19**
3.1 Hardware and Software 19
3.1.1 Microsoft-Kinect 20
3.1.2 NAO robot 21
3.1.3 Robot Operating System (**ROS**) 22
3.2 System Setup Overview 23

4 Motion Generalization and Classification **27**
4.1 Kinesthetic Learning 27
4.1.1 Data Acquisition 27
4.1.2 Movement Learning and Generalization 31
4.1.3 Movement Imitation 34
4.2 Kinect Gesture Learning 35
4.2.1 Kinect Data Acquisition 35
4.2.2 Movement Learning and Generalization 35
4.2.3 Movement Imitation 36
4.3 Classification 37

5 Experiments, Tests and Results **39**

5.1 Kinesthetic Learning 39
5.2 Kinect Gesture Learning 42
5.3 Movement Classification 45

6 Conclusions and Future Work **47**

Bibliography **49**

List of Figures

1.1 System setup components: red - Kinect RGB-D sensor; green - Computer; grey - NAO robot; blue - User. 3

2.1 SVM trained by two different classes. White circles are the class number 1 and the black circles represent class number 2. Image from [36] 10

2.2 Kernel function applied to space transformation, image adapted from [7]. 13

2.3 Dynamic Bayesian Mixture Model overview, image from [36]. 14

3.1 System SetUp Components 20

3.2 Microsoft-Kinect components [10] 20

3.3 NAO H25-V4 description [2] 22

3.4 OpenNI's kinematic model of the human body [6] 23

3.5 System setup overview 25

4.1 Motion Learning and Generalization - Model Overview 28

4.2 NAO H25 V4 arms joints information. (Image adapted from Aldebaran Documentation on NAO) 29

4.3 Upper body joints published by ROS using the kinect Sensor 36

4.4 Kinect joints mapping to NAO robot [28] 37

4.5 5 year old children performing the Byebye movement in the classification process. . . 38

5.1 Bolt movement kinesthetic learning from user 1 39

5.2 Bolt movement kinesthetic learning from user 2. 40

5.3 Principal component number 1 signature generated from 6 repetitions from the Byebye movement 41

5.4 Byebye movement signature - information from joint number 1:'RShoulderPitch'. . . 42

5.5 Byebye movement signature - information from joint number 4:'RElbowRoll' 42

5.6 Byebye movement motion generalization corresponding to the principal component number 1. 44

5.7 Example from the gesture learning, learning the bolt movement. 44

5.8 SVM classifier performance results 45

5.9 NB classifier performance results 45

List of Tables

3.1 Microsoft-Kinect specifications [10] 21

4.1 Dataset from the Kinesthetic Demonstrations 28

4.2 Table of symbols and abbreviations 30

4.3 15 body joints published description 35

4.4 Dataset from the Kinect Body Pose Acquisition 36

4.5 Dataset structure. 38

5.1 PCA Analysis, necessary number of principal components to obtain at least 99% ”come-back” to the original data. 40

5.2 PCA Analysis on byebye movement - necessary number of principal components to otbain at least 99% ”comeback” to the original data. 43

5.3 Confusion matrix obtained on the fly using DBMM classification method. Movements: 1 - ’Byebye’, 2 - ’Arms calibrate pose’, 3 - ’Bolt’, 4 - ’Clapping’, 5 - ’Cruize arms’, 6 - ’Arms to the top’, 7 - ’Arms What’, 8 - ’Airplane’. 46

List of Abbreviations

2D, 3D	Two Dimensional, Three Dimensional
DBMM	Dynamic Bayesian Mixture Model
DTW	Dynamic Time Warping
FCT	Fundação para a Ciência e Tecnologia
fps	frames per second
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
ISR	Institute of Systems and Robotics
NBC	Naive Bayes Classifier
PC	Principal Component
PCA	Principal Component Analysis
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue and Depth
ROS	Robot Operating System
SVM	Support Vector Machine

Chapter 1

Introduction

Works using robots for children therapy have been increasing and have proven they can be useful tools like it is showed in [12]. This kind of interaction demanded for the robots to have social and interactive skills.

The evolution in child-robot interaction, either if it is for therapy purposes or others, will increasingly require the robots to be more skilled. The proposed work tries to develop some of the necessary skills, which one day might be used in a robot fully capacitate for social interaction.

This dissertation explains the research done aiming to develop social skills for NAO robot to be able to interact with children users.

1.1 Motivation

Capacitating robots with social skills has been an object of study for some years now, despite that, there are still many situations where this type of robots were not studied or only recent advances were made. This is the case where children with cerebral palsy (CP) interact with robots for educational or therapeutic purposes. There is still some way to go until it is possible to fully use robots to interact with children who suffer from this condition.

Achieving fully social capacities in robots, for a common use is already a big challenge *per se*, and it becomes an even bigger challenge when applied to children with social difficulties. This is the case of children that suffer from CP, where robot-therapy can be applied (e.g. children attention engagement).

1.2 Objectives

The main objective of the research where this dissertation is based from is to develop a system capable of making NAO suited for social interacting with human users, including children. To accomplish this, the system framework needs to be equipped with different skills.

The robot should be suited to help in a therapeutic environment and can be used for engaging the children attention. It can make imitation or teaching and learning movements games with them. It is expected that it can help with reducing the children's stress. This implies that, as mentioned before, the framework is equipped whith the required skills fit for social interactions. The necessary steps to achieve such a goal encompass the following ones:

- Motion Learning and Generalization - the framework should have the skills to learn and generalize the user(s) movements.
- Motion Imitation - the robot should be able to imitate the generalized movements.
- Motion Classification - the framework needs to be capable of recognizing the movements.
- Testing the developed system in real situations.

1.3 Implementations and main contributions

This dissertation describes the work developed to achieve the proposed system. An overview on the system setup components can be seen in figure 1.1. During the research to reach this system some different approaches were studied. This document explains the algorithms, methods, software and hardware used in the final system. The implementations and main contributions are the following:

Child-robot Interaction System Setup (Chapter 3):

- Description on the hardware and software used.
- Fully explanation on how the system works.

Motion Learning and Classification(Chapter 4):

- Movement kinhestetic learning and generalization implementation based on the work developed in [16].

- Gesture Learning using kinect sensor - adapting the work [16] system to work on a shared learning environment. The new system changes the learning method from a kinhestetic learning to a live gesture learning.
- Movement Classification Module based on the work implemented in [23] [24] [36].

Experimental Results (Chapter 5):

- Motion signature using Kinhestetic learning.
- Motion signature using Gesture learning.
- Classification tests.
- On-the-fly testing of the system.

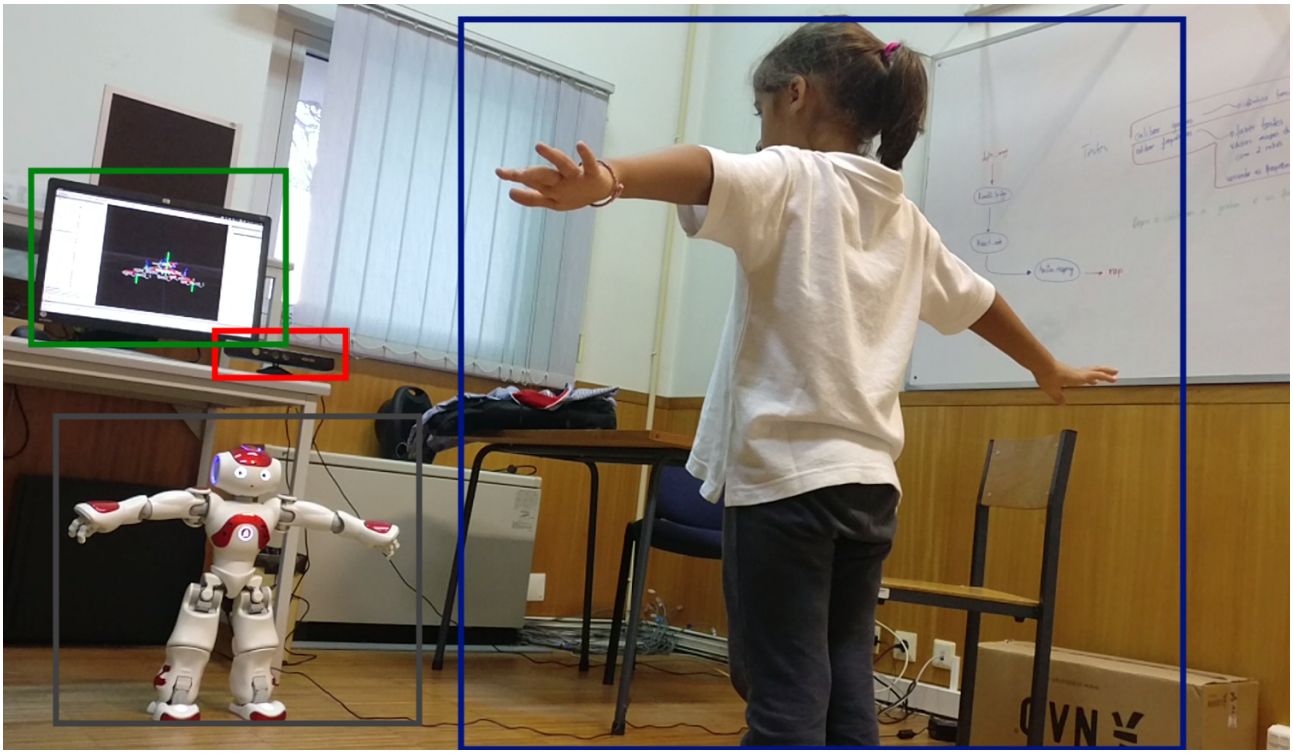


Figure 1.1: System setup components: red - Kinect RGB-D sensor; green - Computer; grey - NAO robot; blue - User.

Finally, chapter 6 provides the conclusions drawn and suggests some guidelines for future work.

Chapter 2

Background and State of the Art

This chapter describes the fundamental methodologies required to understand the work presented in this dissertation.

2.1 Background

In this section an explanation is given on the methods and algorithms that were used or adapted but not originally implemented in this work.

2.1.1 Movement Generalization

2.1.1.1 Principal Component Analysis

Principal component analysis is a statistical procedure that when applied to a set of data tries to identify uncorrelated variables. These variables are called principal components.

PCA was proposed in 1901 by Karl Pearson in [26]. The objective of the algorithm is to describe the data with the minimum number of principal components. If there is correlation in the data, the algorithm will lead to a reduction of the data dimensionality.

Given an input set of data defined by the matrix $X(m,n)$, the first step it to subtract the mean $\bar{X}$ and save it.

$$X' = X - \bar{X} \tag{2.1}$$

For the second step **PCA** will calculate the covariance matrix of X' and the projection o X' in the

latent space defined by $PC(m,n)$.

$$Q = cov\{X'\} \tag{2.2}$$

In the third step the matrix of the eigenvectors of L and a matrix with the corresponding eigenvalues λ are calculated. The eigenvectors are sorted by relevance in a matrix $L(m,n)$, and therefore the corresponding eigenvalues $\lambda(1,n)$ are ordered from high to low, and the sum of all equals 1.

The fourth step is where the reduction of dimensionality happens. Adding the eigenvalues provides the information on how many eigenvectors are needed for describing a certain percentage **(p)** of the original data X', pre-defined by the user. There will always be data loss, but normally it will be equal or lower than p. This procedure will define the number t of necessary vectors to reach the percentage p.

In the fifth step $W(m,t)$ is defined by the first t vectors from $L(m,n)$, the same logic is applied for the latent space $PC'(m,t)$.

For coming back to the original data it is just needed to compute the following step:

$$X_{new} = \bar{X} + PC' * W \tag{2.3}$$

Although X_{new} has slightly less information than the original matrix of input data X, it will still reduce drastically the computational resources.

2.1.1.2 Dynamic time warping

Dynamic time warp is an algorithm used for analysing the similarity between two linear vectors. **DTW** will try to find similarities between the vectors. Using this information it is possible for the algorithm to determine and map an optimal path between the sequences.

Given two temporal sequences:

$$X = \begin{bmatrix} x_1 & (...) & x_m \end{bmatrix}_{\{1,m\}}, k = (1...m) \qquad Y = \begin{bmatrix} y_1 & (...) & y_m \end{bmatrix}_{\{1,n\}}, l = (1...n) \tag{2.4}$$

A 2 dimensions matrix (Z) with the euclidean distance between pairs (X_k, Y_l) is computed in the first step:

$$Z = \begin{bmatrix} e_{(1,1)} & (...) & e_{(1,n)} \\ (...) & (...) & (...) \\ e_{(m,1)} & (...) & e_{(m,n)} \end{bmatrix}_{\{m,n\}} \tag{2.5}$$

In the second step for being able to determine the warping path a cost matrix (A) is computed. The cost matrix (Q) is built by determining the cost of all the possible warping paths.

- A path always start at the first cell (1,1) and end at cell (m,n).
- It is not possible to go back in time. This imposes a restriction that a path can only go forward. If, for example, the path is in cell (1,1) it can only go to the following cells:(1,2), (2,1) or (2,2).
- Each time the path moves to a certain cell (k,l) it will add all the previous accumulated cost to the euclidean distance from the pair $Z_{(k,l)}$. This value is then added to the respective cell in the cost matrix $Q_{(k,l)}$.

$$Q = \begin{bmatrix} q_{(1,1)} & (\ldots) & q_{(1,n)} \\ (\ldots) & (\ldots) & (\ldots) \\ q_{(m,1)} & (\ldots) & q_{(m,n)} \end{bmatrix}_{\{m,n\}} \tag{2.6}$$

In order to find the optimal path the algorithm will minimize the cost matrix.

- The optimal will start at the first last cell (m,n) and end at cell (1,1).
- It is only possible to go back in time. This imposes a restriction that a path can only go backward. If, for example, the path is in cell (m,n) it can only go to the following cells:(m,n-1), (m-1,n) or (m-1,n-1).
- Each time the path is in a certain cell (k,l) it will always move to the cell that minimizes the cost saving it. This next cell as the index from the cell that equals to the $min(Q_{(k,l-1)}, Q_{(k-1,l)} Q_{(k-1,l-1)})$.

$$P = \begin{bmatrix} [m,n] & (\ldots) & [1,1] \end{bmatrix}_{\{2,k\}} \tag{2.7}$$

In the last step the warped vectors (X_w,Y_w) are created by going trough all the indexes in the optimal path.

