UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

SCUOLA DI DOTTORATO DI RICERCA IN: Ingegneria dell'Informazione

INDIRIZZO: Scienza e Tecnologia dell'Informazione

CICLO: XXVI

ROBOT LEARNING BY OBSERVING HUMAN ACTIONS

Direttore della Scuola:	*Ch.mo Prof. Matteo Bertocco*
Coordinatore d'indirizzo:	*Ch.mo Prof. Carlo Ferrari*
Supervisore:	*Ch.mo Prof. Emanuele Menegatti*

Dottorando:
Stefano Michieletto

Abstract

Nowadays, robotics is entering in our life. One can see robot in industries, offices and even in homes. The more robots are in contact with people, the more requests of new capabilities and new features increase, in order to make robots able to act in case of need, help humans or be a companion. Therefore, it becomes essential to have a quick and easy way to teach new skills to robots. That is the aim of Robot Learning from Demonstration. This paradigm allows to directly program new tasks in a robot through demonstrations.

This thesis proposes a novel approach to Robot Learning from Demonstration able to learn new skills from natural demonstrations carried out from naive users. To this aim, we introduce a novel Robot Learning from Demonstration framework by proposing novel approaches in all functional sub-units: from data acquisition to motion elaboration, from information modeling to robot control.

A novel method is explained to extract 3D motion flow information from both RGB and depth data acquired by using recently introduced consumer RGB-D cameras. The motion data are computed over the time to recognize and classify human actions.

In this thesis, we describe new techniques to remap human motion to robotic joints. Our methods allow people to natural interact with robots by re-targeting the whole body movements in an intuitive way. We develop algorithm for both humanoids and manipulators motion and test them in different situations.

Finally, we improve modeling techniques by using a probabilistic method: the Donut Mixture Model. This model is able to manage several interpretations that different people can produce performing a task. The estimated model can also be updated directly by using new attempts carried out by the robot. This feature is very important to rapidly obtain correct robot trajectories by means of few human

demonstrations.

A further contribution of this thesis is the creation of a number of new virtual models for the different robots we used to test our algorithms. All the developed models are compliant with ROS, so they can be used to foster research in the field from all the community of this very diffuse robotics framework. Moreover, a new 3D dataset is collected to compare different action recognition algorithms. The dataset contains both RGB-D information coming directly from the sensor and skeleton data provided by a skeleton tracker.

Sommario

La robotica sta ormai entrando nella nostra vita. Si possono trovare robot nelle industrie, negli uffici e perfino nelle case. Pi i robot sono in contatto con le persone, pi aumenta la richiesta di nuove funzionalit e caratteristiche per rendere i robot capaci di agire in caso di necessit, aiutare la gente o di essere di compagnia. Perci essenziale avere un modo rapido e facile di insegnare ai robot nuove abilit e questo proprio l'obiettivo del Robot Learning from Demonstration. Questo paradigma consente di programmare nuovi task in un robot attraverso l'uso di dimostrazioni.

Questa tesi propone un nuovo approccio al Robot Learning from Demonstration in grado di apprendere nuove abilit da dimostrazioni eseguite naturalmente da utenti inesperti. A questo scopo, stato introdotto un innovativo framework per il Robot Learning from Demonstration proponendo nuovi approcci in tutte le subunit funzionali: dall'acquisizione dei dati allelaborazione del movimento, dalla modellazione delle informazioni al controllo del robot.

Allinterno di questo lavoro stato proposto un nuovo metodo per estrarre l informazione del flusso ottico 3D, combinando dati RGB e di profondit acquisiti tramite telecamere RGB-D introdotte di recente nel mercato consumer. Questo algoritmo calcola i dati di movimento lungo il tempo per riconoscere e classificare le azioni umane.

In questa tesi, sono descritte nuove tecniche per rimappare il movimento umano alle articolazioni robotiche. I metodi proposti permettono alle persone di interagire in modo naturale con i robot effettuando un re-targeting intuitivo di tutti i movimenti del corpo. stato sviluppato un algoritmo di re-targeting del movimento sia per robot umanoidi che per manipolatori, testando entrambi in diverse situazioni.

Infine, sono state migliorate le tecniche di modellazione utilizzando un metodo probabilistico: il Donut Mixture Model. Questo modello in grado di gestire le numerose interpretazioni che persone diverse possono produrre eseguendo un compito. Inoltre, il modello stimato pu essere aggiornato utilizzando direttamente tentativi effettuati dal robot. Questa caratteristica molto importante per ottenere rapidamente traiettorie robot corrette, mediante luso di poche dimostrazioni umane.

Un ulteriore contributo di questa tesi la creazione di una serie di nuovi modelli virtuali per i diversi robot utilizzati per testare i nostri algoritmi. Tutti i modelli sviluppati sono compatibili con ROS, in modo che possano essere utilizzati da tutta la comunit di questo framework per la robotica molto diffuso per promuovere la ricerca nel campo. Inoltre, stato raccolto un nuovo dataset 3D al fine di confrontare diversi algoritmi di riconoscimento delle azioni, il dataset contiene sia informazioni RGB-D provenienti direttamente dal sensore che informazioni sullo scheletro fornite da uno skeleton tracker.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robot Learning From Demonstration (RLfD) [9] [6] also called Imitation Learning is a programming paradigm that uses demonstrations in order to make a robot learn new tasks. Several approaches were adopted in RLfD: Schaal et al. [81] used motion primitives to encode learning data, Akgun et al. [3] extracted keyframes to correctly model a skill, Calinon et al. [16] proposed an Hidden Markov Model/Gaussian Mixture Regression technique to reproduce human demonstrations in a multiple constraints environment. In the last years, Robot Learning from Demonstration (RLfD) has become a major topic in robotics research. The main reason for this is that direct programming a robot motion can be a very difficult and time consuming task. Acquiring examples from humans provides a powerful mechanism to simplify the process of programming complex robot motions.

In this thesis, a novel approach to Robot Learning From Demonstration is exploited. We analyzed how humans approach robots, and in particular robot motion programming, in order to understand an effective way to let people teach new tasks to robots. We focus on allowing non expert users to naturally interact with robots to teach them new behaviors.

This capability is useful in both industrial and service robotics scenarios. In industrial robotics, a natural demonstration of the task could be the proper way to fast automatize the production process without the need of tedious and errors prone manual programming. In service robotics, the robot could be able to learn

new tasks by itself, interacting with non skilled users in very different circumstances.

In the past, different modalities have been used in Robot Learning From Demonstration to convey the demonstrations from the teacher to the robot: motion sensors [13], kinesthetic teaching [37], or vision systems [22]. Motion sensors have to be worn from the user to acquire proper information. In kinesthetic demonstration, instead, the motion are collected directly from the robot moved by the user. Naive users hardly deal with such methods, that are unnatural or uncomfortable. Moreover, as highlighted in [53], state of the art approaches are based on expensive or sophisticated hardware normally inappropriate for household applications. On the other hand, 2D vision systems are usually not diffuse in this field because they used to be often slow and sensitive to clutter and occlusions. In our work, we used a low-cost RGB-D sensor to provide 3D data to Robot Learning from Demonstration system.

The introduction of low-cost RGB-D sensors [36] with good resolution and frame-rate has generated a rapid boosting of computer vision algorithms to estimate human pose [85], skeleton tracking [44] and activities recognition [7]. In this work, novel action recognition algorithms have been developed to extract information about human motion by simply observing local flow in the actor body. Information have been collected from 3D motion flow extracted directly from the person point cloud, we compared these data with skeleton joint positions and orientations. Both 3D motion flow and skeleton joints has been analyzed using several descriptors and classifiers in order to compare the performances of these two methods. The considered techniques use RGB-D data in order to maintain the spatial information about motion with respect the best 2D algorithms, which extract local features from consecutive video frames classifying actions by a bag-of-words approach and do not take track of where these features have been extracted.

Spatial information can be used as input for motion re-targeting techniques making an avatar replicate the same poses performed by an actor. Computer graphics [31] [51] uses such kind of techniques to generate off-line feasible motion for virtual characters, but robotics requires on-line methods to be applied in dynamic environments, in which sensors can provide feedback to avoid dangerous collisions, follow moving objects, or react to changing requests.

In this work, we developed a motion re-targeting system motion able to work on-line to remap human movements to humanoid robots taking account of similar works, which are previously proposed by Pollard [77], Dariush [19], and Miura [64].

We also apply similar remapping techniques to industrial robots, and in particular manipulators. In the past, some attempts were done using an anthropomorphic mapping [23] [57]. These approaches force the demonstrator to act in a "robotic" way, but they can not be adopted if the user is acting in a natural way. For example, if the human knows that his arm directly controls a manipulator, the movement he perform will be concentrated on the arm, with no lower-body motion. A natural execution, instead, involves the whole body. In this thesis, we developed a novel mapping from the whole human body to a manipulator.

In this work a number of models suitable for Robot Learning by Demonstration are tested. Some of them are good for well formed correct demonstrations, some others are feasible to reconstruct trajectories from failed examples.

At first we develop an original version of a state-of-the-art framework based probabilistic method, namely Gaussian Mixture Model (GMM) and Gaussian Mixture Regression (GMR). Good results can be reached with this framework with a relative high number of demonstrations in high repeatable industrial tasks. On the other hand, the GMM/GMR framework can not handle the different interpretations feasible in a generic service robotics environment and another solution has to be found. In particular, this thesis deeply analyzed the possible capabilities of a novel Robot Leaning from Failure Demonstration [35]. We extended the Donut Mixture Model used in Grollman work in order to adapt human demonstrations to robot degrees of freedom. In fact, the mapping between human and robot joints can generate a failed robot attempt, even starting from a successful human example. The model estimated using our approach is then updated using the attempts performed by the robot. This feature is very important to rapidly obtain correct robot trajectories by means of few human demonstrations. The complete framework has been tested in several situations and with different robots. Tests have been done by using both kinesthetic demonstrations and RGB-D data.

1.1 Thesis Outline and Contributions

In Chapter 2, we will describe a novel approach to action recognition based on RGB-D data which exploits 3D motion flow algorithms to build up different kind of gridded descriptors that merge together color and depth information. We will compare this approach with a skeleton information based descriptor that recognizes human action estimating joint positions and orientations from the human body.