- If the the actual cell (k+1,l+1) and the next cell(k,l) in the optimal path decrease a value in both columns, the value in X_k is added to X_{wk} and the value in Y_l is added to Y_{wl}.
- If the next cell (k,l) and the actual cell (k,l+1) in the optimal path have the same value k, in the first column, the value in X_k is added to $X_{w(k+1)}$.
- If the next cell (k,l) and the actual cell (k+1,l) in the optimal path have the same value l, in the second column, the value in Y_l is added to $Y_{w(l+1)}$.

In the way the **DTW** performs the warped vectors (X_w,Y_w) will have equal or higher dimensions than the original sequences (X,Y).

2.1.1.3 Gaussian Mixture Model with EM

Gaussian mixture model is a probabilistic model that is used for data clustering. Each cluster is modelled with Gaussian distributions, meaning each value is defined by a mean and a covariance.

Each mixture component (index c) is defined by a mean u_c, a covariance σ_c and a "size" s_c.

The probability distribution of the data is defined as:

$$p(x) = \sum_c s_c \mathcal{N}(x; u_c, \sigma_c) \tag{2.8}$$

In the latent space a cluster (c) is selected:

$$p(z = c) = s_c \tag{2.9}$$

The probability of a data point x knowing c was selected is given by:

$$p(x|z = c) = \mathcal{N}(x; u_c, \sigma_c) \tag{2.10}$$

The expectation maximization algorithm will iteratively update the clusters in 2 steps:

- E-step, that is called the expectation step, will consider for each point x_i the mean u_c, covariance σ_c and "size" s_c fixed and it will compute a "responsibility" value r_{ic}. This value will measure the probability of x_i belonging to cluster c.

$$\frac{\sum_c r_c \mathcal{N}(x; u_c, \sigma_c)}{\sum_{c'} r_{c'} \mathcal{N}(x; u'_c, \sigma'_c)} \tag{2.11}$$

  In case x_i is not probable to be in cluster c it will have a small r_{ic} and if it is likely to be in cluster c it will have an high r_{ic}.

- M-step, that is called the maximization step, will fix r_{ic} and it will update the components(u_c,σ_c,s_c) of each cluster.

  Using the total responsibility (m_c)allocated to cluster c:

$$m_c = \sum r_{ic} \tag{2.12}$$

  the algorithm will update s_c by normalizing it to the total responsibility m from all the cluster.

$$s_c = \frac{m_c}{m} \tag{2.13}$$

s_c represents the data point probabilities assign to cluster c. The mean u_c will be updated by computing the weighted average of the data:

$$u_c = \frac{1}{m_c} \sum r_{ic} * x_i \tag{2.14}$$

The covariance σ_c will be updated by computing the weighted average of the data:

$$\sigma_c = \frac{1}{m_c} \sum r_{ic} * (x_i - u_c)^T (x_i - u_c) \tag{2.15}$$

For both the mean and the covariance if r_ic is small x_i will have a low influence. In the opposite situation if r_ic is large x_i will have a big influence in the cluster update.

In each step the log-likelihood of the model will increase:

$$\sigma_c = \sum log \sum r_{ic} * (x_i - u_c)^T (x_i - u_c)_i \tag{2.16}$$

The algorithm will iterate until it converges. Normally it will stop when a pre-defined log-likelihood is reached or when the log-likelihood as a small difference between iterations.

In the end **GMM** as distributed the clusters that hopefully will describe the sub-sets of data.

2.1.1.4 Classification Algorithms

2.1.1.5 Support Vector Machine

In classification Support Vector Machines (SVM) are a set of supervised learning methods commonly used. In 1963 the first SVM algorithm was created by Vladimir N. Vapnik and Alexey Ya. Chervonenkis. The present version, soft margin, was presented in [20] in 1995.

SVM is used for linearly separate binary sets. It will try to separate two different sets with a straight line. This is done by centering the line in the maximum distance between two sets of data as it can be seen in2.1. This line is also referenced to as hyperplane.

For determining the hyperplane the normal vector $\bar{w}$ to it must be calculated.

An unknown sample $\bar{s}$ can only belong to one of two classes:

1. if $\bar{w} \cdot \bar{s} + b \geq 0$, then $\bar{s}$ is from the first class

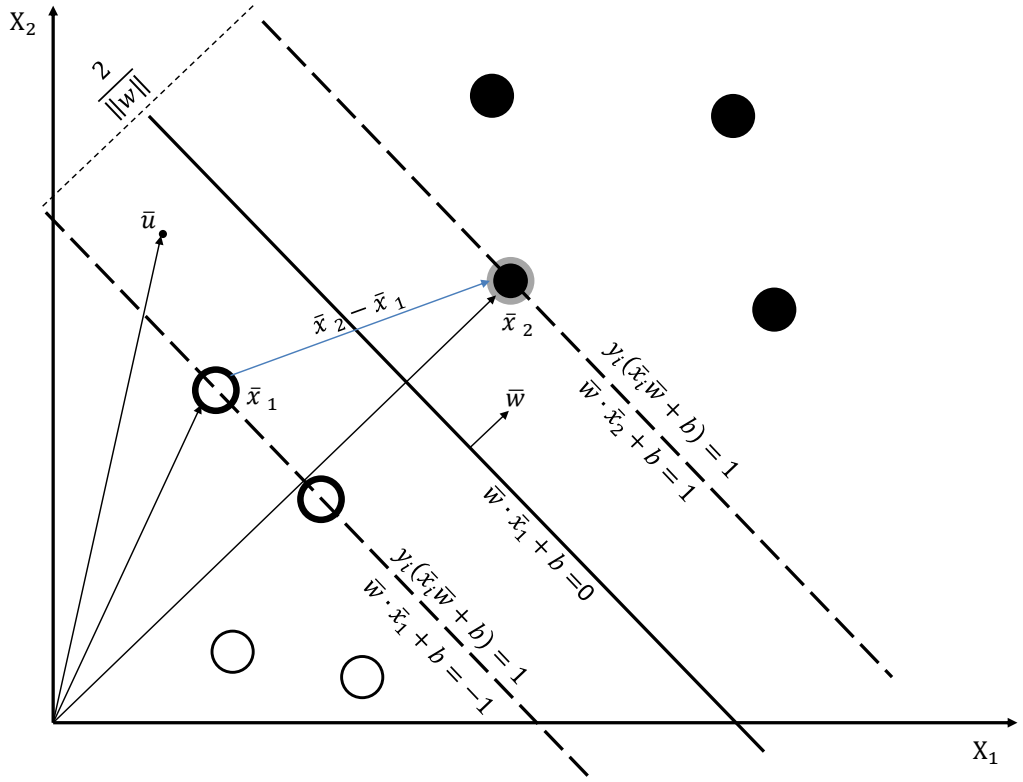


Figure 2.1: SVM trained by two different classes. White circles are the class number 1 and the black circles represent class number 2. Image from [36]

.

2. if $\bar{w}\cdot\bar{s}+b\leq 0$, then $\bar{s}$ is from the second class

(2.17)

Both the constant b and $\bar{w}$ need to be determined and the constraints in 2.18 must be met to accomplish this.

1. $\bar{w}\cdot\bar{x}_i+b\leq -1, \bar{x}_i$ belongs to the first class

2. $\bar{w}\cdot\bar{x}_i+b\geq 1, \bar{x}_i$ belongs to the second class

(2.18)

The variable y_i as to meet some requirements. When the samples belong to the first class $=> y_i = 1$ and when they belong to the second $=> y_i = -1$. The variable is then multiplied to (2.18) and it is obtained:

1. $y_i(\bar{w}\cdot\bar{x}_i+b)\geq 1$, for $\bar{x}_i$ from the first class

$$2.\ y_i(\bar{w}\cdot\bar{x}_i+b)\geq 1, \text{for } \bar{x}_i \text{ from the first class} \tag{2.19}$$

Now it is possible to describe samples from either class with one equation,

$$y_i(\bar{w}\cdot\bar{x}_i+b)-1\geq 0 \Rightarrow y_i(\bar{w}\cdot\bar{x}_i+b)=1 \text{ if } \bar{x}_i \text{ is in the margin} \tag{2.20}$$

The total margin or separability from 2.1 needs to be calculated. In order to do it the distance between the hyperplanes in each margin is computed:

$$separability = (\bar{x}_1-\bar{x}_2)\cdot\frac{\bar{w}}{\|\bar{w}\|}, \tag{2.21}$$

where $\bar{x}_1$ and $\bar{x}_2$ are support vector that are the samples from each class closer to the center of the hyperplane, they are also called margin vectors. Using (2.20) it is determined:

$$\bar{x}_1\cdot\bar{w}=1-b \; and \; -\bar{x}_2\cdot\bar{w}=1+b \tag{2.22}$$

The next step is to maximize the separability. In order to do that equation 2.22 is substituted in the equation (2.21). With this it is obtained that the $separability=\frac{2}{\|\bar{w}\|}$ and it needs to be maximized :

$$max\left(\frac{2}{\|\bar{w}\|}\right)\Leftrightarrow max\left(\frac{1}{\|\bar{w}\|}\right)\Leftrightarrow min\left(\|\bar{w}\|\right)\Leftrightarrow min\left(\frac{1}{2}\|\bar{w}\|^2\right). \tag{2.23}$$

Minimizing $\left(\frac{1}{2}\|\bar{w}\|^2\right)$ is a nonlinear optimization task, solved by the Karush-Kuhn-Tucker(KKT) conditions[32]. To solve this optimization problem, the Lagrange multipliers α are used:

$$L=\frac{1}{2}\|\bar{w}\|^2-\sum_{i=1}^{n}\alpha_i\left[y_i(\bar{w}\cdot\bar{x}_i+b)-1\right]. \tag{2.24}$$

In the next step L is derived in with respect to $\bar{w}$ and b:

$$\frac{\partial L}{\partial\bar{w}}=\bar{w}-\sum_{i=1}^{n}\alpha_i y_i\bar{x}_i=0\Rightarrow\bar{w}=\sum_{i=1}^{n}\alpha_i y_i\bar{x}_i, \tag{2.25}$$

$$\frac{\partial L}{\partial b}=-\sum_{i=1}^{n}\alpha_i y_i=0\Rightarrow\sum_{i=1}^{n}\alpha_i y_i=0. \tag{2.26}$$

It is possible to conclude that from (2.25) $\bar{w}$ will be a linear combination from the training vectors. The α_i higher than 0 correspond to the support vectors. By replacing $\bar{w}$ from equation 2.25 in (2.24) it is obtained:

$$\begin{aligned} L &= \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i \bar{x}_i\right)\left(\sum_{j=1}^{n}\alpha_j y_j \bar{x}_j\right) - \sum_{i=1}^{n}\alpha_i y_i \bar{x}_i\left(\sum_{j=1}^{n}\alpha_j y_j \bar{x}_j\right) - \underbrace{\sum_{i=1}^{n}\alpha_i y_i b}_{0} + \sum_{i=1}^{n}\alpha_i \\ &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=0}^{n}\sum_{j=0}^{n}\alpha_i\alpha_j y_i y_j \bar{x}_i \cdot \bar{x}_j \end{aligned} \tag{2.27}$$

To solve the maximization in equation (2.27) the method described in [35] was used. The maximum separability of the hyperplane is given by:

$$w^* = \sum_{i=1}^{n}\alpha_i^* y_i x_i, \tag{2.28}$$

$$b^* = -\frac{1}{2}\langle w^*,\, x_r + x_s\rangle, \tag{2.29}$$

$$\alpha_r > 0, \alpha_s > 0 \text{ and } y_r = -1\, y_s = 1 \tag{2.30}$$

Where x_r and x_s are the support vectors from each class where the conditions in 2.30.

In situations where it is not possible to linearly separate the two classes a transformation in to an higher space dimension is needed. This is accomplished recurring to a Kernel function. In the higher dimension space ϕ it is necessary to maximize $\phi(\bar{x}_i)\cdot\phi(\bar{x}_j)$.

$$K(\bar{x}_i, \bar{x}_j) = \phi(\bar{x}_i)\cdot\phi(\bar{x}_j). \tag{2.31}$$

In many situations there is the need of classifying multiple classes. For solving this there are two wildly used methods: ”one-against-all” and ”one-against-one” approaches. A detailed explanation on this can be found in [29].

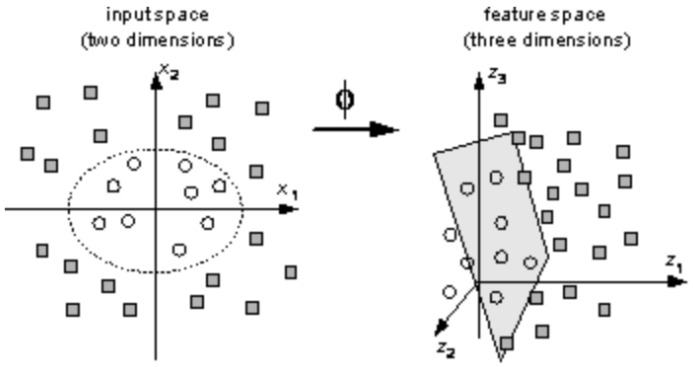


Figure 2.2: Kernel function applied to space transformation, image adapted from [7].

2.1.1.6 Naive Bayes

Naive Bayes is a classification algorithm that is based in Thomas Bayes theorem. Naive Bayes assumes that each feature is independent and therefore unrelated to each other.

Bayes' theorem tries to describe a probability of an event and its mathematical form is defined as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{2.32}$$

A and B are events and $P(A)$ and $P(B)$ are the prior probabilities of those events. Sometimes the equation 2.32 is also defined with Bayesian terms a it is written as:

$$posterior = \frac{likelihood \times prior}{evidence} \tag{2.33}$$

When applied to classification if the features vectors are defined as $A_1(...)A_n$ and the classes are defined as C the Bayes' theorem establishes the following:

$$P(C|A_1,...,A_n) = \frac{P(C)P(A_1,\ldots,A_n|C)}{P(A_1,\ldots,A_n)} \tag{2.34}$$

Then by using the naive independence of the features the algorithm assumes that

$$P(A_i|C,A_1,\ldots,A_{i-1},A_{i+1},\ldots,A_n) = P(A_i|C), \tag{2.35}$$

By simplifying the equation 2.35 the naive Bayes probability model is obtained:

$$P(C \mid A_1,\ldots,A_n) = \frac{P(C)\prod_{i=1}^{n} P(A_i \mid C)}{P(A_1,\ldots,A_n)} \tag{2.36}$$

Combining the model NB probability model with a decision rule algorithm it is possible to make an estimation of $P(C|A)$ and $P(A_i|C)$. The resulting Naive Bayes Classifier will then be as follows:

$$\hat{y} = \underset{C}{\operatorname{argmax}}\, P(C) \prod_{i=1}^{n} P(A_i|C). \tag{2.37}$$

The most commonly used algorithm for decision rule is the maximum a posteriori estimation. A detailed explanation on MAP can be found in [27].