In Chapter 3, we will propose novel motion re-targeting techniques for both humanoids and manipulators by elaborating the same skeleton information used in the previous chapter. On one hand, the proposed techniques are thought to be used from naive users that act in a natural way. On the other hand, they are required to be very fast and robust in different situations. In both, humanoids and manipulators, the whole body will be involved in the re-targeting algorithm, so that humanoids re-targeting will take care of robot stability, while manipulators re-targeting will exploit a remapping from human body to a non humanoid structure.

The motion re-targeting techniques will lead to a novel approach to Robot Learning from Demonstration, which will be described in Chapter 4. In Robot Learning from Demonstration, users usually are constrain to modalities understandable by the robot. In this work we will remove such constrain in order to let the demonstrator act as natural as possible. An initial framework based on Gaussian Mixture Model and Gaussian Mixture Regression will be applied on an industrial task. While a more flexible Donut Mixture Model framework will be tested in service robotics environment.

Finally, Chapter 5 will conclude the thesis and summarizes the results achieved.

Appendix A will describe a further contribution of this thesis, which is the creation of a new RGB-D action recognition dataset called *IAS-Lab Action Dataset*. This dataset has been use to perform the comparison between the action recognition techniques developed in this thesis. A brief presentation of the robots used in this work will be listed in Appendix B. A virtual model has been developed for most of them in order to test the algorithms proposed in this thesis in a simulated environment. Both dataset and virtual models have been publically released in order to foster research in the field.

1.2 Publications

The work described in this thesis has also been presented in the publications listed here below, divided by topic.

People detection and tracking

- [30] S. Ghidoni, S. M. Anzalone, M. Munaro, S. Michieletto and E. Menegatti. *A distributed perception infrastructure for robot assisted living.* To appear in Robotics and Autonomous Systems (RAS) Journal, Elsevier, 2014.

- [68] M. Munaro, F. Basso, S. Michieletto, E. Pagello and E. Menegatti. *A software architecture for RGB-D people tracking based on ROS framework for a mobile robot.* Frontiers of Intelligent Autonomous Systems, Volume 466, pp 53-68, Springer 2013.

- [8] F. Basso, M. Munaro, S. Michieletto and E. Menegatti. *Fast and robust multi-people tracking from RGB-D data for a mobile robot.* In Proceedings of the 12th Intelligent Autonomous Systems (IAS) Conference, Jeju Island (Korea), 2012

Robot Learning by Demonstration

- [59] S. Michieletto, S. Ghidoni, E. Pagello, M. Moro and E. Menegatti. *Why teach robotics using ROS*. To appear in Journal of Automation, Mobile Robotics & Intelligent Systems (JAMRIS). 2014.

- [10] A. Bisson, A. Busatto, S. Michieletto and E. Menegatti. *Stabilize humanoid robot teleoperated by a RGB-D sensor*. Popularize Artificial Intelligence (PAI2013). 2013.

- [79] G. Pozzato, S. Michieletto and E. Menegatti. *Towards smart robots: rock-paper-scissors gaming versus human players*. Popularize Artificial Intelligence (PAI2013). 2013.

- [62] S. Michieletto, D. Zanin and E. Menegatti. *NAO robot simulation for service robotics purposes.* European Modelling Symposium EMS2013 (EMS2013). 2013.

- [58] S. Michieletto, N. Chessa and E. Menegatti. *Learning how to approach industrial robot tasks from natural demonstrations*. IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO2013). 2013.

- [61] S. Michieletto, A. Rizzi and E. Menegatti. *Robot learning by observing humans activities and modeling failures*. IROS workshops: Cognitive Robotics Systems (CRS2013). 2013.

- [60] S. Michieletto and E. Menegatti. *Human action recognition oriented to humanoid robots action reproduction*. In Matteo Baldoni, Federico Chesani, Bernardo Magnini, Paola Mello, Marco Montai (eds.), Popularize Artificial Intelligence, proceedings of the AI*IA Workshop and Prize for Celebrating 100th Anniversary of Alan Turing's Birth (PAI. 2012), Rome, Italy, June. 2012. pp. 35-40.

Action recognition

- [66] M. Munaro, G. Ballin, S. Michieletto and E. Menegatti. *3D flow estimation for human action recognition from colored point clouds.* Journal on Biologically Inspired Cognitive Architectures, vol. 5, pp 42-51, 2013.

- [69] M. Munaro, S. Michieletto and E. Menegatti. *An evaluation of 3D motion flow and 3D pose estimation for human action recognition.* In Proceedings of Robotics Science and Systems 2013: Workshop on RGB-D - Advanced Reasoning with Depth Cameras, Berlin (Germany), 2013.

Chapter 2

Action Recognition

In recent years, robotics perception has grown very fast, introducing new applications that were considered unfeasible before. This success has been fostered by the introduction of RGB-D sensors with good resolution and framerate [1] and open source software for robotics development [80]. Thanks to these progresses, we can now think about robots capable of smart interaction with humans. One of the most important skills for a robot interacting with a human is the ability to recognize what the human is doing. For instance, a robot with this skill could assist elderly people by monitoring them and understanding if they need help or if their actions can lead to a dangerous situation.

In this work a novel method for real time 3D flow estimation from point cloud data is presented in which the whole person motion is encoded by using a 3D grid-based descriptor. Our 3D motion flow technique is compared with features based on skeleton information by means of a newly created dataset which contains RGB-D and skeleton data for 15 actions performed by 12 different actors.

The remainder of the chapter is organized as follows: Section 2.1 provides a review about the recent advances in human action recognition systems. In Section 2.2, the 3D motion flow estimation algorithm is described, while in Section 2.3 the characteristics of the skeleton data are explained. In Section 2.4 we detail the descriptors used for encoding person motion and skeletal information. Section 2.5 reports experiments on the *IAS-Lab Action Dataset* and on the dataset used in [7], while Section 2.6 summarize the the chapter and outlines the future

work.

2.1 Related Work

The first RGB-D related work is signed by Microsoft Research [54]. In [54], the relevant postures for each action are extracted from a sequence of depth maps and represented as bags of 3D points. The motion dynamics are modeled by means of an action graph and a Gaussian Mixture Model is used to robustly capture the statistical distribution of the points. Subsequent studies mainly refer to the use of three different sensor technologies: Time of Flight cameras [38, 39], motion capture systems [73], [47], [88] and active matricial triangulation systems (i.e.: Kinect-style cameras) [87], [96], [97], [71], [78], [52], [11], [98], [63], [93]. The most used features are related to the extraction of the skeleton body joints [87], [96], [11], [73], [93], [88]. Usually, these approaches first collect raw information about the body joints (e.g.: spatial coordinates, angle measurements). Next, they summarize the raw data into features, in order to characterize the posture of the observed human body. Differently from the other joints-related publications, [73] computes features which carry a physical meaning. Indeed, in [73], a Sequence of Most Informative Joints (SMIJ) is computed based on measures like the mean and variance of joint angles and the maximum angular velocity of body joints.

Other popular features are the result of the extension to the third dimension of typical 2D representations. Within this category, we should distinguish between local and global representations. Features in [97], [71], [98], [63] are local representations since they aim to exploit the well-known concept of STIPs [49, 50] by extending it with depth information. Examples of global representations in the 3D domain can be found in [78], [38, 39]. In [78], Popa *et al.* propose a Kinect-based system able to continuously analyze customers' shopping behaviours in malls. Silhouettes for each person in the scene are extracted and then summarized by computing moment invariants. In [38, 39], a 3D extension of 2D optical flow is exploited for the gesture recognition task. Holte *et al.* compute optical flow in the image using the traditional Lukas-Kanade method and then extend the 2D velocity vectors to incorporate also the depth dimension. At the end of this process, the 3D velocity vectors are used to create an annotated velocity cloud. 3D

Motion Context and Harmonic Motion Context serve the task of representing the extracted motion vector field in a view-invariant way. With regard to the classification task, [38] and [39] do not follow a learning-based approach, instead a probabilistic Edit Distance classifier is used in order to identify which gesture best describes a string of primitives. [39] differs from [38] because the optical flow is estimated from each view of a multi-camera system and is then combined into a unique 3D motion vector field.

Finally, works in which trajectory features are exploited [52], [47] recently emerged. While in [52] trajectory gradients are computed and summarized, in [47], an action is represented as a set of subspaces and a mean shape.

Unlike [38] and [39], which compute 2D optical flow and then extend it to 3D, a method to compute the motion flow directly on 3D points with color has been proposed in [7]. From the estimated 3D velocity vectors, a motion descriptor is derived and a sequence of descriptors is concatenated and classified by means of Nearest Neighbor. Tests are reported on a dataset of six actions performed by six different actors.

2.2 3D Motion Flow

Optical flow is a powerful cue to be used for a variety of applications, from motion segmentation to structure-from-motion passing by video stabilization. As reported in Section 2.1, some researchers proved its usefulness also for the task of action recognition [24], [94], [4]. The most famous algorithm for optical flow estimation was proposed by Lukas and Kanade [56]. The main drawbacks of this approach were that it only works for highly textured image patches and, if repeated for every pixel of an image, it results to be highly computational expensive. Moreover, 2D motion estimation in general has the limitation to be dependent on the viewpoint and closer objects appear to move faster because they appear bigger in the image.

When depth data are available and registered to the RGB/intensity image, the optical flow computed in the image can be extended to 3D by looking at the corresponding points in the depth image or point cloud [38, 39]. This procedure allows to compute 3D velocity vectors, thus overcoming some of the limitations of 2D-

only approaches, such as viewpoint and scale dependence. However, the motion estimation process is still completely based on the RGB image and it does not exploit the available 3D information for obtaining a better estimate. Moreover, the computational onerosity is still high.

In this work, we improve the technique recently proposed in [7] for computing 3D motion of points in the 3D-color space directly. This method consists in estimating correspondences between points of clouds belonging to consecutive frames. Our approach is fast and able to overcome some singularities of optical flow estimation in images by relying also on 3D points coordinates. Moreover, it is applicable to any point cloud containing XYZ and RGB information, and not only to those derived from a 2D matrix of depth data (projectable point clouds).

Given two point clouds (called *source* and *target*) containing 3D coordinates and RGB/HSV color values of an object of interest (in this work, a person), the following pipeline is applied:

1. correspondence finding: for every point of the target point cloud, we select K nearest neighbors in the source point cloud in terms of Euclidean distance in the XYZ space; among the resulting points, we select the nearest neighbor in terms of HSV coordinates. We preferred HSV to RGB because it is more perceptually uniform. If $\mathcal{N}_{\mathbf{p}_i^{target}}$ is the set of K nearest neighbors in the source point cloud to the point $\mathbf{p}_i$ in the target point cloud, then $\mathbf{p}_i^{target}$ is said to match with

$$\mathbf{p}_*^{source} = \underset{\mathbf{p}_i^{source} \in \mathcal{N}_{\mathbf{p}_i^{target}}}{\operatorname{argmin}} d_{HSV}\left(\mathbf{p}_i^{target}, \mathbf{p}_i^{source}\right), \tag{2.1}$$

   where d_{HSV} is the distance operator in the HSV space. The number of neighbors K is a function of the point cloud density. In this work, we filter the point clouds to have a voxel size of $0.02m$ and we set K to 50.

2. outlier rejection by means of reciprocal correspondences: this method consists in estimating correspondences from target to source and from source to target. Then, points which match in both directions are kept.

3. computation of 3D velocity vectors $\mathbf{v}_i$ for every match i as spatial displace-

ment over temporal displacement of corresponding 3D points $\mathbf{p}_i$ from target and source:

$$\mathbf{v}_i = \left(\mathbf{p}_i^{target} - \mathbf{p}_i^{source}\right) / \left(t_i^{target} - t_i^{source}\right) \tag{2.2}$$

4. unlike in [7], we perform an additional outlier rejection: points with 3D velocity magnitude $\|\mathbf{v}_i\|$ below a threshold are discarded. Isolated moving points (not near to other moving points) are also deleted. In particular, points moving faster than 0.3 m/s are retained and a moving point is considered to be isolated if none of its neighbors moves faster than 0.75 m/s.

The reciprocal correspondence technique for outlier rejection can be considered as a 3D extension of the *Template Inverse Matching* method [55], which has been widely used to estimate the performance of 2D optical flow estimation. The constraints we apply on the flow magnitude and on the proximity to other moving points are meant to remove spurious estimates which can be generated from the noise inherent in the depth values.

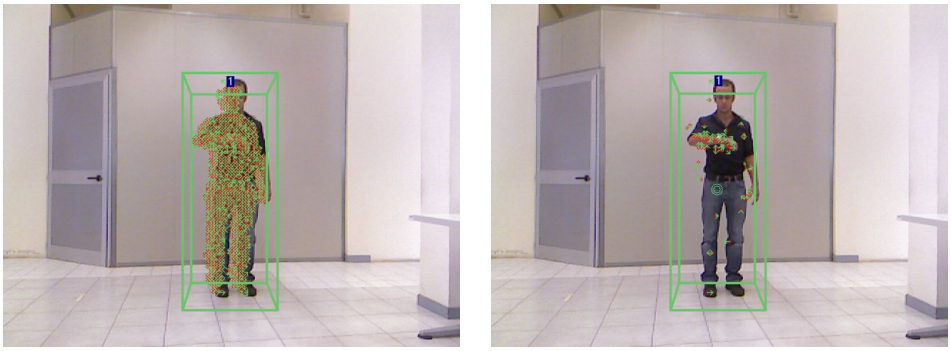
In this work, we segment point clouds to isolate humans from the rest of the scene by means of the people detection and tracking method for RGB-D data described in [67] and then we apply the flow estimation algorithm to the detected the clusters containing a person.

In Figure 2.1, we report two consecutive RGB frames of a person performing the *Check Watch* action. Green arrows show magnitude and direction of the estimated flow when reprojected to the image. It can be noticed how outlier rejection successfully removes the majority of the noisy measurements, while preserving the real motion at the right arm position.

2.3 3D Pose

In this work skeletal tracking capabilities comes from a software development kits released by OpenNI. In particular, the skeletal tracking algorithm is implemented in a freeware middleware called NiTE[1] built on top of the OpenNI SDK.

[1] `http://www.primesense.com/solutions/nite-middleware`



(a) Before (b) After

Figure 2.1: Example of 3D flow estimation results reprojected to the image (a-b) for action *Check watch*. Flow is visualized as green arrows in the image, before (a) and after (b) outlier removal.

NiTE uses the information provided by a RGB-D sensor to estimate the position of several joints of the human body. The framerate working on CPU at 30 Hz that means a very good performance in terms of precision and velocity with respect to its image-based counterparts. Since this technique is based on depth data, it is also invariant to illumination changes.

NiTE skeletal tracker is able tracks people in the range 1.2-3.5 m by using a people detection algorithm based on motion detection and exploiting the information from multiple depth frames to improve the tracking performance. As drawback, a person has to move at startup for being detected. It can track upto 9 skeletons and 15 joints for each person providing a label for every joint stating if it is tracked or inferred. Differently from the skeletal tracker implemented in the Microsoft Kinect SDK[2] that is trained on frontal people, NiTE skeletal tracker is able to track also people seen from the back side. On the other hand, it often poorly estimates the whole skeleton when some joints are not visible. For this reason, when using NiTE skeletal tracker in this work, we used only frames with all joints marked as tracked.

[2]http://www.microsoft.com/en-us/kinectforwindows/develop

2.4 Descriptors

In this section, we describe the frame-wise and sequence-wise descriptors we extract for describing actions.

2.4.1 SUMFLOW

In order to compute a descriptor accounting for direction and magnitude of motion of every body part, we center a 3D grid of suitable dimensions around a person point cloud. This grid divides the space around the person into a number of cubes. In Figure 2.2, a person point cloud is reported, together with the 3D grid which divides its points into different clusters represented with different colors. The size of the grid is proportional to the person's height in order to contain the whole limbs motion and to make the flow descriptor person-independent.

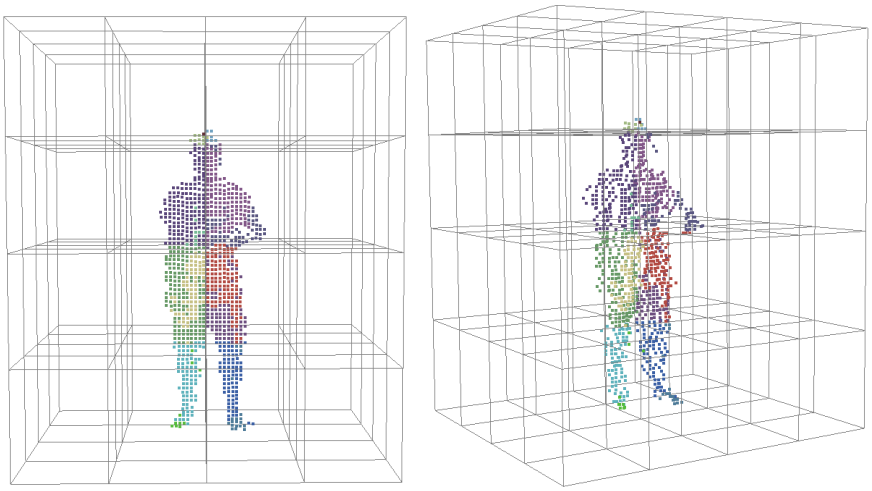


Figure 2.2: Two different views of the computed 3D grid: 4 partitions along the x, y and z axis are used.

For every cube of the grid, we extract flow information from all the points inside the cube. Unlike [7], which exploits the mean flow vector of every cube, we compute the sum of the motion vectors of every cube. This choice is due to the fact that the mean would amplify the noise contribution when little motion is present. The resulting vectors for all the cubes are concatenated into a single descriptor which is then L2-normalized for making it invariant to the speed at

which an action is performed. We will refer to our descriptor as the *SUMFLOW* descriptor. If this work, the grid is divided into four parts in every dimension, thus the total number of cubes is $C = 64$. If $x_i^{sF}, y_i^{sF}, z_i^{sF}$ are the coordinates of the flow sum vector for the $i-th$ cube, the SUMFLOW descriptor can be written as

$$\mathbf{d}_{SUMFLOW} = \left[x_1^{sF}\ y_1^{sF}\ z_1^{sF}\ \ldots\ x_C^{sF}\ y_C^{sF}\ z_C^{sF}\right]. \tag{2.3}$$

2.4.2 Skeleton Descriptor

The skeleton information provided by the NITE middleware consists of $N = 15$ joints from *head* to *foot*. Each joint is described by position (a point in 3D space) and orientation (a quaternion). On these data, we perform two kinds of normalization: the former scales the joints positions in order to report the skeleton to a standard height, thus achieving invariance to people height, the latter makes every feature to have zero mean and unit variance. Starting from the normalized data, we extracted three kinds of descriptors: a first skeleton descriptor ($\mathbf{d}_P$) is made of the set of joints positions concatenated one to each other; for the second one ($\mathbf{d}_O$), normalized joints orientations are gathered. Finally, we tested also a descriptor ($\mathbf{d}_{TOT}$) concatenating both position and orientation of each normalized joint:

$$\mathbf{d}_P = [x_1\ y_1\ z_1\ \ldots\ x_N\ y_N\ z_N], \tag{2.4}$$

$$\mathbf{d}_O = \left[q_1^1\ q_1^2\ q_1^3\ q_1^4\ \ldots\ q_N^1\ q_N^2\ q_N^3\ q_N^4\right], \tag{2.5}$$

$$\mathbf{d}_{TOT} = \left[\mathbf{d}_P^1\ \mathbf{d}_O^1\ \ldots\ \mathbf{d}_P^N\ \mathbf{d}_O^N\right]. \tag{2.6}$$

2.4.3 Sequence Descriptor

Since an action actually represents a sequence of movements over time, the use of multiple frames can provide more discriminant information to the recognition task with respect to approaches in which only a single-frame classification is performed. For this reason, we compose a single descriptor from every pre-segmented sequence of frames to be classified. In particular, we select a fixed number of frames evenly spaced in time from every sequence and we concatenate the single-frame descriptors to form a single sequence descriptor. Thanks to

this approach, we take into account the temporal order in which the single frame descriptors occurs.