2.1.1.7 Dynamic Bayesian Mixture Model

Dynamic Bayesian Mixture Model (DBMM) was proposed in [23] with the purpose of increasing the classification on human activity recognition. This was accomplished by combining single base classifiers. An extended version of this method was implemented in [24]. In this version DBMM already included a dynamic update of the weights attributed to each classifier by using the passed information of the system. Using this process the weights of each classifier are updated using the memory from past behaviours. The principles of Bayesian Mixture Models (BMM) are used dynamically for combining the outputs from different probabilistic classifiers, resulting in the DBMM algorithm. An overview on the algorithm can be seen in 2.3. The detailed mathematical explanation is documented in [23] and [24].

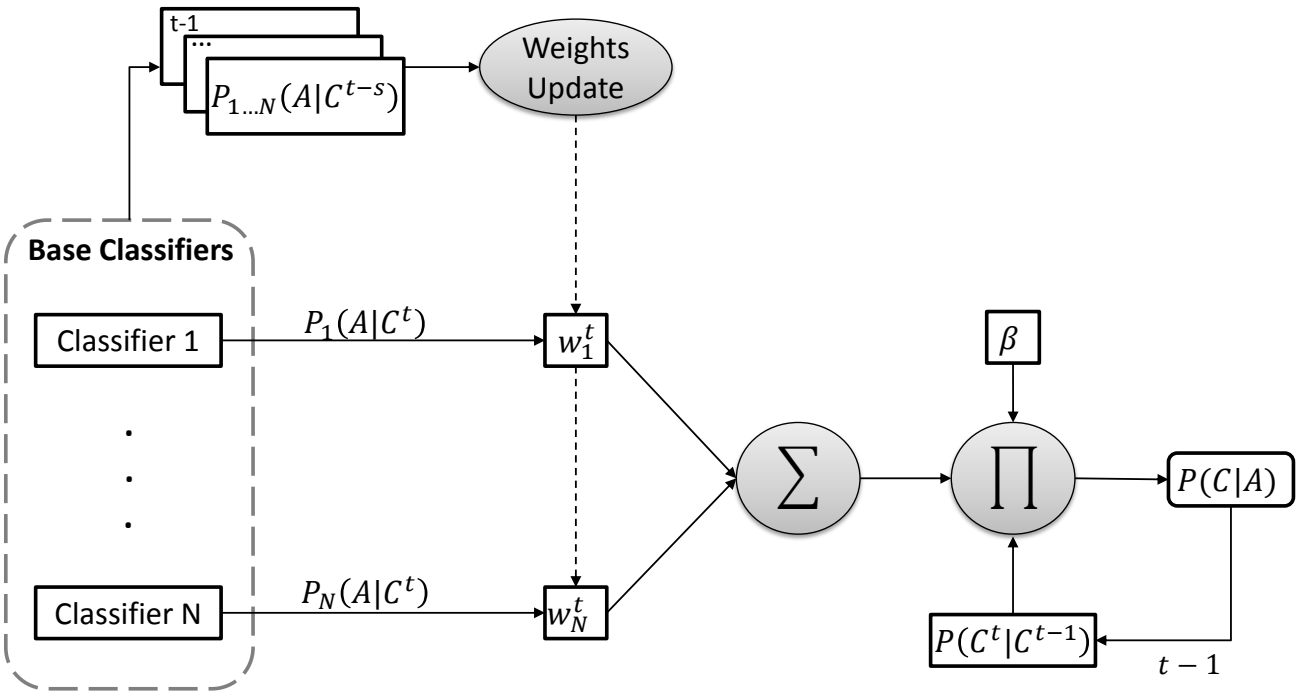


Figure 2.3: Dynamic Bayesian Mixture Model overview, image from [36].

2.2 State of the Art

It is proved that a robot can be a catalyst for increasing social interaction in children with social disabilities [25].

Child-robot interaction in teaching and educational environments require many different functionalities. In order for the robotic platform to be able to interact with children it needs different social skills. The functionalities of the used framework in the interaction must implement those skills. Skills like people detection, movement generalization, movement classification and speech recognition processing are essential in any interaction.

Humanoid robots, like NAO robot, are great investigation tools while at the same time have great results engaging the children attention.

In the rest of this chapter an overview on the related works to the proposed system is presented.

2.2.1 Robot Learning by Imitation

Robot learning by demonstration is a commonly used approach for human movement generalization. In [18] a real-time motion imitation by an humanoid robot was presented. Some years later robotic learning by kinhestetic demonstration was proposed in [15]. In [17] an approach based on Hidden Markov Model and Gaussian Mixture Regression for movement learning and reproduction was presented. Other works were also done by acquiring the motion information trough the use of wearable sensors on the human body. The work in [11] accomplishes this and also uses NAO as the robotic platform for reproducing the movements.

Many other works were developed in this area, the ones closer to the proposed method are [22] and [33]. In [22] the Kinect sensor was used as the movement teaching method for the robot gesture imitation. In [33] the kinect sensor was also used as a teaching method and NAO robot was used as the platform for the movement reproduction. The proposed method used the work developed in [16]. The method was adapted for a more user friendly acquisition by replacing kinesthetic learning system with a gesture learning acquisition using the kinect sensor.

2.2.2 Child-Robot iteraction

There are many works in children-robot interaction with different purposes. The worked proposed here is intended for being continued until the proposed system is fitted to work in an environment with children with disabilities, more specifically children that suffer from cerebral palsy (CP).

The investigation on this field started long time ago when computers started being used in the treatment of children with autism [19]. Ever since many different works were developed, and robotic platforms started to be introduced in the interactions. In [21] the authors address the problem of what are the social skills needed in order for the robotic platforms to develop a 'social-behavior. In [30] the authors used Keepon a creature-like robot in order to engage the attention of children with autism. The work proposed in [14] presented Robota, a mini-humanoid doll-shaped robot for building children–robot social interactions with children with autism. In [37] presented an explanation on how mobile robots could teach children with autism social skills, [30] is a work where this was implemented. In [25] robot interaction with children with autism spectrum disorders was studied. Later in 2012 a similar work [34] was done recurring to NAO robot as the robotic platform. The closest work to the work proposed here is proposed in [38] where the NAO robot was used with the purpose of motivating childrens with CP to keep them engaged in therapy. A common limitation on all the works mentioned is that all the actions are predefined. The work developed here tries to implement a system also capable of learning new movements from the interaction with the user.

In table 2.2.2 it is exposed some of the related works in child-robot interaction. It includes a small description from each work.

Work	Area of work	Description
Ramirez-Serrano et al. [13]	Medi robot interacting with children while taking a vaccine	Children interacted with Medi robot in order to engage the children attention while taking a vaccine at an hospital. This study proved that the robot helped reducing the stress by the fear of taking a vaccine.
Kozima et al. [31]	Keepon robot interacting with children with autism	Children interacted with Keepon robot in orther to engage in pediatric interaction. In this study it was proved that autistic children approached Keepon and engaged in dyadic interaction with it.
Feil-Seifer et al. [25]	Robot Interaction with children with autism	In this work computers and robot have been used in interaction with children with ASD. It was proved that they can be a catalyst on social interactions.
Shamsuddin et al. [34]	NAO robot Interaction with children with autism	Children with autistic disorder engaged with NAO robot. Results showed a decrease on the autistic behaviours. This research suggests that autistic children of moderately impaired intelligence show good response to robot-based intervention.
Yussof et al. [38]	Humanoid robot Interaction with children with Cerebral Palsy	Children with cerebral palsy (CP) interacted with NAO robot for therapeutic purpose.This study proved how the humanoid robot NAO can be used as tool to assist in improving the outcome of conventional therapy by imitation learning.

Chapter 3

Child-robot Social Interaction System Setup

This chapter will give an overview on the components that compose the setup and on how the selected method is integrated in it.

Firstly a description on the hardware and software is provided which is followed by a detailed explanation on how the setup works is given.

3.1 Hardware and Software

This section describes the hardware and software used in the setup (these components can be seen in figure 3.1). This figure also intends to demonstrate how the system framework works in a real-life situation.

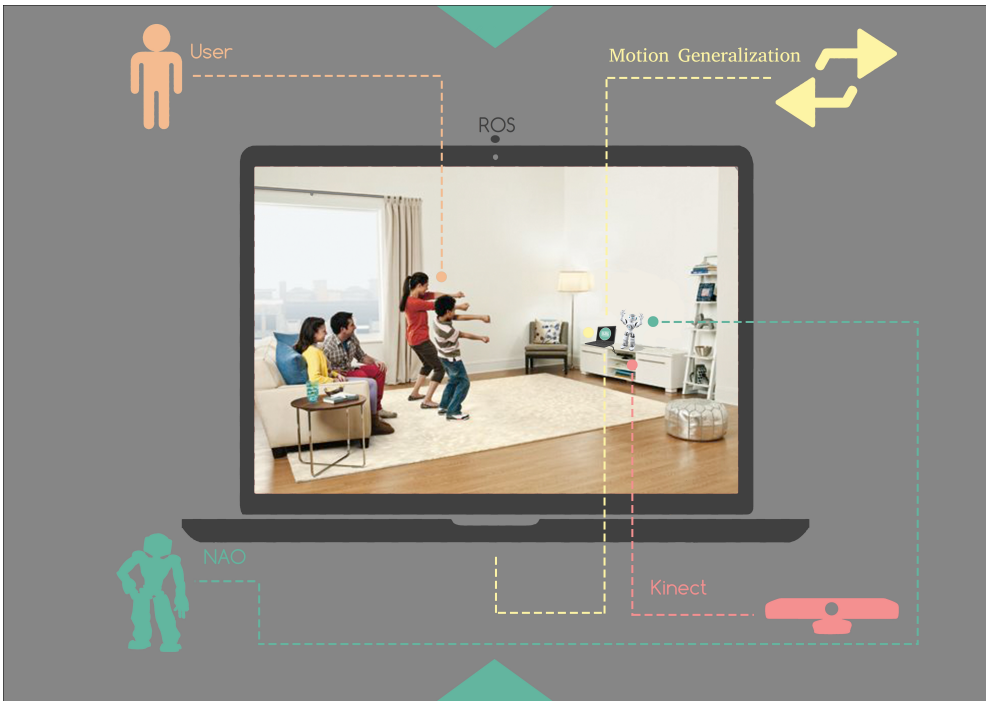


Figure 3.1: System SetUp Components

3.1.1 Microsoft-Kinect

Having the need of a setup suited for a shared-learning environment the kinect was the sensor chosen for depth data acquisition. Furthermore, it is commonly used in movement classification what made it ideal for this system.

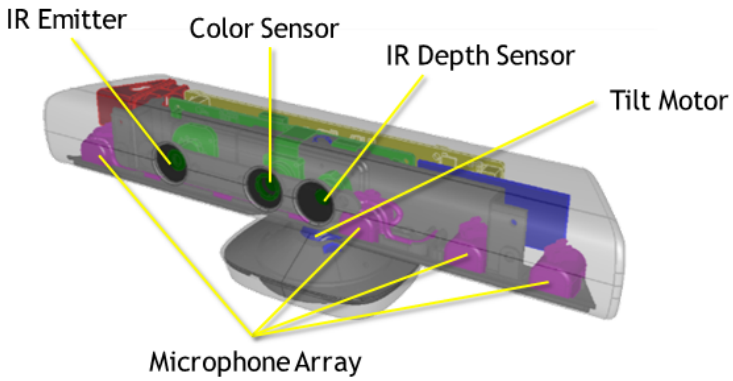


Figure 3.2: Microsoft-Kinect components [10]

The Kinect sensor components figure 3.2 are composed by an infrared, a laser and an RGB camera. The camera specifications are explained in table 3.1.

One of the great advantages of the Kinect sensor is its skeletal tracking, which is the feature that

Kinect	**Array Specifications**
Viewing angle	43° vertical by 57° horizontal field of view
Viewing tilt range	$\pm$ 27°
Frame rate (depth and color stream)	30 frames per second (FPS)
Audio format	16-kHz, 24-bit mono pulse code modulation (PCM)
Audio input characteristics	A four-microphone array with 24-bit analog-to-digital converter (ADC) and Kinect-resident signal processing including acoustic echo cancellation and noise suppresssion
Accelerometer characteristics	A 2G/4G/8G accelerometer configured for the 2G rande, with a 1° accuracy upper limit

Table 3.1: Microsoft-Kinect specifications [10]

allowed data acquiring from the body pose. [1] explains how this process is done. **ROS** will use this feature for publishing the joint values over time (this is explained in more detail in section 3.4).

3.1.2 NAO robot

NAO robot (H25-V4) was the chosen robotic platform for the child-robot system. NAO is a programmable humanoid robotic platform produced by Aldebaran Robotics. Its full documentation can be found in [2] where all the sensors, actuators and joints of the platform are described. Figure 3.3 summarizes in resume this information.

This platform is widely used for investigation purposes, including similar works done in human-interaction therapy for children's with autism in [34].

Since the focus of the proposed system, is for it to work with children, this platform as the right stature since it has $57,4cm$ in height. A really positive feedback was obtained when children interacted with the robotic platform.

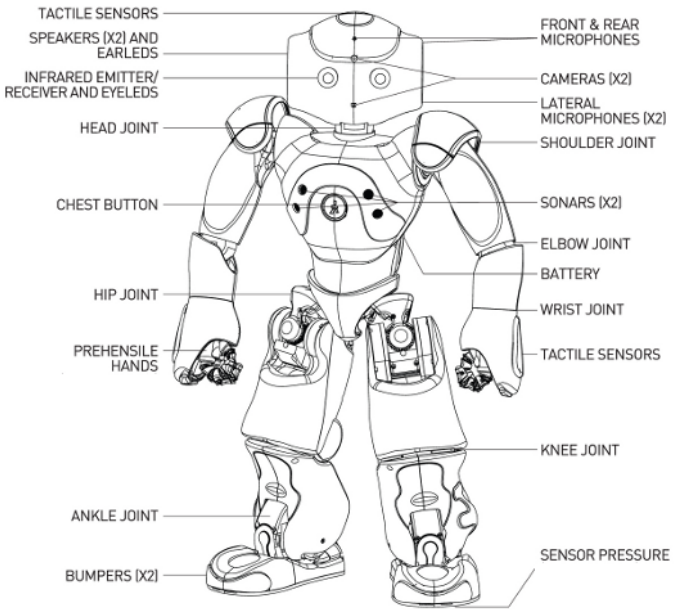


Figure 3.3: NAO H25-V4 description [2]

3.1.3 Robot Operating System (ROS)

ROS is an open source operating system that is used for programming robotic platforms. **ROS** is commonly used in robotics research since it supports and incentivize code sharing and reuse.