2.5 Experiments

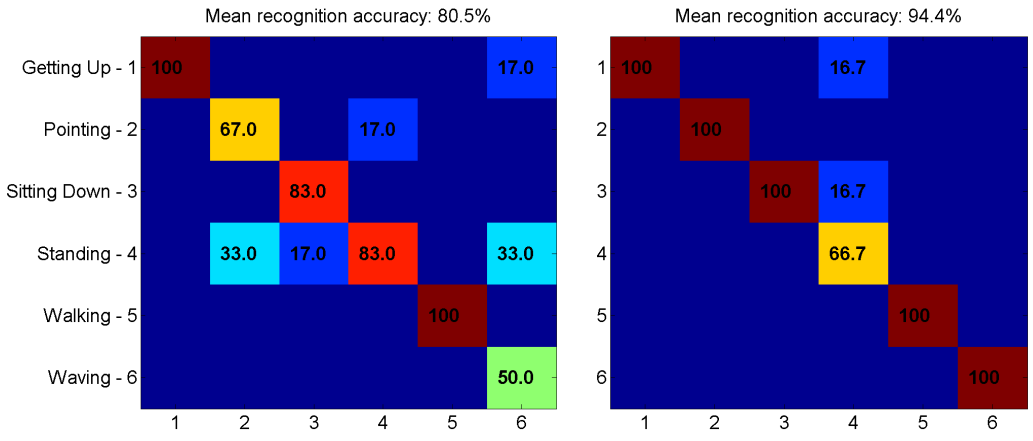
In this section, we report the human action recognition experiments we performed by exploiting the descriptors presented in Section 2.4.

We used the people detection and tracking algorithm described in [67] for segmenting people point clouds out from raw Kinect data. That method also performs a voxel grid filtering of the whole point cloud in order to reduce the number of points that should be handled. It is worth noting that a voxel size of 0.06m proved to be ideal for people tracking purposes, but it resulted to be insufficient for capturing local movements of the human body useful in an action recognition context. For this reason, we chose the voxel size to be of 0.02m.

We first evaluated our 3D motion flow descriptor on the action dataset used in [7]. That dataset contains six types of human actions: *getting up, pointing, sitting down, standing, walking, waving*. Each action is performed once by six different actors and recorded from the same point of view. Every action is already segmented out into a video containing only one action. Each of the segmented video samples spans from about 1 to 7 seconds. Unfortunately, no skeleton information is provided, thus the skeleton-based descriptors could not be tested.

For assigning an action label to every test sequence, we performed Nearest Neighbor classification with a *leave-one-person-out* approach, that is we trained the actions classifiers on the videos of all the persons except one and we tested on the video containing the remaining person. Then, we repeated this procedure for all the people and we computed the mean of all the rounds for obtaining the mean recognition accuracy. In Figure 2.3(b), we report the confusion matrix obtained on this dataset with our approach based on the *SUMFLOW* descriptor when using 10 frames for composing the sequence descriptor. The mean accuracy is 94.4% and the only errors occur for recognizing the *standing* action, which is sometimes confused with the *getting up* and *sitting down* actions. The accuracy we obtained in this work is considerably higher than what obtained in [7] (Figure 2.3(a)), which was of 80.5%. This improvement is due to: The outlier rejection performed after

the motion flow estimation, the use of the *SUMFLOW* descriptor, which is less sensitive to noise than the one in [7], and the choice of the frames which are concatenated to compose the sequence descriptor. In fact, we select frames evenly spaced within a sequence, while, in [7], the central frames of every sequence were chosen, thus encoding only the central part of an action.

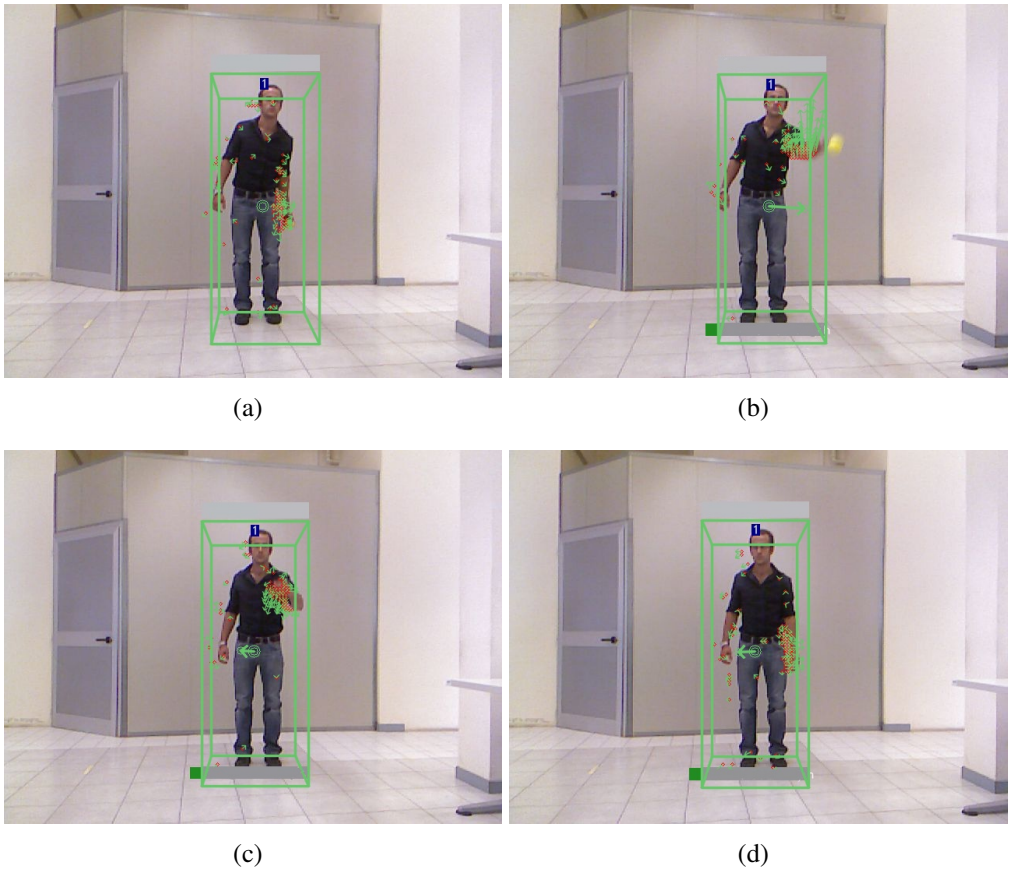


(a) [7] (b) This work

Figure 2.3: Confusion matrix obtained on the dataset presented in [7].

The single contributions of this work have been also evaluated on the *IAS-Lab Action Dataset*(Appendix A). In Figure 2.4, we show an example of 3D flow estimation for some key frames of the *Throw from bottom up* action. We adopted the same leave-one-person-out approach described above for computing the recognition accuracy.

In Figure 2.5, we report the mean recognition accuracy obtainable on the *IAS-Lab Action Dataset* when using the SUMFLOW frame-wise descriptor and varying the number of frames used for composing the sequence descriptor. It can be noticed how the accuracy rapidly increases until 5 frames per sequence and continues to considerably improve until 30 frames are used, reaching a recognition rate of 85.2%, which can be considered as a very good score given that the people used as test set were not present in the training set. In the confusion matrices shown in Figure 2.6, 30 frames have been selected from every action sequence to compute the sequence descriptors. It can be noticed how the *SUMFLOW* descrip-



(a) (b)

(c) (d)

Figure 2.4: Example of 3D flow estimation for some key frames of the *Throw from bottom up* action of the *IAS-Lab Action Dataset*.

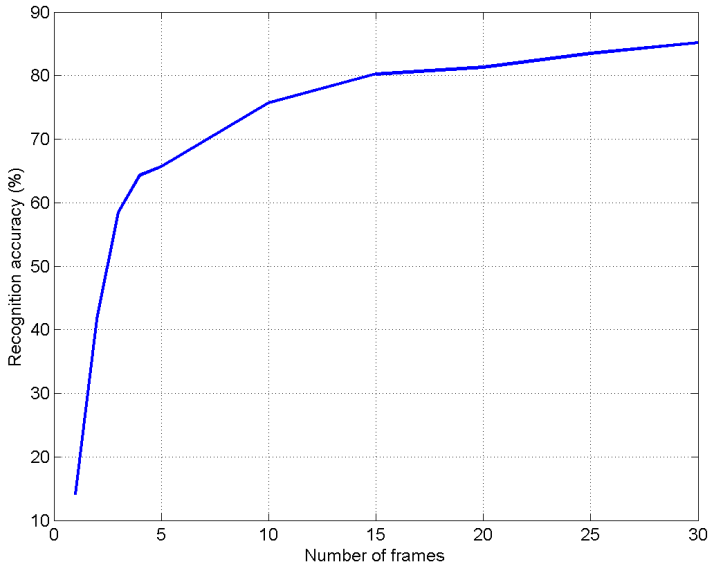


Figure 2.5: Mean recognition accuracy obtained with the SUMFLOW descriptor on the *IAS-Lab Action Dataset* when varying the number of frames used for composing the sequence descriptor.

tor (Figure 2.6(c)) reaches the best recognition accuracy of 85.2%. Most of errors occurred for the action *Point* and *Wave*, and in particular actions with little motion are sometimes confused with the *Standing* action. As a reference, Figure 2.6(a) shows the confusion matrix obtained with the descriptor in [7] and (b) reports the results obtained with the SUMFLOW descriptor if outlier rejection is not applied in the motion flow estimation process. The clear drop in performance, of 27.2% and 3.3% respectively, confirms the validity of the choices made in this work. It is worth noting that, even if the *Standing* action is not included in all the datasets we reported in Section A.1, it is very important for the task of action detection: an algorithm able to reliably distinguish this action from the rest could be easily extended to detect actions from an online stream, rather than needing pre-segmented sequences.