ROS is what allows the communication between all the components from the system setup. It also provides the necessary resources for programming the algorithms used.

In the rest of this section a brief explanation on the main applications and utilities from **ROS** is presented.

- **Openni_tracker** is the package that makes the communication with the kinect sensor. This package publishes the transforms corresponding to each body joint. Each joint is defined as transformation **(x,y,z,yaw,pitch,roll)** from the camera. The list of body joints published can be seen in figure 3.4.

- **Pi_speech** processes the voice. This package grants the possibility of a contact less setup. Using this package and **Openni_tracker** conceived the system a shared-learning environment.

- **Mvm_database** is where the movement signatures are saved for future reproductions.

- **Mvm_classify** is the package which includes the classification process defined in section 4.3. This is used when the setup needs to assess if the movement already exists in the database.

- **Motion_genralization** correspond to the method explained in section 4.1.2. When the setup requires the signature for a new movement this algorithm is called.

- **Nao_control** is the package that controls the robotic platform. Aldebaran Robotics provides a python API which allows communication between **ROS** and **NAO** (more details can be found in [3]).

- **Full_app** is the package that controls all the setup, works as the bridge between all the packages described above.

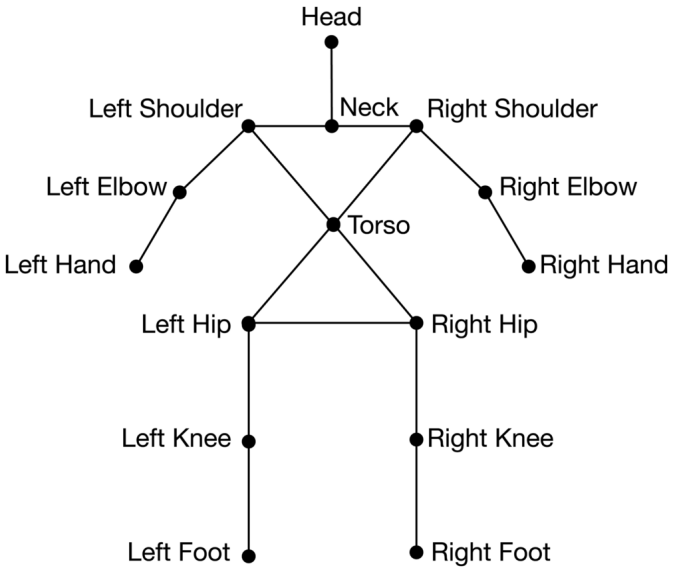


Figure 3.4: OpenNI's kinematic model of the human body [6]

3.2 System Setup Overview

In this section a detailed explanation on the way the system setup works is presented. An overview of the system is shown in figure 3.5.

There are some pre-requirements for the system to work. The user should be within the range of vision of the Kinect camera and relatively close to the computer microphone. Once in within

range of the Kinect sensor, the user must do a calibration pose by raising his arms. This will activate **Openni_tracker** package, which will start publishing the skeleton's joints. Once the pre-requisites are met the system is ready to use.

The setup starts with the imitation mode by making NAO imitate automatically the users arms movements. The package **Pi_speech** is in standby waiting for a command. When the user asks if NAO recognizes the movement **(light-blue arrow)**, **Pi_speech** will trigger **Tf_saving**. **Tf_saving (dark-blue arrow)** will then record sets of 5 seconds of the joint transforms for being compared in classification. Once **Mvm_classify** finishes the classification, one of two options will occur:

- The movement is recognized **(green arrow)** and **ROS** will search in the movement database sending it then to the robot for reproduction. NAO will acknowledge to the user that he recognizes the movement and will reproduce it.
- The movement is not recognized **(red arrow)** and **ROS** will send a command to **NAO** to inform the user. **NAO** will explain to the user that the movement was not recognized and will ask for more repetitions with the purpose of learning it. The user replies and once the command "START" is sent, **(grey arrow) Pi_speech** will activate **Tf_saving(yellow arrow)**. Afterwards, when the "STOP" command is sent **ROS** will generated and send the dataset to the **Motion_genralization** method**(orange arrow)**, which will then generate a movement signature. The new movement will be saved in the **Mvm_dataset** and ROS will send a command to **NAO** to reproduce it. Finally, **NAO** will inform the user that a signature from the movement was generated and saved for future reproductions.

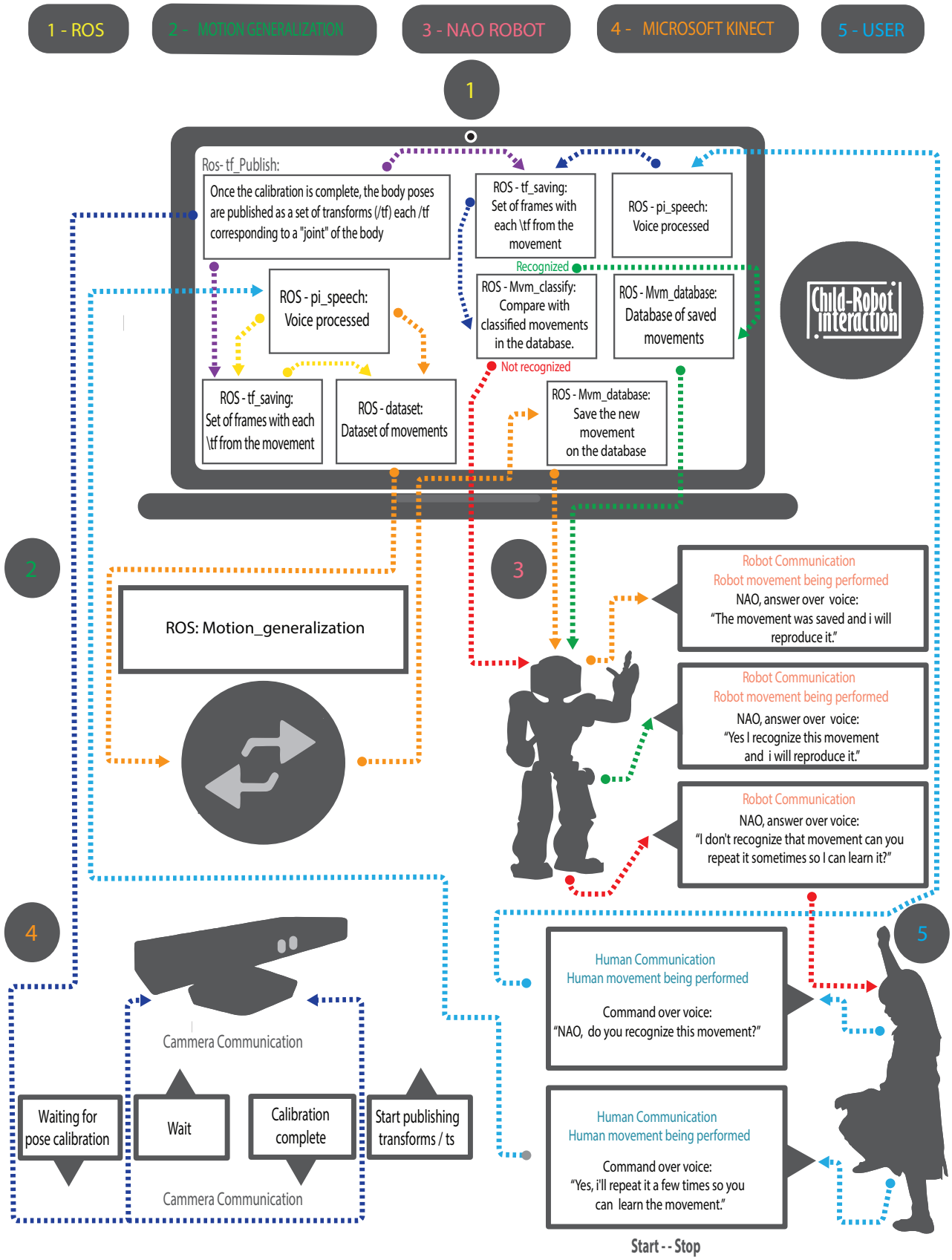


Figure 3.5: System setup overview

Chapter 4

Motion Generalization and Classification

In the previous chapter the system requirements were exposed. It was also explained how the system setup functions. The system requires motion generalization and classification.

This chapter explain the two approaches used to accomplish this. It will explain the necessary steps to learn, generalize and reproduce a movement. The method developed here was based on what is accomplished in [16], being that work adapted to meet the system objectives.

Firstly the same method of the work [16] was applied, using kinhestetic learning to learn and generalize new movements with NAO robot. Being the aim of the work proposed, for the robot to interact with disabled children's there was the need for creating a more user friendly setup. The system needed a contact less movement learning system that would be suited for those children, being the kinect gesture learning the proposed method to accomplish this. This made it possible for the system setup to work on a shared-learning environment.

Finally this chapter will explain the movement classification used based on the work proposed in[24] and [36].

4.1 Kinesthetic Learning

4.1.1 Data Acquisition

In order to learn and generalize a movement it is required a way to acquire the information of a movement. In orther to do so, the first approach followed the same method used in [16] where kinesthetic

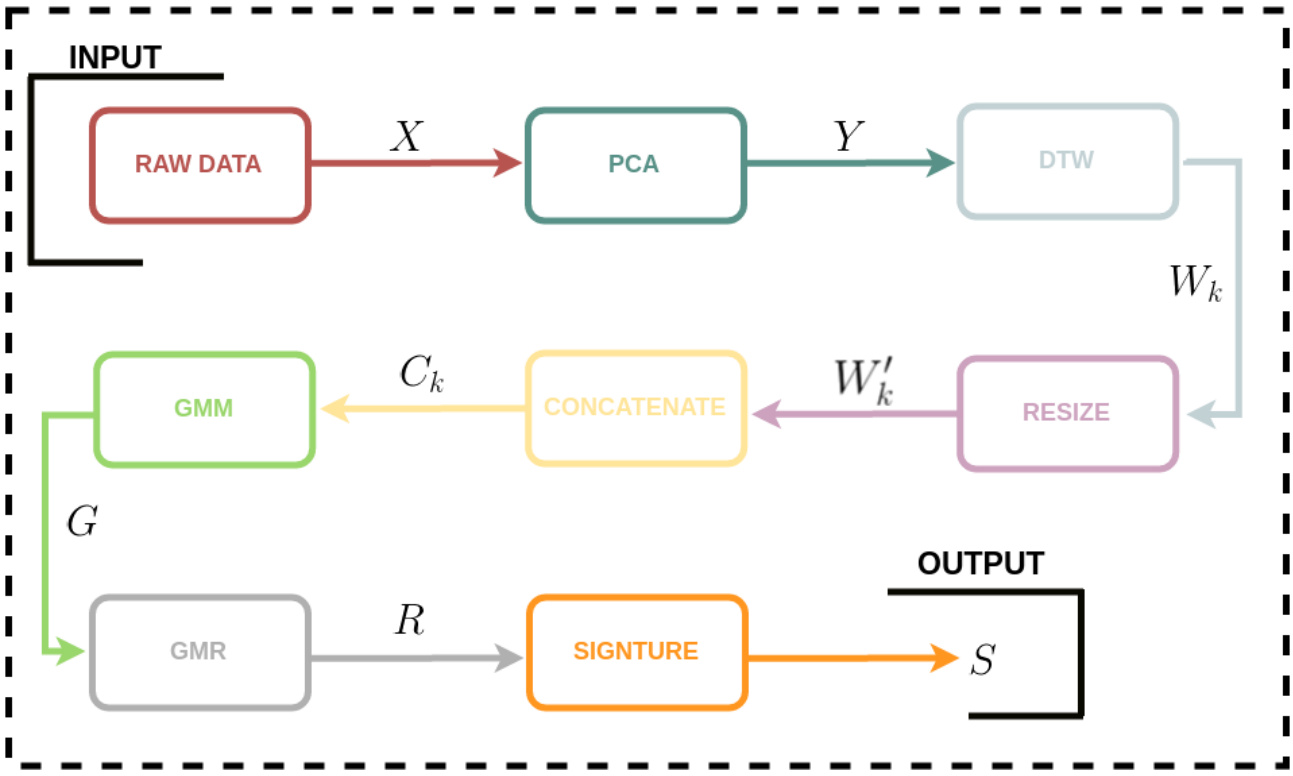


Figure 4.1: Motion Learning and Generalization - Model Overview

Movement	N. of Persons	N. of Repetitions	N. Repetitions(i) total	N. of Joints From NAO
Clapping (1)	2	3	6	12
Raise arms (2)	2	3	6	12
Bolt (3)	2	3	6	12
Bye Bye (4)	2	3	6	12

Table 4.1: Dataset from the Kinesthetic Demonstrations

learning was used to collect the robot joints information along the movement.

Kinesthetic learning, when applied to robot movement learning, consists on acquiring the joints information over time while an experienced user manipulates the robot to perform the desired movement.

This work focused only on the upper body movements, so the joint information collected targeted only the joints described in 4.2,in which each joint name is written in bold.

Two different experienced users reproduced 4 different movements, each user repeated each movement 3 times. The data from each joint was acquired at a frequency of 10 hz. The acquired dataset structure can be seen in table 4.1.

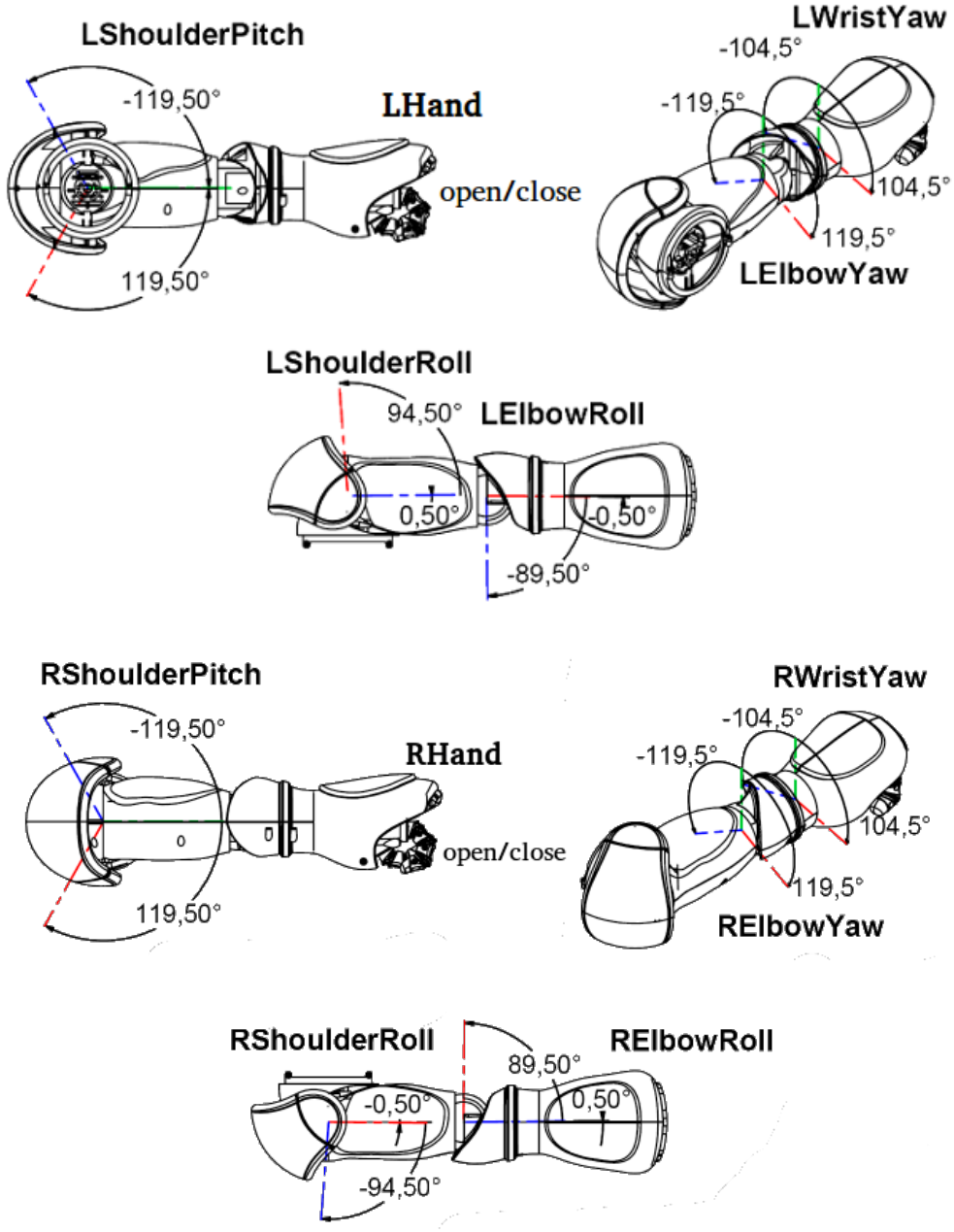
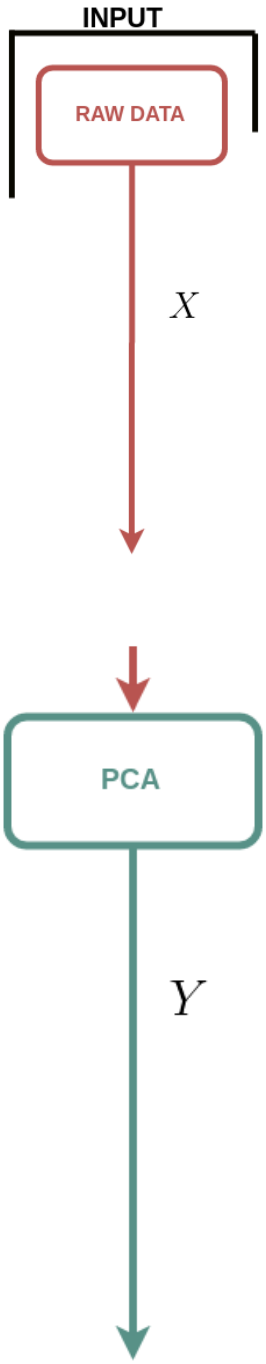


Figure 4.2: NAO H25 V4 arms joints information. (Image adapted from Aldebaran Documentation on NAO)

Symbol	Meaning
i	Index corresponding to each of the movement repetition
l_i	Corrresponds to the size of all of the joint vectors (v_j) in the ith repetition
v_a	Vector corresponding to the joint number a, where $a = (1...12)$
d_i	Matrix with the data from the 12 v_j corresponding to the the ith repetition, where i=1 ... 6
X	Input dataset
j	Number of necessary principal components to obtain a 99% comeback to the original data
u_k	Vectors corresponding to each principal component, $k = (1...j)$
Y	Matrix composed by the principal components of the 6 repetitions, representing the data X in the latent space
w_i	Matrix composed by the principal component coefficients corresponding to the repetition i
pc_i	Matrix composed by the principal components corresponding to the repetition i
λ_i	Matrix composed by the eigenvalues of the latent space corresponding to the repetition i
W_k	Matrix composed by the warped principal components of the 6 repetitions
W'_k	Matrix composed by the resized warped principal componentsof the 6 repetitions
C_k	Matrix composed by the concatenation of W'_k and the corresponding temporal constraints
m	Number of clusters used
γ	Mean corresponding to a specific cluster
σ	Mean corresponding to a specific cluster
G_k	Matrix composed by the means and covariances from all the clusters
u_{r_k}	Regression vector of each principal component, $k = (1...k)$
R_k	Matrix composed by all the u_{r_k}.
v'_a	Signature vector corresponding to the joint number a, where $a = (1...12)$
S	Matrix composed by all the signature vectors from each joint

Table 4.2: Table of symbols and abbreviations

4.1.2 Movement Learning and Generalization



As explained before, each movement is repeated 6 times in total and a movement is composed by 12 joints (v_a), which are described in table 4.1. Each of the 6 repetitions of a movement can be defined with the following matrix:

$$d_i = \begin{bmatrix} v_1^{l_i} & (...) & v_{12}^{l_i} \end{bmatrix} \tag{4.1}$$

The input (X) is composed by all 6 demonstrations of a specific movement defined in the equation 4.1 and is defined as:

$$X = \begin{bmatrix} d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \end{bmatrix} \tag{4.2}$$

For all movements the data acquired has shown redundancy making it possible to represent it in a latent space of lower dimensions. This allowed to reduce the computational time and the amount of computational resources used. PCA was applied to the input X. A detailed explanation on how this algorithm works can be found in section 2.1.1.1. PCA will orthogonally transform the vectors d_i in X into a latent space of principal components Y as follows:

$$Y = \begin{bmatrix} pc_1 & pc_2 & pc_3 & pc_4 & pc_5 & pc_6 \end{bmatrix} \tag{4.3}$$

$$pc_i = \begin{bmatrix} u_1 ... u_j \end{bmatrix}, i = 1...6 \tag{4.4}$$

All the principal component vectors (u) in pc_i have the same size l_i as the corresponding vectors in d_i. Each pc_i matrix has associated to itself a matrix of coefficients (w_i) and a matrix of eigenvalues (λ_i). Eigenvalues have an important role since they can be used as a metric of "relevance" of each pc_i.

It is at this point where the reduction of dimensionality happens, *PCA* sorts each column of the pc_i by "relevance", and the eigenvalues tell us the importance of each column. With this information it is chosen the number (j) of necessary columns (u), in order to reach at least 99% of "comeback" to the original data.

As it can be seen in the results 5.2 for all movement repetitions of each movement 2 to 3 vectors u_k of a pc_i matrix were enought to achieve the required percentage of "comeback". Since the mean is subtracted before performing *PCA*, the mean of each $d_i(v_a)$ is saved as $\bar{d}_i$.

$$[w_i(t_k), pc_i(u_k), \lambda_i(s_k)] = \mathbf{PCA}(d_i(v_a)), \tag{4.5}$$

In the end there is the need to "comeback" to the original data, this is done as follows:

$$pc_i(u_k) = (d_i(v_a) - \bar{d}_i(v_a)) * w_i(u_k) \tag{4.6}$$

since w is orthogonal this can be rewritten as,

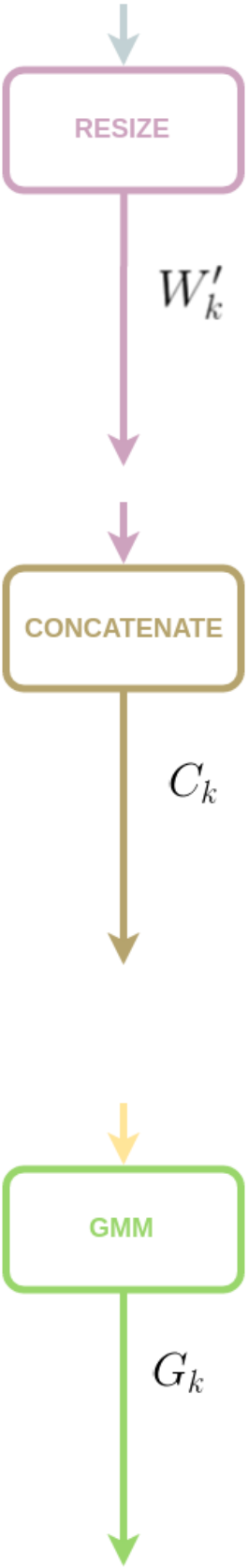
$$d_i(v_a) = \bar{d}_i(v_a) + pc_i(u_k) * w_i'(u_k) \tag{4.7}$$



As each repetition has a different duration, if there is no pre-processing GMM may have a poor performance. Therefore, in the next step of the method dynamic time warp (DTW) is applied to each $pc_i(u_k)$. Since the same movement can be reproduced with different lengths and still be the same movement, DTW is used to align the movement repetitions.

As explained in the background, DTW will align a vector with a reference vector. An average vector with data from 6 repetitions was generated to work as the reference vector. To accomplish this a resizing function which performs a polynomial interpolation (spline) was used in order to resize all the vectors to the average size. The resized vectors were then used to create the average vector.

$$W_k = \begin{bmatrix} pc_{w_1}(t_k) & pc_{w_2}(t_k) & pc_{w_3}(t_k) & pc_{t_4}(t_k) & pc_{w_5}(t_k) & pc_{w_6}(t_k) \end{bmatrix} \tag{4.8}$$



The warped vectors in W_k have higher dimension because of the way the DTW algorithm performs. When working with movements the dimension of the movement vector is important as each value has a temporal value associated to itself. To avoid this problem caused by the DTW algorithm the resizing function was used again to resize each vector in pc_{w_i} to its original size:

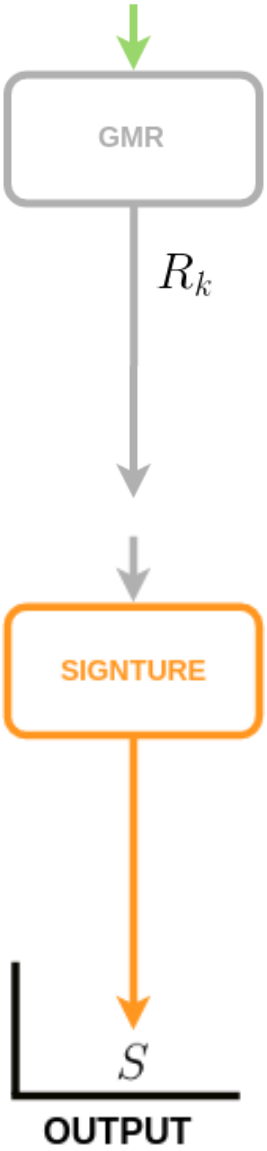
$$W'_k = \begin{bmatrix} pc_{w_1}(u_k) & pc_{w_2}(u_k) & pc_{w_3}(u_k) & pc_{t_4}(u_k) & pc_{w_5}(u_k) & pc_{w_6}(u_k) \end{bmatrix} \tag{4.9}$$

Before performing Gaussian mixture model (GMM) the warped vectors from W'_k are concatenated with the respective temporal constraint in to a matrice C_k:

$$C_k = \begin{bmatrix} t_{1k} & t_{2k} & t_{3k} & t_{4k} & t_{5k} & t_{6k} \\ pc_{w_1}(u_k) & pc_{w_2}(u_k) & pc_{w_3}(u_k) & pc_{t_4}(u_k) & pc_{w_5}(u_k) & pc_{w_6}(u_k) \end{bmatrix}_{k=(1\ldots j)} \tag{4.10}$$

In the next step,(GMM), a probabilistic model for clustering is a applied to C_k in order to to itsdescribe subsets of data. GMM was applied to each group of principal components C_k.GMM uses a predefined number m of clusters for representing the data, which must be chosen carefully: if it is too high overfit may occur; being too low can lead to a poor representation of the data. Following the suggestion in [16] 10 clusters ($m = 10$) were used and good results were obtained in all the movements. Each cluster is defined by a mean (γ) and a covariance (σ) and these 2 values are updated in each iteration. The algorithm iterates until it converges to a satisfying representation of the data. In the end all the clusters are represented in matrix G_k as follows:

$$G_k = \begin{bmatrix} \gamma^1 & \ldots & \gamma^m \\ \sigma^1 & \ldots & \sigma^m \end{bmatrix}_{k=(1\ldots j)} \tag{4.11}$$



4.1.3 Movement Imitation

Using the means and covariances from GMM, gaussian mixture regression (GMR) will create a regression(R_k) for each of the principal components:

$$R_k = \begin{bmatrix} u_{r_1} & \dots & u_{r_j} \end{bmatrix} \tag{4.12}$$

Using the regression of each principal component and the average of each vector, following the procedure in (4.7), the motion signature is generated:

$$S = \begin{bmatrix} v'_1 & \dots & v'_{12} \end{bmatrix} \tag{4.13}$$

In the end, in order for the robot to reproduce the movement the vectors v'_a from the motion signature S are sent to the robot joints at the same frequency of acquisition 10 h_z. It is expected that the motion signature has a duration equal to the average of the durations of the movement repetitions.

The algorithm and the code was adapted from [15].

4.2 Kinect Gesture Learning

As previously mentioned, to learn a movement trough kinhestetic learning an experienced user is required to reproduce the movement with the robot. As one may expect, this is not a common user friendly approach, since not every subject can reproduce a movement in this way. This problem, adding to the fact that the final objective is to work with disabled children's, leads to the need of finding an alternative to the kinesthetic learning.

In this final approach kinect gesture learning was the method chosen to acquire motion since it doesn't need any direct iteration. This allowed to work on a shared-learning environment where any subject can teach new movements to the setup.