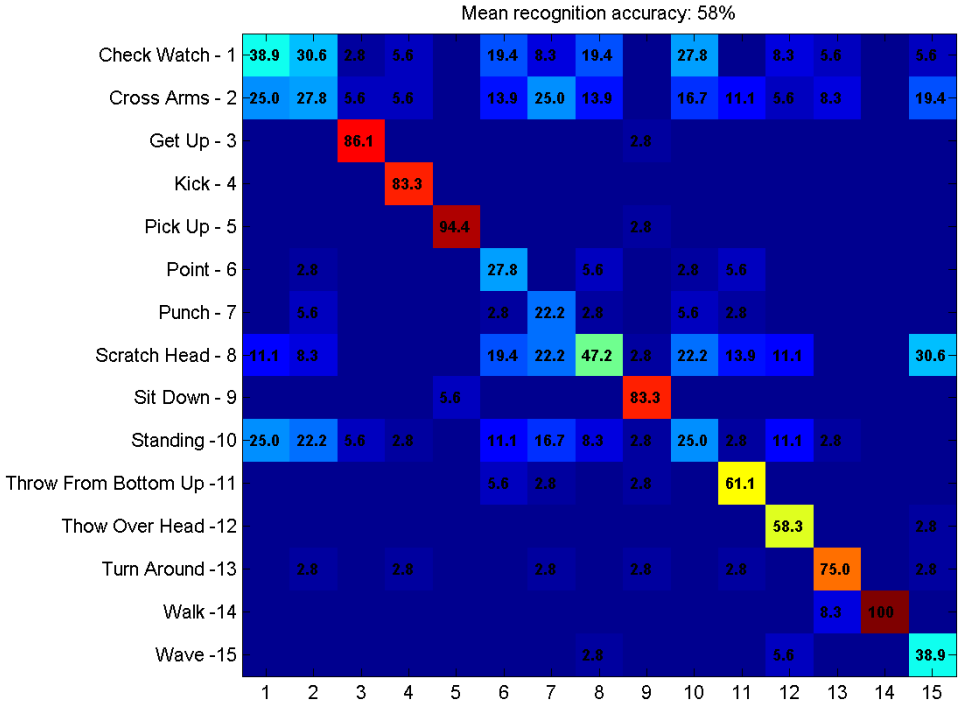
For what concerns the skeleton-based descriptors, the classification of the joints orientation and position descriptors reaches, respectively, 55.9% and 76.7% accuracy, while the combined use of joints angles and positions leads to a result

which is in the middle of the two, namely 66.9%. In Figure 2.6(b), the confusion matrix relative to the joints position descriptor is reported. The majority of errors obtained are due to the fact that the *Standing*, *Turn Around* and *Walk* actions are featured by very similar skeleton poses. These results prove that 3D local motion is highly discriminative for the action recognition task and can also lead to better results than those which can be obtained by exploiting skeleton information. This is also due to the fact that the noise intrinsic in a consumer depth camera and some challenging human poses can make the skeleton to be sometimes unreliably estimated.

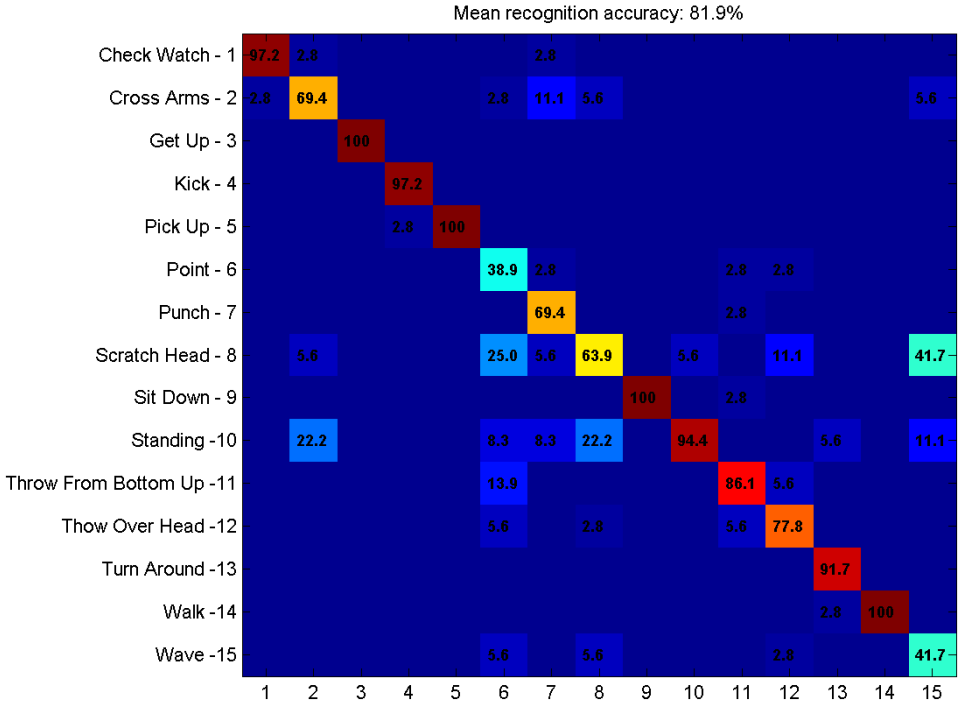
It is worth noting that both the approaches we considered in this work would obtain similar performances also in cluttered environments or when the background is less clean than in the *IAS-Lab Action Dataset*. The background is removed since only moving points of a person or its skeleton are considered for computing a descriptor.

In Figure 2.7, we report the mean recognition accuracy obtainable on the *IAS-Lab Action Dataset* when varying the number of frames used for composing the sequence descriptor. By comparing the curves with and without PCA projection, it can be noticed how the accuracy obtainable without PCA and 30 frames per sequence can be obtained with PCA and half (15) of the frames, thus allowing faster comparison between sequence descriptors.

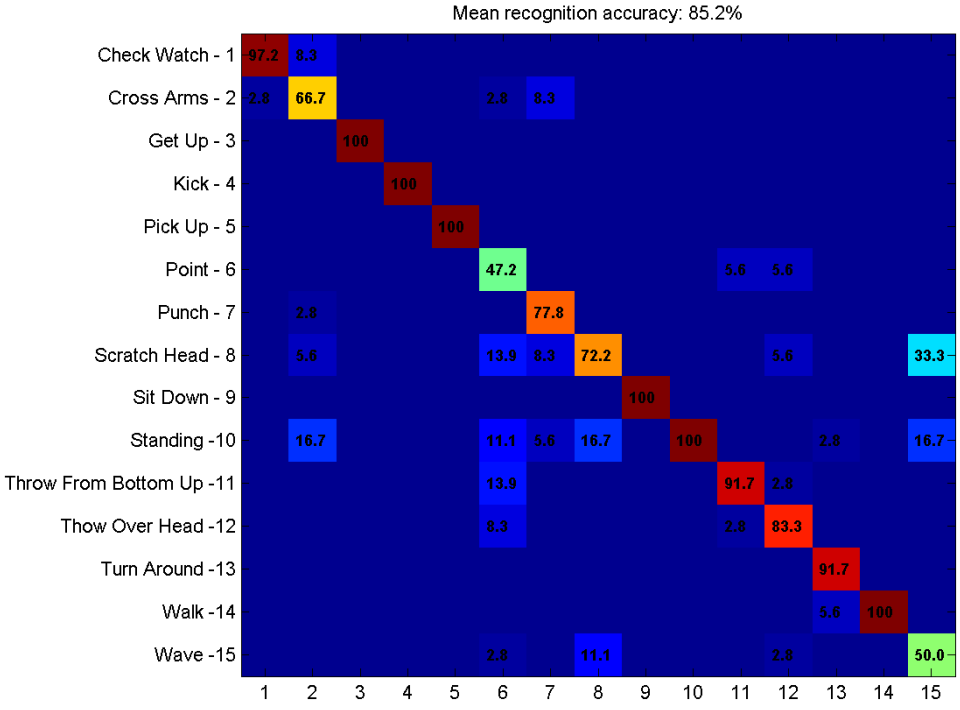
In terms of runtime performance, the skeleton description is very fast to compute, because all the information is already provided by the skeletal tracker algorithm. Instead, the overall runtime of the 3D motion-based classification approach is of about 0.25s, meaning a framerate of 4 frames per second on a notebook with an Intel i7-620M 2.67GHz processor and 4GB of RAM, which suggests that an optimized version of this algorithm could be executed onboard of robots with limited computational resources. The most demanding operation is the matching between the previous and current point clouds, that is the search for correspondences.



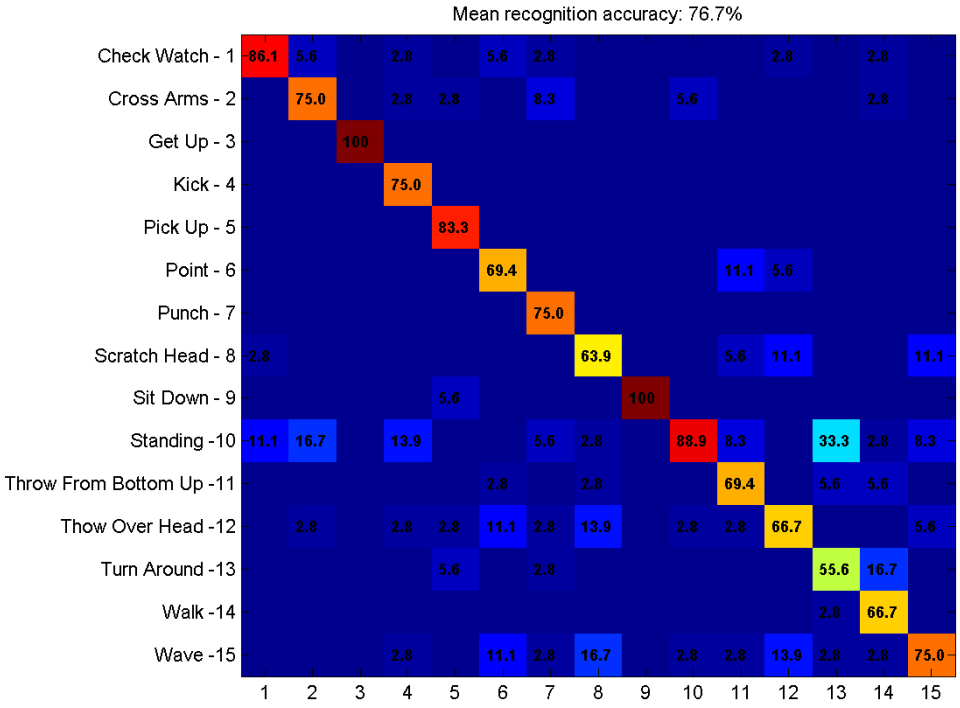
(a) Descriptor in [7]



(b) SUMFLOW without outlier rejection



(c) SUMFLOW



(d) Skeleton joints position descriptor $\mathbf{d}_P$

Figure 2.6: Confusion matrix obtained on the *IAS-Lab Action Dataset* with the descriptor in [7], our SUMFLOW descriptor (b) without and (c) with outlier rejection and (d) the skeleton-based descriptor.