4.2.1 Kinect Data Acquisition

Microsoft Kinect RGB-D sensor was used on a ROS environment with the purpose of acquiring body pose information. The sensor accomplishes this by acquiring a depth map using the infra-red sensor. In chapter 3 a detailed description of the setup can be found. ROS will publish, at a frequency of 10 *hz*, the information from 15 body joints 4.3. This work only focus on the upper body joints identified in bold in table 4.2.1. Only the translational information of each human body joint is needed to map to the motion to the robot. This means that, only the x, y, and z information over time will be used, ignoring the yaw, pitch and roll information. This is due to the fact that the angular information formed through the body joints will be computed as explained later in this section.

/head	**/left_shoulder**	**/right_shoulder**	/left_hip	/right_hip
/neck	**/left_elbow**	**/right_elbow**	/left_knee	/right_knee
/torso	**/left_hand**	**/right_hand**	/left_foot	/right_foot

Table 4.3: 15 body joints published description

The new input dataset was also acquired by two different persons, making 3 repetitions of a movement each. The new dataset structure can be seen in table 4.4.

4.2.2 Movement Learning and Generalization

The motion learning follows the same method from figure 4.1 that is explained in subsection 4.1.2 with the small difference that each repetition vector has higher dimension being composed by 9 joints and 3

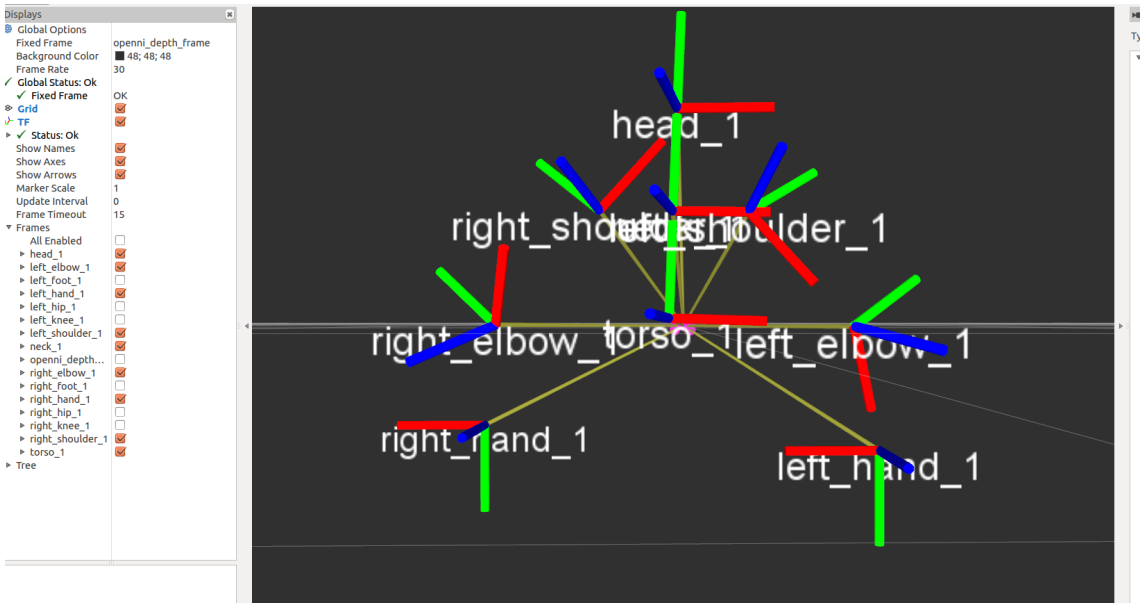


Figure 4.3: Upper body joints published by ROS using the kinect Sensor

Movement	N. of Persons	N. of Repetitions	N. Repetitions(i) total	N. of Body Joints (x,y,z)
Clapping	2	3	6	9
Raise arms	2	3	6	9
Bolt	2	3	6	9
Bye Bye	2	3	6	9

Table 4.4: Dataset from the Kinect Body Pose Acquisition

values(x,y,z) each. One repetition as the following structure:

$$d_i = \begin{bmatrix} v_1^{l_i} & (...) & v_{18}^{l_i} \end{bmatrix} \tag{4.14}$$

The input (X) is composed all the 6 demonstrations of a specific movement and is defined as:

$$X = \begin{bmatrix} d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \end{bmatrix} \tag{4.15}$$

4.2.3 Movement Imitation

As previously mentioned, the same method 4.1 was followed and the movement signature is generated as follows:

$$S = \begin{bmatrix} v'_1 & ... & v'_{18} \end{bmatrix} \tag{4.16}$$

In order to reproduce a movement there is the need to map the movement signature from the subject body joints to the robot joints. This was achieved by adapting the work done in [9] and [8]. One important feature of this work is the way this mapping is done. By using the angles created by the triangles of the arm joints of the user it is possible to calculate the robot joints 4.4. A key feature of the way this is implemented since the mapping does not depend on the size of the user. This particularity makes the setup suitable for adult and children users. Additional filtering was done to the joints values, for noise removal.

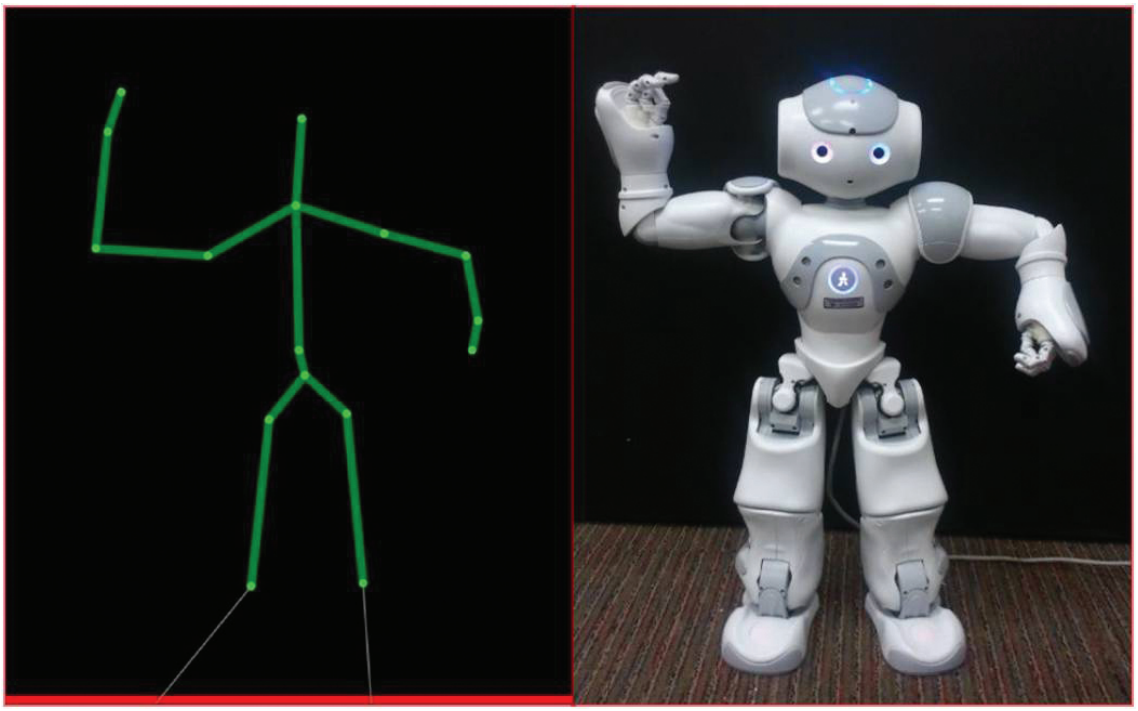


Figure 4.4: Kinect joints mapping to NAO robot [28]

4.3 Classification

A dataset for movement classification was created by 4 different persons repeating 8 different movements. Table 4.5 shows a structure of the dataset with the number of frames from each movement. The following movements: Goodbye, Bolt, Clapping, Cruising Arms, Raising arms to the top, Arms on the hips, Air-plane and Arms calibration pose were the chosen movements for classification. This are common movements in human interaction and allow the robot to engage the user attention.

Kinect sensor was used for the acquisition of the dataset. The information saved the sequences from the body poses in relation to the camera and in relation to the torso. For the classification being suited

Table 4.5: Dataset structure.

Activity	Number of frames
Goodbye	1143
Bolt	818
Clapping	1064
Cruising Arms	986
Raising arms to the top	934
Arms on the hips	933
Air-plane	1158
Arms calibration pose	927

for every user it included 4 different persons, three female subjects and one male subject with different ages. One of the female was a children with 5 years old, once it is a key step to include children in the work. Each subject repeated the different movements several times from different positions. An example from the data acquisition can be seen in figure 4.5

Different features were taken in order to accomplish the classification such as the Euclidean distance between body joints, angle variation, angle speed among others. The work developed here was adapted from the work developed in [36]. Detailed explanation on the features and algorithms can be found in that work. Here the SVM and naive NB were used alongside with DBBM with weights update, as explained in the Background section.

This work presented similar results to the obtained in other works [36] and [24]. The classification obtained good results performing on the fly. A confusion matrix 5.3 for all the movements was computed and the DBBM scored 98.20% on average.

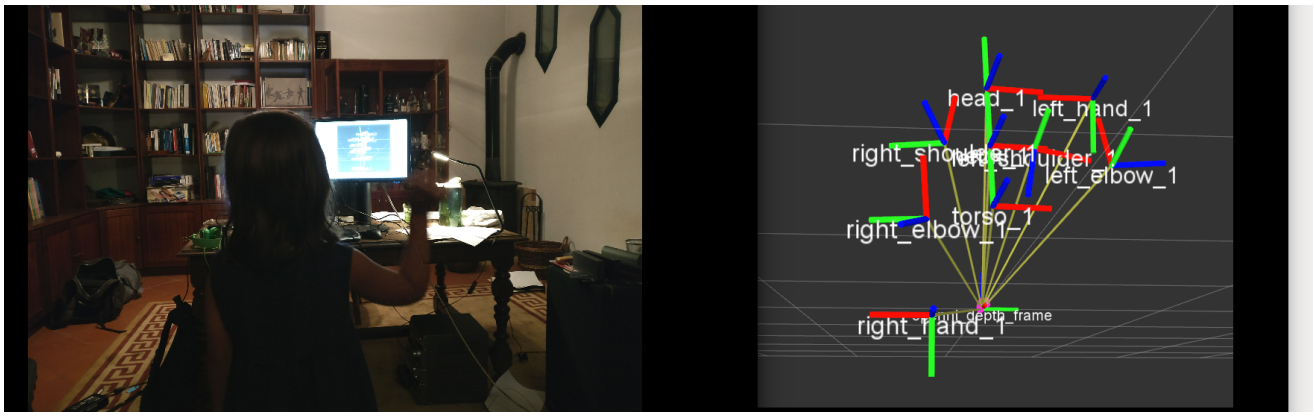


Figure 4.5: 5 year old children performing the Byebye movement in the classification process.

Chapter 5

Experiments, Tests and Results

5.1 Kinesthetic Learning

Kinesthetic learning was used to generalize NAO robot movements. The kinesthetic movements were taught to NAO robot, this can be seen in figures 5.1 and 5.2. To accomplish this a dataset was created which consists of six movements performed by two persons repeating the same movement three times, i.e., six repetitions in total for each movement of the dataset. Six different movements were taught to the robot: goodbye movement; clapping movement; bolt movement; side arms movement; what movement; and finally the raise arms movement. Since these different movements only involved moving the arms, the work focused on collecting the data related to the joints from the robot's arms. NAO robot has 6 joints in each arm, so in total it was collected data from 12 different joints.

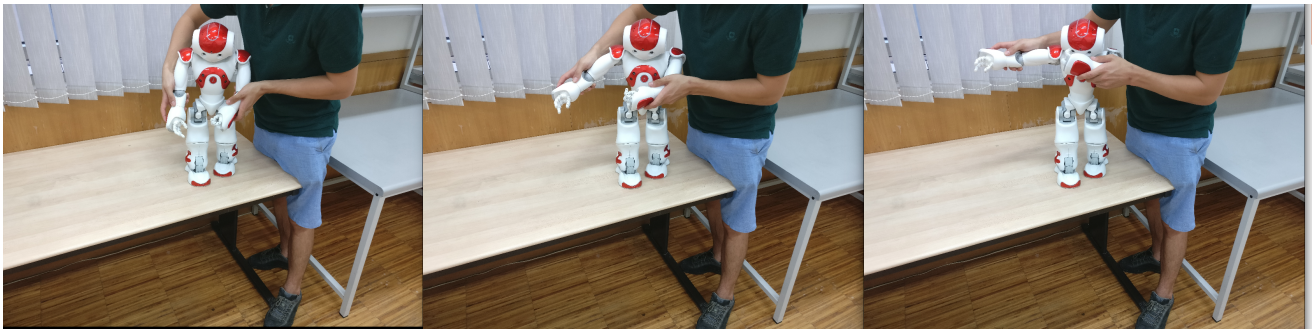


Figure 5.1: Bolt movement kinesthetic learning from user 1

For the data collection a python script was created. The script was programmed to read joints' values every 100 milliseconds. After that, it is created a text file containing the formatted data in 12 vectors, each vector corresponding to one joint. Knowing the frequency of acquisition a temporal constraint is

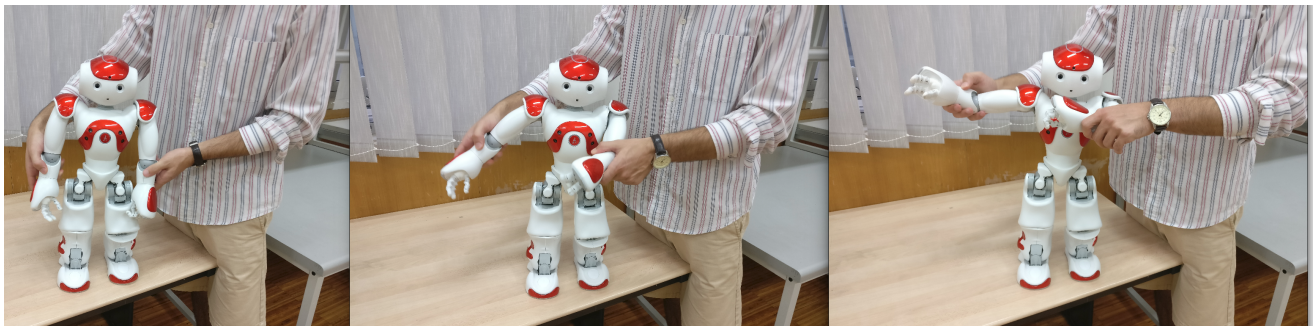


Figure 5.2: Bolt movement kinesthetic learning from user 2.

created for being associated with each vector.

After this step the motion learning explained in section 4.1 was applied and a signature was created for each movement.

The results presented next only refer to the byebye movement. Similar type of results were obtained for the other movements.

Byebye Movement	**Performance %**	**Number of PC used**
medeiros-bey2015-12-7-17-4-31	performance1 = 99.4509	num_pcmvm1 = 3
medeiros-bey2015-12-7-17-5-12	performance2 = 99.4852	num_pcmvm2 = 2
medeiros-bey2015-12-7-17-5-12	performance3 = 99.7948	num_pcmvm3 = 3
rui-bey2015-12-7-17-5-49	performance4 = 99.0976	num_pcmvm4 = 2
rui-bey2015-12-7-17-6-3	performance5 =99.3972	num_pcmvm5 = 2
rui-bey2015-12-7-17-6-14	performance6 =99.3706	num_pcmvm6 = 2

Table 5.1: PCA Analysis, necessary number of principal components to obtain at least 99% ”comeback” to the original data.

After applying **PCA** to the data, testes were done in order to verify how many components are needed to obtain 99% of comeback to the original data (12 vectors, 1 corresponding to each joint). Looking at table 5.1 it is possible to conclude that for this movement at least 3 principal components are required to obtain the desired performance. This happens because the first and the third movement repetition needed 3 principal components to meet the necessary performance.

By using the DTW the next step temporally align the different movement repetitions.

After the two pre-processing steps (PCA,DTW) the data is concatenated in to a 2D matrix using the temporal vectors as a temporal constraint. In the next step GMM and GMR is applied in order to obtain each principal component signature.

Figure 5.3 shows the results from the above steps corresponding to the first principal component

from the Byebye movement.

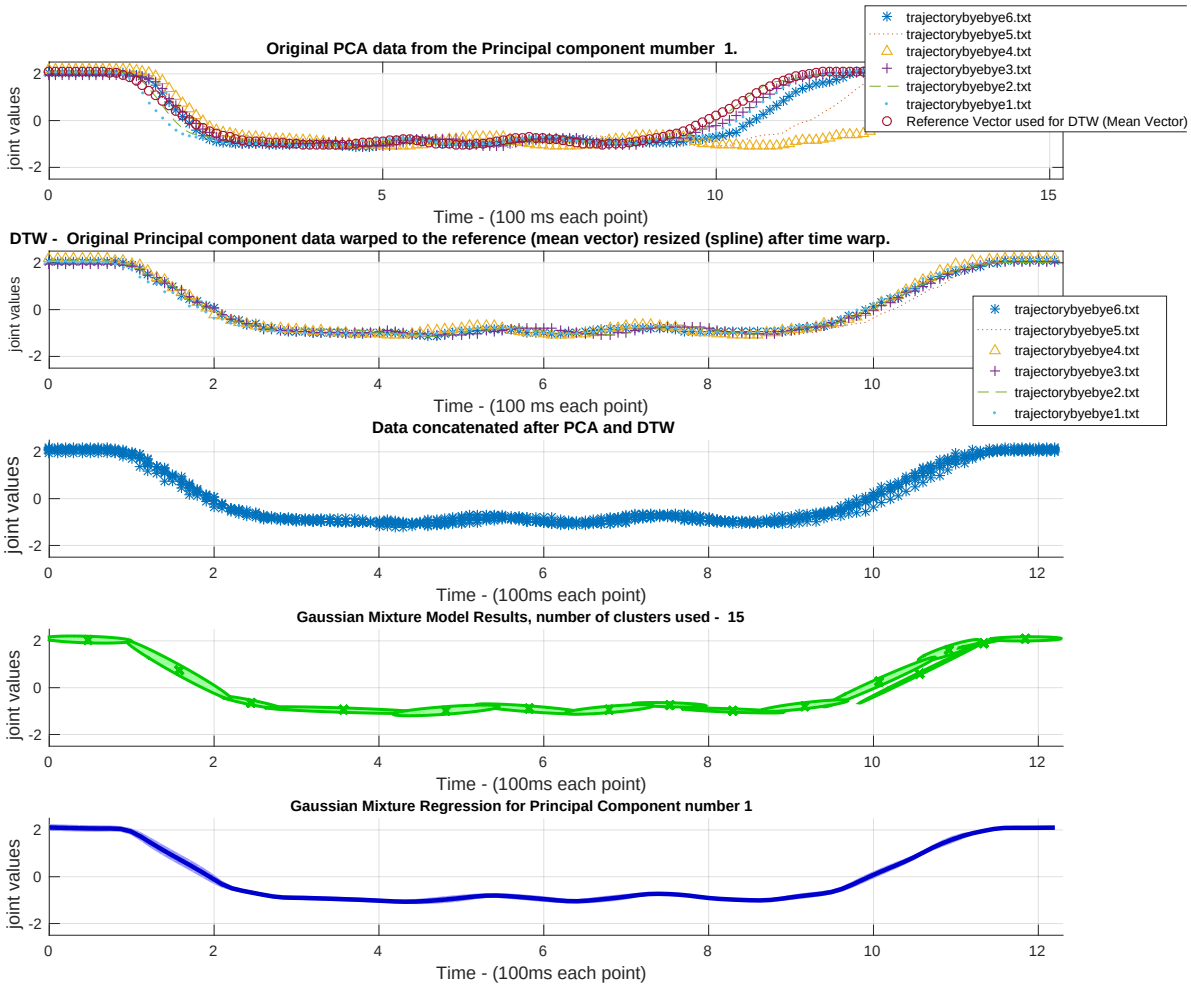


Figure 5.3: Principal component number 1 signature generated from 6 repetitions from the Byebye movement .

It is possible to observe that 15 clusters were used in the first principal component signature. Each movement is composed by the 12 joint values variation overtime. Since this would fill the work with too much information, here it is only presented the two joints with higher variation in this movement. This results can be seen in figure 5.4 and 5.5.

As it would be expected the joints with higher variation were the ones from the right arm of the robot, the one used in the kinhestetic reproduction.

A video with the results from all the movements can be found online in [5].

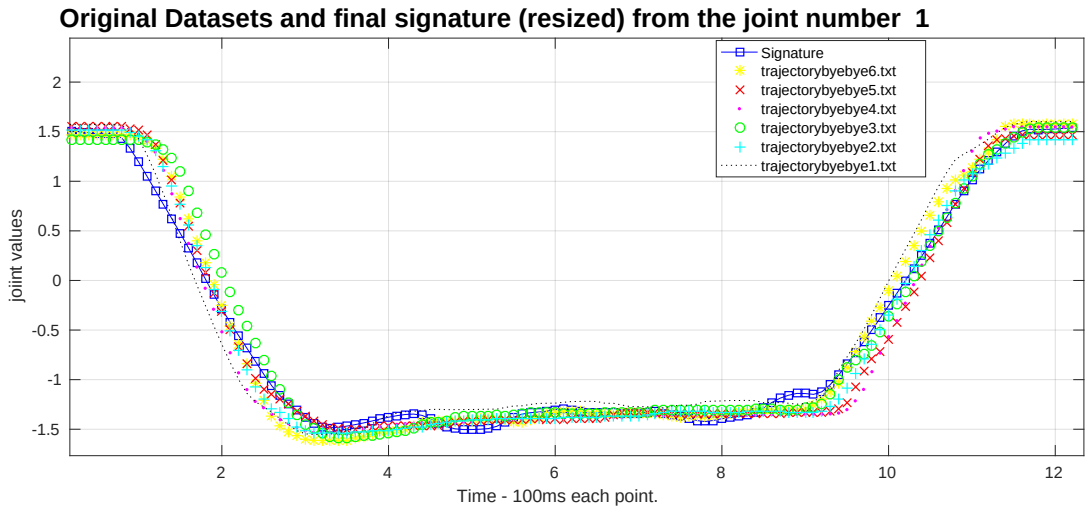


Figure 5.4: Byebye movement signature - information from joint number 1:'RShoulderPitch'.

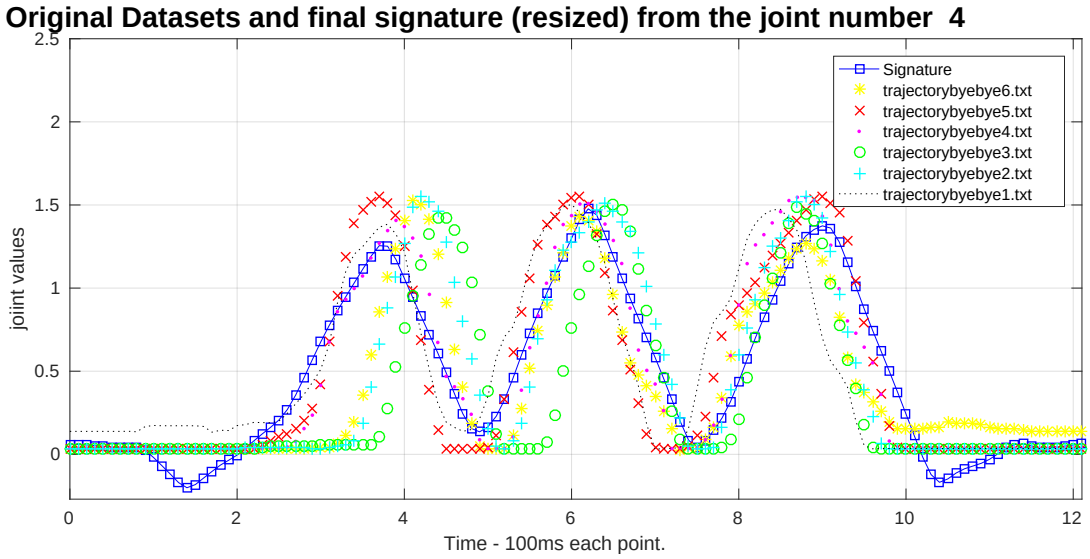


Figure 5.5: Byebye movement signature - information from joint number 4:'RElbowRoll'

5.2 Kinect Gesture Learning

Gesture learning with kinect RGB-D sensor was used for generalization of human movements. Given the context of the work in this section only a single user was used for acquiring movement data since, as explained in chapter 3, only one user interacts with the system. For accomplishing this, each movement was repeated 6 times by an user. The same 6 movements taught in section 5.1 were acquired. The joint information(x,y,z) from the 15 body joints was acquired at a frequency of 10 hertz for each movement. Due to the big dimension of the results it is only presented results for the Byebye movement.

The PCA results, table 5.2, on the Byebye movement required at least 3 principal components need

Byebye Movement	**Performance %**	**Number of PC used**
medeiros-bey2015-12-7-17-4-31	performance1 = 99.7255	num_pcmvm1 = 3
medeiros-bey2015-12-7-17-5-12	performance2 = 99.6881	num_pcmvm2 = 3
medeiros-bey2015-12-7-17-5-12	performance3 = 99.6321	num_pcmvm3 = 3
medeiros-bey2015-12-7-17-5-49	performance4 = 99.6967	num_pcmvm4 = 3
medeiros-bye2015-12-7-17-6-3	performance5 =99.7798	num_pcmvm5 = 3
medeiros-bye2015-12-7-17-6-14	performance6 =99.7170	num_pcmvm6 = 3

Table 5.2: PCA Analysis on byebye movement - necessary number of principal components to otbain at least 99% ”comeback” to the original data.

to be used in all the movement repetitions. PCA still performed really well since it reduced the data dimension from 45 vectors (15 - joints times 3 - joint values) to 3 vectors, showing there is redundancy in the data.

The method for gesture learning only differs on the type of data acquired, the rest of the method executes the same steps as the ones explained in section 5.1. Figure 5.6 shows the PCA signature from principal component number 1 being generated, using the same steps as before. Although it is not presented here the graphics from all the information from each joint were obtained.

In the end these values are mapped trough a set of functions that map body joints into NAO robot joints.

Figure 5.7 shows an example on the bolt movement generalization. First the user reproduces de movement several times. Using the motion repetitions the movement is learned and generalized. The motion signature is then mapped and reproduced by the robot.

A video with the full results from all the movements can be found online in [4].

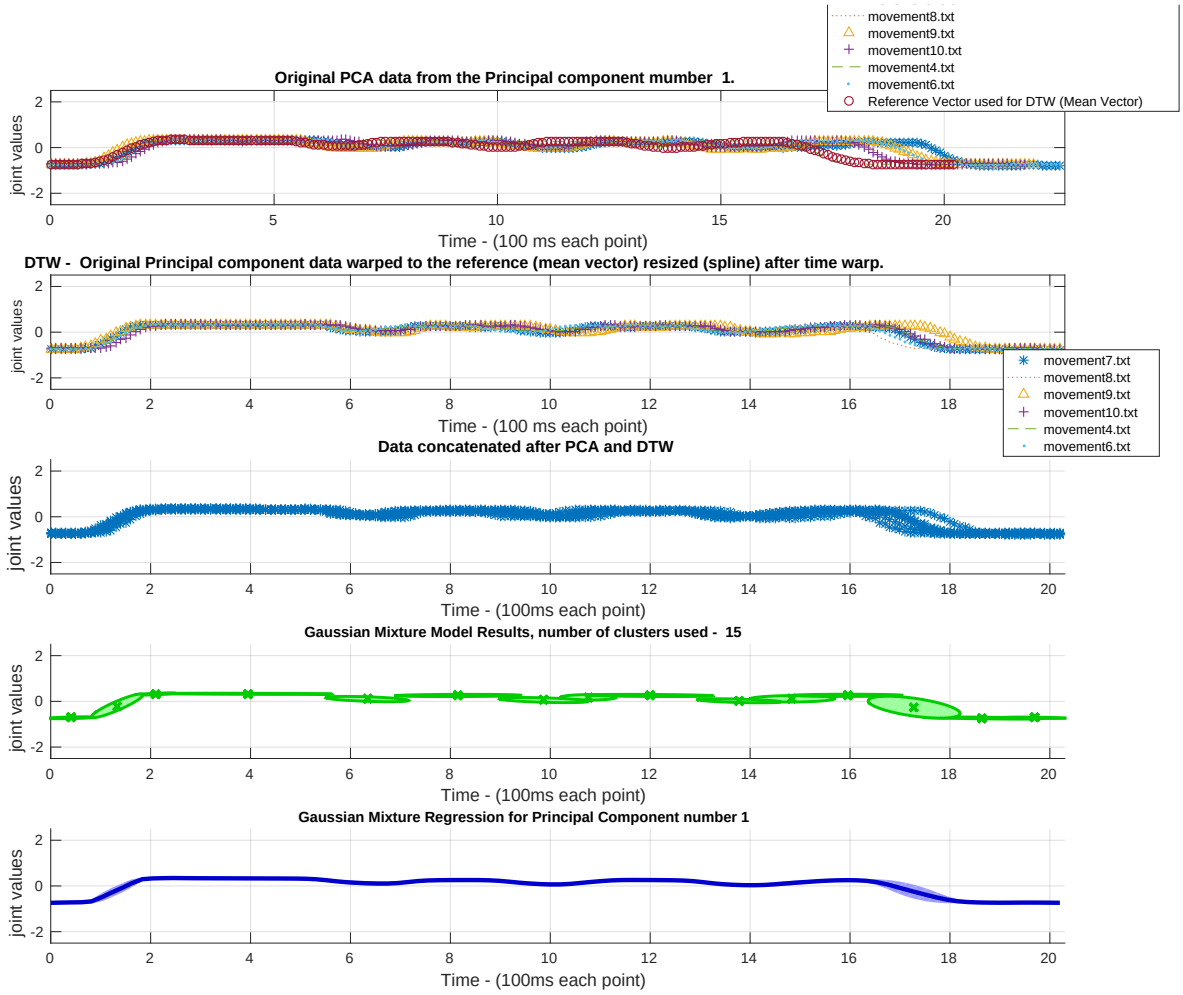


Figure 5.6: Byebye movement motion generalization corresponding to the principal component number 1.