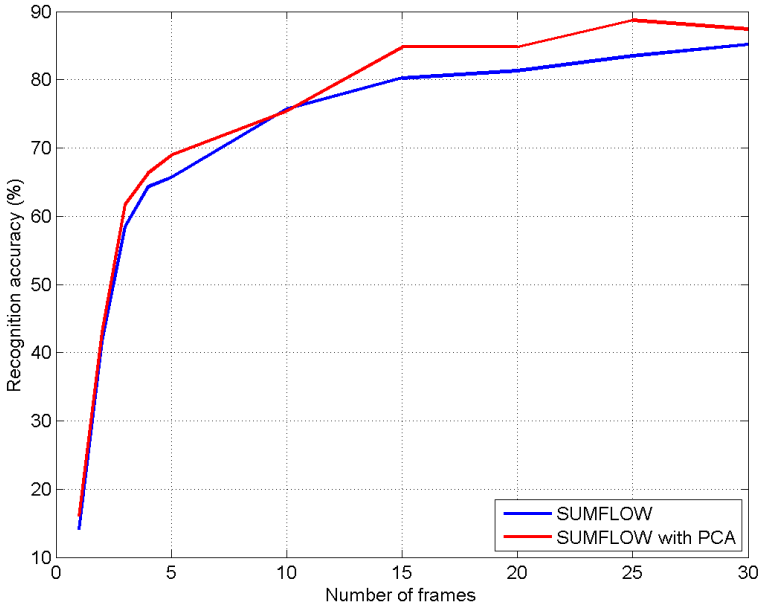


Figure 2.7: Mean recognition accuracy when varying the number of frames used for composing the sequence descriptor.

2.6 Summary

In this chapter, we presented a novel method for real-time estimation of 3D motion flow from colored point clouds and a complete system for human action recognition which exploits this motion information. Moreover, we compared this method with an action recognition technique which classify the skeleton information on a newly created dataset with a high number of people performing the actions and providing both RGB-D data and skeleton pose for every frame. The tested 3D flow technique reported very good results in classifying all the actions of the dataset, reaching 85.2% of accuracy and outperforming of 8.5% the skeleton-based method.

As future works, on one hand we plan to extend our 3D flow-based action recognition approach in order to make it work from a mobile robot in conjunction with our people tracking and re-identification system. On the other hand we schedule to test histogram-based descriptors for encoding 3D motion information inside each grid partition.

Chapter 3

Motion re-targeting

Motion re-targeting techniques capture information from an actor and replicate them on an avatar assuming similar poses. In this work, we developed novel mapping from the whole human body to two different kind of robots: humanoids and manipulators. The aim is to use the human joints remapped on robots as input of a Learning from Demonstration framework.

First works about motion re-targeting was developed in the Computer Graphics community. In this field such kind of techniques are used to generate off-line feasible motion for virtual characters [31] [51]. Computer Graphics require very sophisticated algorithms involving the whole body of both actor and avatar. Instead, Robotics requires on-line methods to be applied in dynamic environments, in which sensors can provide feedback to avoid dangerous collisions, follow moving objects, or react to changing requests.

On-line robotics-oriented systems have been developed to re-target motion from human to humanoids [77] [19] [64] or from human to robot limbs [23] [57]. In the cited works, the mapping from human to robot is anthropomorphous. This practice is good if the demonstrator is conscious to teach tasks to a robot, while a different approach should be adopted if he is acting in a natural way. For example, if the human knows that his arm directly controls a manipulator, the movement he perform will be concentrated on the arm, with no lower-body motion. A natural execution, instead, involves the whole body.

The reminder of the chapter is organized as follow. In Section 3.1, the data

acquisition system is presented. Section 3.2 describe the approach we used in humanoids remapping, while the manipulator motion re-targeting technique is reported in Section 3.3. A brief summary of the chapter is contained in Section 3.4.

3.1 Data acquisition

The aim in this work is to make a robot learn a new task from human demonstrations and one of our main objective is to let the demonstrator act as more natural as possible, thus choosing the right acquisition system is a crucial step. Different data acquisition systems can be used to convey the demonstrations from users to the robot: motion sensors [13], kinesthetic teaching [37], or vision systems [22].

A visual technique fits our needs better than a kinesthetic demonstration because the user is not required to rethink the task from a robot point of view. By using a motion capture system (MoCap) we can achieve a great accuracy in data acquisition, but the demonstrator have to wear some markers in order to make the system works. In some circumstances a marker system could be unwieldy or dangerous (i.e. in an industrial environment). Naive users hardly deal with such methods, that are unnatural or uncomfortable.

We propose to use a RGB-D sensor able to acquire the scene at 30 fps with a resolution of 640x480 pixels for both color and depth information. Recently, a large variety of computer vision algorithms using RGB-D sensors has been developed to estimate human pose [85], perform skeleton tracking [44] and recognize actions and activities [66]. We used a skeleton tracking algorithm [43] running at 30 fps to acquire joint positions and orientations (Figure 3.1) subsequently processed to understand the motion and guide the robot in the learning phase.

3.2 Human to humanoids re-targeting

3.2.1 Motion evaluation

In this subsection human movements are mapped on a small humanoid, namely the Vstone Robovie-X (Section B.3). Referring to Figure 3.1, we chose the refer-

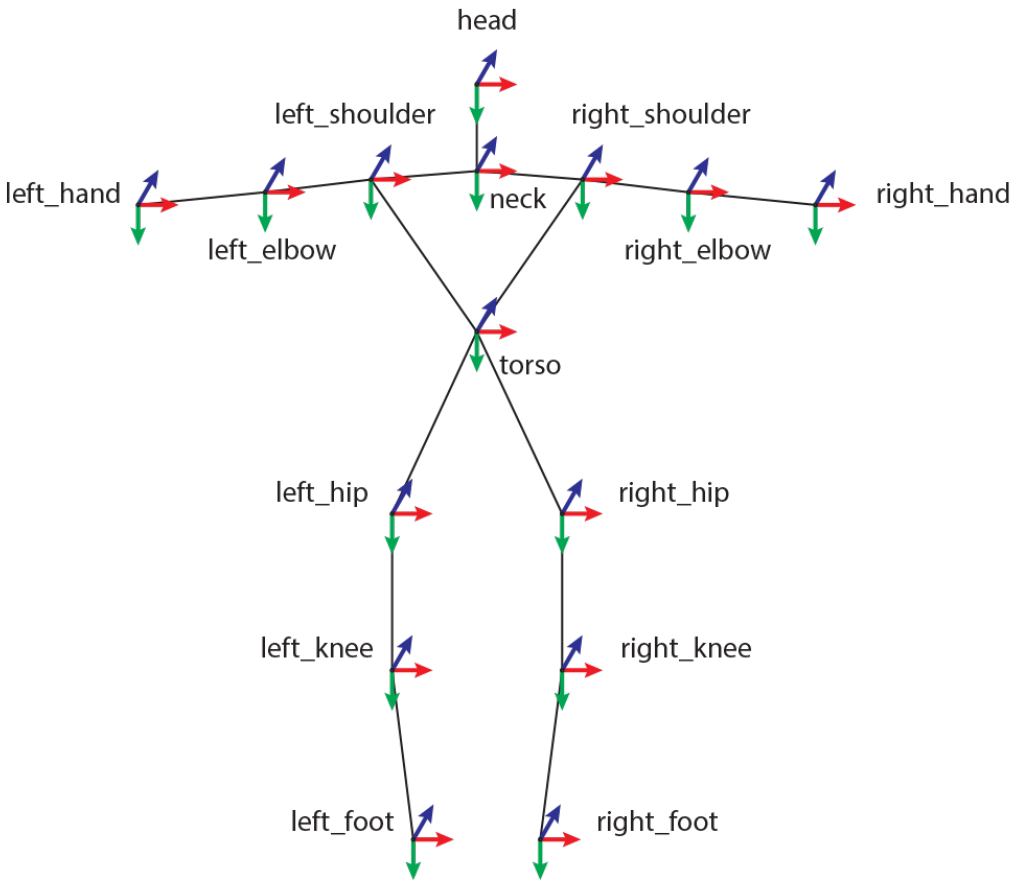


Figure 3.1: Skeleton joints provided by the tracker.

ence system on the neck, so if we consider two joints a and b the vectors from the neck to the joints will be respectively v_a and v_b, while the vector between the two joints is denoted by v_{a-b}.

The remapping algorithm can be split into two phases: upper limbs and lower limbs. For each part only the equation to map the right side are explained, the left side equations can be easily calculated with symmetric operations.

We start with the upper limbs. It is easy to see that the arm position is connected to the three vectors $v_{shoulder}$, v_{elbow}, and v_{hand}. In particular, for the Robovie-X, the three mentioned vectors have to lie on the same plane Π to obtain a valid arm position. The plane Π can be defined by means of the two vectors $v_{shoulder-elbow}$ and $v_{shoulder-hand}$. It is important to choose the proper correspondences: movements in the vectors have to correspond to similar movements in the plane to avoid that small shifts lead to high plane changes.

Shoulder Pitch

The right shoulder pitch angle α can vary between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. The value of the angle depends directly from the elbow position as described in Equation 3.1.

$$\alpha = \begin{cases} \arctan(z_{shoulder-elbow}, -y_{shoulder-elbow}) & if \quad y_{shoulder-elbow} \leq 0; \\ \arctan(-z_{shoulder-elbow}, y_{shoulder-elbow}) & if \quad y_{shoulder-elbow} > 0. \end{cases} \tag{3.1}$$

where

- $y_{shoulder-elbow}$ is the y component of the vector $v_{shoulder-elbow}$;
- $z_{shoulder-elbow}$ is the z component of the vector $v_{shoulder-elbow}$.

Shoulder Roll

The right shoulder roll angle β also depends on the elbow position and it ranges from $-\pi$ to 0. The formula is reported in Equation 3.2.

$$\beta = \begin{cases} \arcsin\left(\frac{x_{shoulder-elbow}}{\|v_{shoulder-elbow}\|}\right) & if \quad y_{shoulder-elbow} \leq 0; \\ -\pi + \arcsin\left(\frac{x_{shoulder-elbow}}{\|v_{shoulder-elbow}\|}\right) & if \quad y_{shoulder-elbow} > 0. \end{cases} \tag{3.2}$$

where $x_{shoulder-elbow}$ is the x component of the vector $v_{shoulder-elbow}$.

Elbow Roll

The last joint to consider is the right elbow roll γ, which ranges between $-\frac{5}{4}\pi$ and $\frac{\pi}{4}$. It depends from the two angles previously calculated and from the vector $v_{elbow-hand}$. The resulting formula is reported in Equation 3.3.

$$\gamma = \arctan\left(v_{elbow-hand}\hat{v}_{\alpha,\beta}, v_{elbow-hand}\hat{v}_{shoulder-elbow}\right) \tag{3.3}$$

where

- $\hat{v}_{\alpha,\beta}$ is the versor $e_1 = (1,0,0)$ rotated of a roll angle $-\alpha$ and of a yaw angle $-\beta$;
- $\hat{v}_{shoulder-elbow} = \frac{v_{shoulder-elbow}}{\|v_{shoulder-elbow}\|}$ is the versor in the $v_{shoulder-elbow}$ direction.

We also have to avoid singularities in which the plane equation has multiple solution and the robot configuration suddenly changes from one to another due to the noisy sensor. An hysteresis threshold has been used to maintain the system stable even with relatively uncertain data.

The parameter λ is introduced to set the hysteresis size and it depends on the distance between the hand and the shoulder, so that the it continuously changes in time following Equation 3.4.

$$v_{shoulder-hand} = \lambda v_{shoulder-elbow} + (1-\lambda)v_{elbow-hand} \tag{3.4}$$

Figure 3.2 shows the trend of the λ parameter depending on the hand distance from the shoulder.

The lower limbs motion is quite similar to upper limbs, if we consider the three vectors v_{hip}, v_{knee}, and v_{ankle}. However, there is a difference: the reference system is not integral with the joints involved in the motion, that is the case of the upper limbs. We have to take account of the rotation between $v_{shoulder}$ and v_{torso}

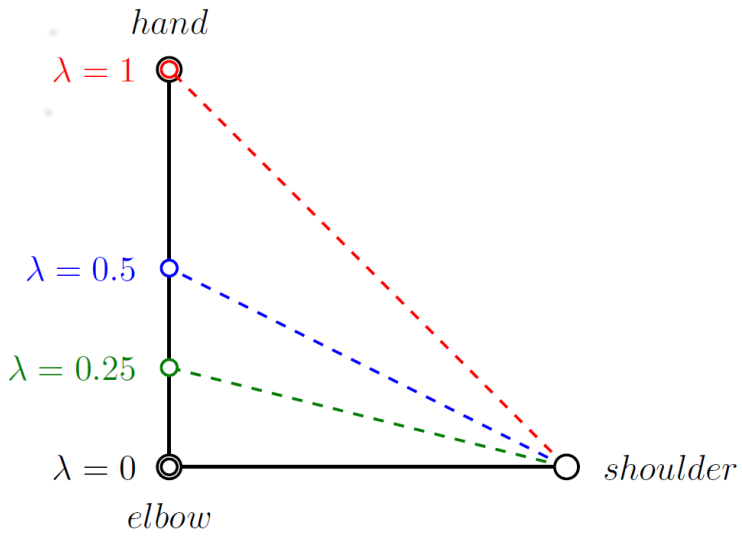


Figure 3.2: The trend of the λ parameter depending on the hand distance from the shoulder.

(Equation 3.5).

$$R_{shoulder-torso} = \begin{bmatrix} \hat{v}_{shoulder} \\ \hat{v}_{torso} \\ \hat{v}_{shoulder \times torso} \end{bmatrix}^{-1} \tag{3.5}$$

where

- $\hat{v}_{shoulder} = \frac{v_{shoulder}}{\|v_{shoulder}\|}$ is the versor in the $v_{shoulder}$ direction;
- $\hat{v}_{torso} = \frac{v_{torso}}{\|v_{torso}\|}$ is the versor in the $v_{shoulder}$ direction;
- $\hat{v}_{shoulder \times torso} = \frac{v_{shoulder \times torso}}{\|v_{shoulder \times torso}\|}$ is the versor in the $v_{shoulder \times torso}$ direction.

The resulting rotation matrix $R_{shoulder-torso}$ is used to compute the vectors:

- $w_{hip} = R_{shoulder-torso} \times v_{hip}$;
- $w_{knee} = R_{shoulder-torso} \times v_{knee}$;
- $w_{ankle} = R_{shoulder-torso} \times v_{ankle}$;
- $w_{hip-knee} = w_{hip} - w_{knee}$;

- $w_{knee-ankle} = w_{knee} - w_{ankle}$.

The remapping for the lower limbs can now be calculated in a proper way.

Hip Pitch

The right hip pitch angle φ can vary between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. The value of the angle depends directly from the knee position as described in Equation 3.6.

$$\varphi = -\arctan\left(-z_{hip-knee}, y_{hip-knee}\right) \tag{3.6}$$

where

- $y_{hip-knee}$ is the y component of the vector $w_{hip-knee}$;
- $z_{hip-knee}$ is the z component of the vector $w_{hip-knee}$.

Hip Roll

The right hip roll angle ψ also depend from the elbow position and it varies from $-\frac{3}{4}\pi$ to 0. The formula is reported in Equation 3.7.

$$\psi = \arcsin\left(\frac{x_{hip-knee}}{\|w_{hip-knee}\|}\right) \tag{3.7}$$

where $x_{hip-knee}$ is the x component of the vector $w_{hip-knee}$.

Knee Pitch

The last joint to consider it the right knee roll ω, its range is between $-\frac{5}{4}\pi$ and $\frac{\pi}{4}$. It depends from the two angles previously calculated and from the vector $v_{knee-ankle}$. The resulting formula is reported in Equation 3.8.

$$\omega = \arctan\left(w_{knee-ankle}\hat{w}_{\varphi,\psi}, w_{knee-ankle}\hat{w}_{hip-knee}\right) \tag{3.8}$$

where

- $\hat{w}_{\varphi,\psi}$ is the versor $e_2 = (0,1,0)$ rotated of a yaw angle $-\psi$ and of a roll angle φ;

- $\hat{w}_{hip-knee} = \frac{w_{hip-knee}}{\|w_{hip-knee}\|}$ is the versor in the $w_{hip-knee}$ direction.

By using the equation described the Robovie-X is able to copy human movements. However, the resulting motion still has some limitations due to the different degrees of freedom between human and robot.

3.2.2 Upper body motion and refinement

The robot considered in this subsection is an Aldebaran NAO H25 v 4.0. NAO is one of the most popular humanoid robots in the market, it is used for both research and service purposes and for these reasons we chose it to test our algorithms.

NAO also has more degrees of freedom with respect to Robovie-X, this aspect is particularly critical in the upper body movements of the new robot. Thus, a novel approach has been used in order to take account of the additional joints and to solve the problems highlighted from the previous technique.

Starting from the skeleton data, we extract the information relative to the right arm movement (Figure 3.4), but as before a similar approach can be used to the left arm as well. The information is used to properly control the robot arm (Figure 3.3)[1].

Shoulder Roll

The right shoulder roll angle α is computed using the normalized scalar product of vector $u(neck, r\_soulder)$, starting from the neck skeleton joint and ending to the right shoulder skeleton joint, and the vector $v(r\_soulder, r\_elbow)$, starting from the right shoulder skeleton joint and ending to the right elbow skeleton joint. In Equation 3.9 the complete analytic formula is reported.

$$\alpha = \frac{\pi}{2} - \arccos\left(\frac{u \cdot v}{\|u\|\|v\|}\right) \tag{3.9}$$

[1] More information can be found at *http://www.aldebaran-robotics.com/documentation/family/nao_h21/joints_h21.html*