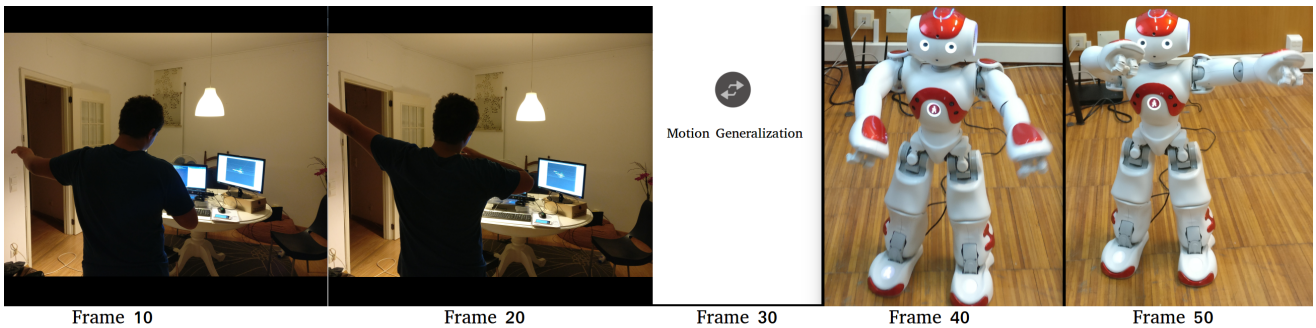


Figure 5.7: Example from the gesture learning, learning the bolt movement.

5.3 Movement Classification

Movement classification was trained for recognizing the 8 following movements: Byebye, Arms calibrate Pose, Bolt, Clapping, Cruising arms, Arms to the top, Arms to the hips and the air plane.

Figures 5.8 and 5.9 show the performance from the single classifiers trained. The SVM classifier scored 94,8 % and the NB classifier scored 74.5 %.

SVM score: 0.948276

	precision	recall	f1-score	support
1.0	0.86	1.00	0.93	115
2.0	0.92	0.95	0.93	95
3.0	1.00	0.92	0.96	83
4.0	0.99	0.94	0.97	109
5.0	0.94	0.95	0.95	101
6.0	0.98	0.92	0.95	96
7.0	0.94	0.96	0.95	95
8.0	0.99	0.94	0.97	118
avg / total	0.95	0.95	0.95	812

Figure 5.8: SVM classifier performance results

Naive Bayes score: 0.745074

movement	precision	recall	f1-score	support
1.0	0.91	0.79	0.85	115
2.0	0.68	0.77	0.72	95
3.0	0.97	0.80	0.87	83
4.0	0.75	0.76	0.76	109
5.0	0.60	0.55	0.58	101
6.0	0.95	0.64	0.76	96
7.0	0.50	0.97	0.66	95
8.0	0.98	0.70	0.82	118
avg / total	0.80	0.75	0.75	812

Figure 5.9: NB classifier performance results

Table 5.3 presents the results from the confusion matrix acquired by the DBMM classification method [24]. This confusion matrix was obtained on the fly while the user performed all the 8 movements several times.

Movement number:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	**98.50**	0	0	0	0	0	1.50	0
2	0	**97.33**	0	0	0	2.67	0	0
3	0	0	**99.03**	0	0	0	0	0.97
4	0	0	0	**97.78**	2.22	0	0	0
5	0	0	0	0.94	**98.12**	0	0.94	0
6	0	2.02	0	0	0	**97.98**	0	0
7	2.10	0	0	0	0	0	**97.90**	0
8	0	1.05	0	0	0	0	0	**98.95**

Table 5.3: Confusion matrix obtained on the fly using DBMM classification method. Movements: 1 - 'Byebye', 2 - 'Arms calibrate pose', 3 - 'Bolt', 4 - 'Clapping', 5 - 'Cruize arms', 6 - 'Arms to the top', 7 - 'Arms What', 8 - 'Airplane'.

The DBMM scored 98.20 % on average showing better results than the single classifiers used for training (SVM and NB).

As it can be seen the system corresponded well in a continuous environment where information is acquired on the fly. This allowed the classification to be included in the system setup.

In figure 4.5 it is possible to see the classification on the fly being performed by a 5 year old children.

Chapter 6

Conclusions and Future Work

The use of robots in therapeutic environments for children has already proved to be a useful tool. Robots, with the required social skills, can be applied to engage the children attention in order to reduce stress. In this work, a fully implemented framework was developed for creating some of those social skills. The system is composed by a computer, NAO robot and a kinect RGB-D depth sensor. By communicating with the different components, the developed system allows NAO robot to interact with children with the following skills: movement learning and generalization; movement imitation; movement classification.

The work was tested in real-life situations with 3 different users, among them a child. All the skills were tested and the system was able to perform the different tasks with the 3 users.

The system framework was developed in ROS and it comprises all the different modules. The implemented modules are: communication module, gesture learning and generalization module, imitation module, camera module and the robot module. ROS runs the modules in parallel making it possible for the system to interact between the different parts.

Respecting to the proposed objectives in this work it reached the following contributions:

- The system is able to learning and generalize a movement either in offline (kinestetic learning) or online (kinect gesture learning) contexts;
- The robot is able to imitate the generalized movements;
- The system is able to classify the movements on-the-fly and if they are recognized the robot is able to reproduce them.

Despite the satisfying results, there is always space for improvement. The system is already fitted for interacting with children but it needs extra features for being suited for children that suffer from cerebral palsy. The system should have extra security measures that can detect abnormal situations and in those cases it should be ale to warn someone.

It would be also interesting if the system could detect the children emotions and at the same time being capable of emote. This would highly improve the decision making process and it would be easier to engage the children attention. For example, if the system detected that the children is sad it could use the imitation module implemented in this work to make an imitation game. Also a following module would make the child-robot interaction more natural.

Also, the possibility of including a physical therapy module could be explored. The kinect RGB-D sensor would detect wrong movements and the robot would reproduce how to do them properly.

Bibliography

[1] J. maccormick, ?how does the kinect work?? http://pages.cs.wisc.edu/ ahmad/kinect.pdf , accessed on 05/08/2016.

[2] Nao h25-v4 documentation http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html , accessed on 20/08/2016.

[3] Nao python sdk, http://doc.aldebaran.com/1-14/dev/python/install_guide.html , accessed on 14/03/2016.

[4] Nao robot - learning from demonstrations (kinect- skeleton motion) https://www.youtube.com/watch?v=rj9xbku7mkc.

[5] Nao robot - learning from demonstrations (kinesthetic motion) https://www.youtube.com/watch?v=emamodvafe8.

[6] Openni's kinematic model of the human body http://www.mdpi.com/1424-8220/13/9/12406/html , accessed on 20/08/2016.

[7] Svm, Website: http://www.neural-forecasting.com/support_vector_machines.htm, accessed on 20/08/2016.

[8] Website: https://gihub.com/rsait/rsait_public_packages/blob/master/nao_rsait/nao_teleop/nao_teleop_gestur , accessed on 12/04/2016.

[9] Website: https://github.com/aharobot/yale-ros-pkg/blob/master/sandbox/imitation/nodes/angle_calculator.p , accessed on 6/04/2016.

[10] Website: https://msdn.microsoft.com/en-us/library/jj131033.aspx, accessed on 22/08/2016.

[11] Saminda Abeyruwan, Faisal Sikder, Ubbo Visser, and Dilip Sarkar. Activity monitoring and prediction for humans and nao humanoid robots using wearable sensors. In *FLAIRS Conference*, pages 342–347, 2015.

[12] Tony Belpaeme, Paul E Baxter, Robin Read, Rachel Wood, Heriberto Cuayáhuitl, Bernd Kiefer, Stefania Racioppa, Ivana Kruijff-Korbayová, Georgios Athanasopoulos, Valentin Enescu, et al. Multimodal child-robot interaction: Building social bonds. *Journal of Human-Robot Interaction*, 1(2):33–53, 2012.

[13] Tanya N. Beran, Alex Ramirez-Serrano, Otto G. Vanderkooi, and Susan Kuhn. Reducing children's pain and distress towards flu vaccinations: A novel and effective application of humanoid robotics. *Vaccine*, 31(25):2772 – 2777, 2013.

[14] Aude Billard. Robota: Clever toy and educational tool. *Robotics and Autonomous Systems*, 42(3):259–269, 2003.

[15] S. Calinon. *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009. EPFL Press ISBN978-2-940222-31-5, CRC Press ISBN 978-1-4398-0867-2.

[16] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286–298, 2007.

[17] Sylvain Calinon, Florent D'halluin, Eric L Sauser, Darwin G Caldwell, and Aude G Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics & Automation Magazine*, 17(2):44–54, 2010.

[18] Kuniyoshi Y Cheng G. Extracting data from human manipulation of objects towards improving autonomous robotic grasping. *Proceedings of the sixth international conference on intelligent autonomous systems (IAS 2000)*, pages 273–280, 2000.

[19] Kenneth Mark Colby. The rationale for computer-based treatment of language difficulties in non-speaking autistic children. *Journal of autism and childhood schizophrenia*, 3(3):254–260, 1973.

[20] V. Cortes, C.; Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[21] Kerstin Dautenhahn. Socially intelligent robots: dimensions of human–robot interaction. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 362(1480):679–704, 2007.

[22] Alessandro Fantoni, Artur J. Ferreira, Jose Rosado, Filipe Silva, and Vitor Santos. Conference on electronics, telecommunications and computers ? cetc 2013. using kinect for robot gesture imitation. *Procedia Technology*, 17:423 – 430, 2014.

[23] Diego R. Faria, Cristiano Premebida, and Urbano Nunes. A probalistic approach for human everyday activities recognition using body motion from RGB-D images. In *IEEE RO-MAN'14, * Kazuo Tanie Award Finalist*, 2014.

[24] Diego R. Faria, Mario Vieira, Cristiano Premebida, and Urbano Nunes. Probabilistic human daily activity recognition towards robot-assisted living. In *IEEE RO-MAN'15: IEEE International Symposium on Robot and Human Interactive Communication. Kobe, Japan, August*, 2015.

[25] David Feil-Seifer and Maja Mataric'. Robot-assisted therapy for children with autism spectrum disorders. In *Proceedings of the 7th International Conference on Interaction Design and Children*, IDC '08, pages 49–52, New York, NY, USA, 2008. ACM.

[26] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572, 1901.

[27] Jean-Luc Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *Speech and Audio Processing, IEEE Transactions on*, pages 291–298, 1994.

[28] Melissa Guo, Shuvajit Das, Jacob Bumpus, Esubalew Bekele, and Nilanjan Sarkar. Interfacing of kinect motion sensor and nao humanoid robot for imitation learning. 2013.

[29] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Trans. Neur. Netw.*, 13(2):415–425, March 2002.

[30] Hideki Kozima, Cocoro Nakagawa, and Yuriko Yasuda. Children?robot interaction: a pilot study in autism therapy. In C. von Hofsten and K. Rosander, editors, *From Action to Cognition*, volume 164 of *Progress in Brain Research*, pages 385 – 400. Elsevier, 2007.

[31] Hideki Kozima, Cocoro Nakagawa, and Yuriko Yasuda. Children?robot interaction: a pilot study in autism therapy. In C. von Hofsten and K. Rosander, editors, *From Action to Cognition*, volume 164 of *Progress in Brain Research*, pages 385 – 400. Elsevier, 2007.

[32] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif., 1951. University of California Press.

[33] Yongsheng Ou, Jianbing Hu, Zhiyang Wang, Yiqun Fu, Xinyu Wu, and Xiaoyun Li. A real-time human imitation system using kinect. *International Journal of Social Robotics*, 7(5):587–600, 2015.

[34] Syamimi Shamsuddin, Hanafiah Yussof, Luthffi Ismail, Fazah Akhtar Hanapiah, Salina Mohamed, Hanizah Ali Piah, and Nur Ismarrubie Zahari. Initial response of autistic children in human-robot interaction therapy with humanoid robot nao. In *Signal Processing and its Applications (CSPA), 2012 IEEE 8th International Colloquium on*, pages 188–193. IEEE, 2012.

[35] Robert J. Vanderbei. LOQO: an interior point code for quadratic programming. *Optimization Methods and Software*, 11(1-4):451–484, 1999.

[36] M. Viera. Recognition of daily activities and risk situations towards robot-assisted living. 2015.

[37] Iain Werry, Kerstin Dautenhahn, Bernard Ogden, and William Harwin. Can social interaction skills be taught by a social agent? the role of a robotic mediator in autism therapy. In *Cognitive technology: instruments of mind*, pages 57–74. Springer, 2001.

[38] Hanafiah Yussof, Muhammad Fahmi Miskon, Norjasweeen Abdul Malik, Hanafiah Yussof, Fazah Akhtar Hanapiah, Rabiatul Adawiah Abdul Rahman, and Husna Hassan Basri. 2015 ieee international symposium on robotics and intelligent sensors (ieee iris2015) human-robot interaction for children with cerebral palsy: Reflection and suggestion for interactive scenario design. *Procedia Computer Science*, 76:388 – 393, 2015.