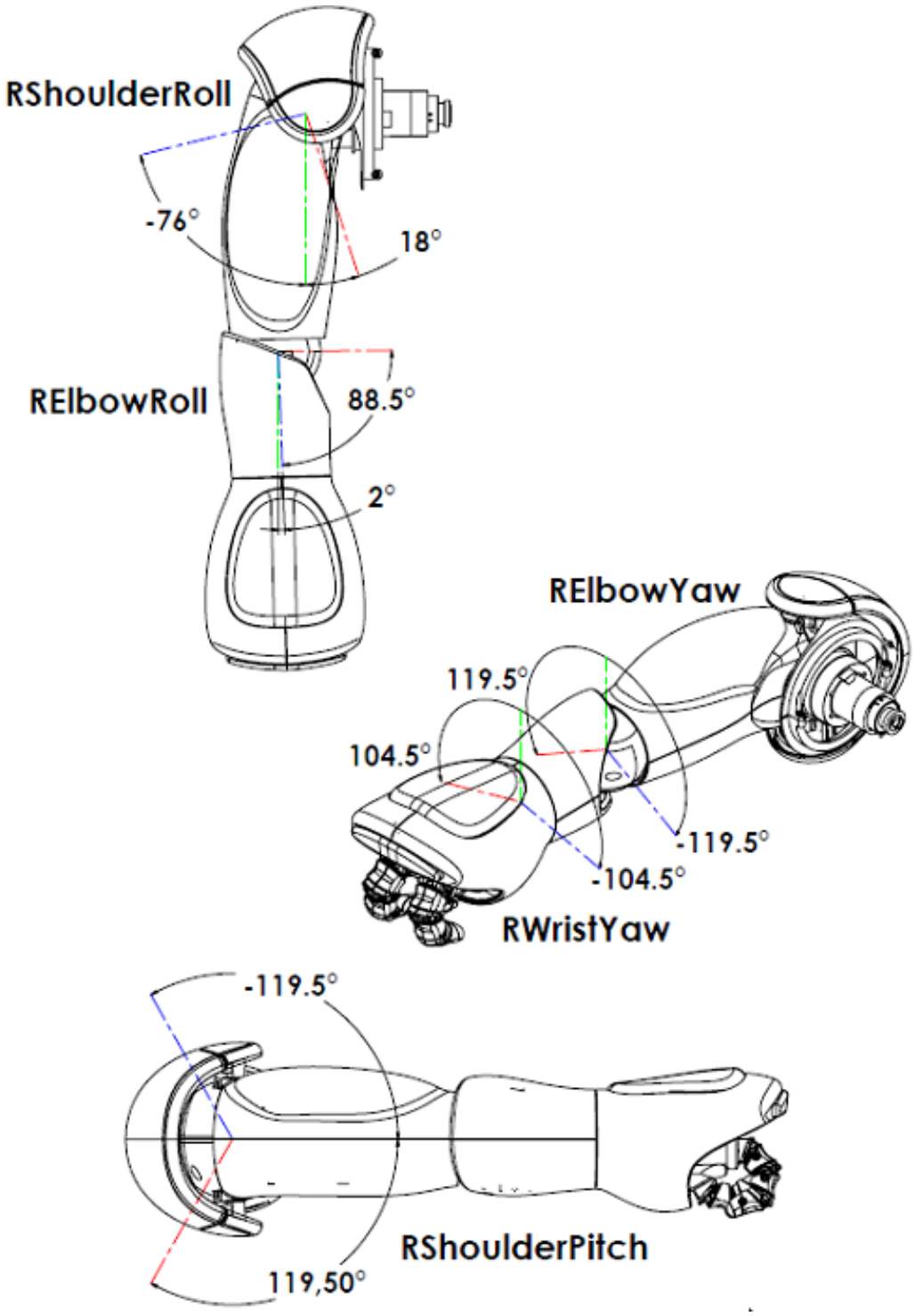


Figure 3.3: Aldebaran NAO, right arm working area.

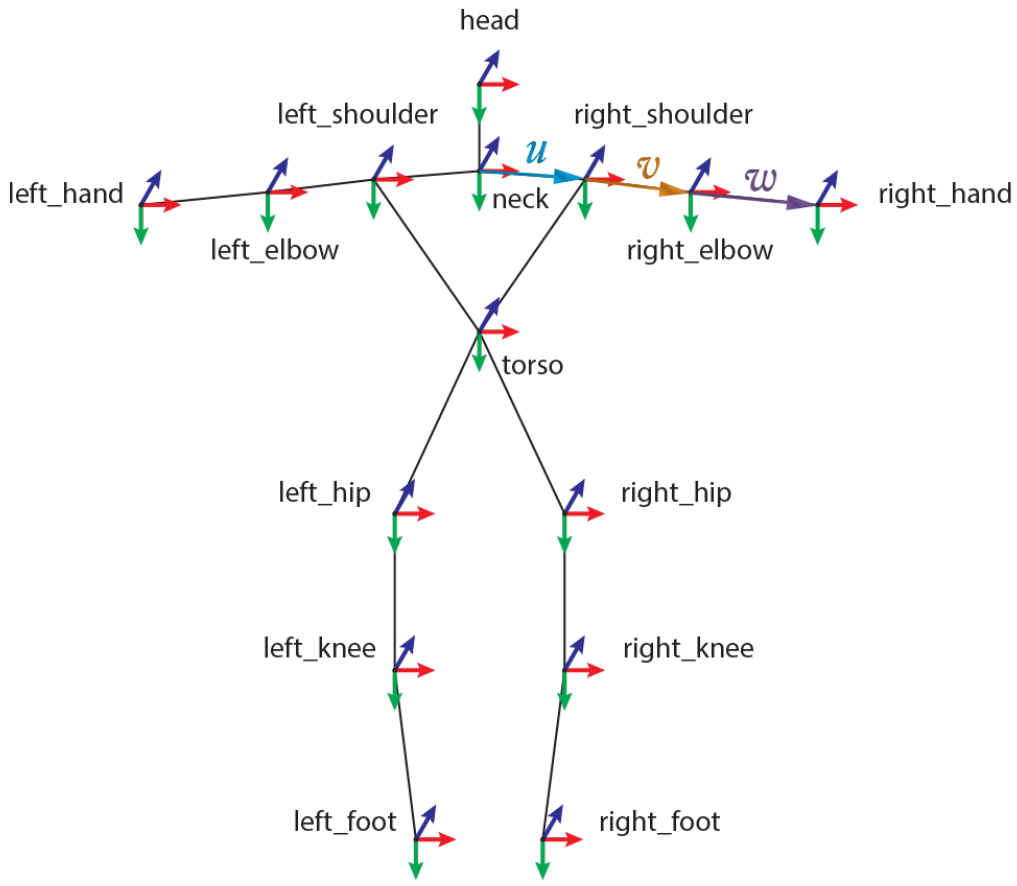


Figure 3.4: Vectors calculated starting from skeleton joints.

Shoulder Pitch

The right shoulder pitch angle β is calculated projecting the vector $v(r\_soulder, r\_elbow)$, starting from the right shoulder skeleton joint and ending to the right elbow skeleton joint, onto the XY plane, and then using the formula in Equation 3.10.

$$\beta = \begin{cases} \arccos\left(\frac{y_v}{\mathscr{P}_{XY}(v)}\right) & if \quad z_v < 0; \\ \pi - \arccos\left(\frac{y_v}{\mathscr{P}_{XY}(v)}\right) & if \quad z_v \geq 0, y_v \geq 0; \\ -\pi - \arccos\left(\frac{y_v}{\mathscr{P}_{XY}(v)}\right) & if \quad z_v \geq 0, y_v < 0. \end{cases} \tag{3.10}$$

where y_v and z_v are respectively the y and z components of the considered vector v.

Elbow Roll

In order to compute the right elbow roll angle γ, the vector $-v(r\_soulder, r\_elbow)$, from the right elbow skeleton joint to the right shoulder skeleton joint, and the vector $w(r\_elbow, r\_hand)$, from the right elbow skeleton joint to the right hand skeleton joint, are treated in a similar way than in Equation 3.9. The resulting formula is reported in Equation 3.11

$$\gamma = \pi - \arccos\left(\frac{-v \cdot w}{\|v\|\|w\|}\right) \tag{3.11}$$

Elbow Yaw

The right elbow yaw angle δ is calculated considering the y component of the vector $w(r\_elbow, r\_hand)$, starting from the right elbow skeleton joint and ending to the right hand skeleton joint. In Equation 3.12 the complete analytic formula is reported.

$$\delta = -\arcsin(y_w) \tag{3.12}$$

No data are available from the skeleton tracker to properly calculate the wrist rotations.

Human joints are more "advanced" than the robotic ones: the angle values can overcome the limits of the robot and the movements vary a lot from one person

to another. Some thresholds were introduced to clearly identify the human movement to remap. All the data before a starting angle and after an ending angle are automatically discarded as outliers. Finally the angle distance between the maximum and minimum angle of each motion are properly rescaled from the algorithm to fulfill the robot joint limits.

3.2.3 Lower body motion: stability control

A further step should be developed in order to properly re-targeting the lower part of a humanoid: the robot has to automatically avoid unstable situations. In this subsection is described a simple stability algorithms to make a robot picking up an object teleoperated by an human actor.

Base strategy

As we said before, our primary goal is to make a robot picking up an object laying in front of it by imitating the human movements coming from a skeleton tracking system. The main challenge is to keep the robot stable while it is crouching and grasping the object.

A consolidated method to maintain the robot stability [90] is to keep the Center of Mass (CoM) projection point inside the contact area of the feet with the ground. The CoM projection point of the robot should be calculated in order to reach two different purposes: maintain the point inside the safe balanced area and keep the robot movements as similar as possible to the user motions.

At each instant, the CoM is equal to:

$$CoM_x = \frac{\sum_{k=1}^{N} m_k x_k}{M}; \quad CoM_y = \frac{\sum_{k=1}^{N} m_k y_k}{M}; \quad CoM_z = \frac{\sum_{k=1}^{N} m_k z_k}{M} \tag{3.13}$$

where $N = 17$ is the number of joints, m_k is the k^{th} joint inertial mass, M is the mass of the robot, and x_k, y_k, z_k are the coordinates of the k^{th} joint with respect to the Torso joint.

Thus, the ground projection of the CoM is given by (CoM_x, CoM_y), and it has to satisfy the constraints $-90cm < Com_x < 90cm$ and $-35.5cm < Com_y < 35.5cm$

that is the area covered by the robot foot in the initial standing position. The selected movement is quite simple, so several solutions are feasible to solve the problem. We imposed a strong relation between hip, knee an ankle joints in order to involve all the lower body joints and at the same time adapt to the human natural behavior.

$$\gamma = \alpha = \frac{\beta}{2}, \qquad 0 \leq \beta \leq 157^o \tag{3.14}$$

where

- α is the pitch hip angle at instant t;
- β is the knee angle at instant t;
- γ is the pitch ankle angle at instant t.

Figure 3.5 shows the three described angles in the robot model. The method was tested experimentally on both simulated and real robot.

Refinement

We refine the described technique applying different strategies in order to avoid some rough robot movements we noticed during the initial tests. Studying the dynamics of the task, we decided to limit roll movements (i.e. lateral movements) in joints not involved in reaching the goal, like hips and ankles.

Fast and sudden movements could threaten the robot stability while performing an activity. Ensuring the smoothness of all robot moves is essential to correctly balance the robot. Without any focused control, the input data can make the robot move jerkily. This problem is due to the fast human movements compared with the frame rate of the sensor and to the margin of error of the skeleton tracker computing joints positions.

The data acquired by the RGB-D sensor are filtered to remove the noise by calculating the mean value of the last three angle values of every joint in order to avoid rough robot movements and obtain a smoother motion.

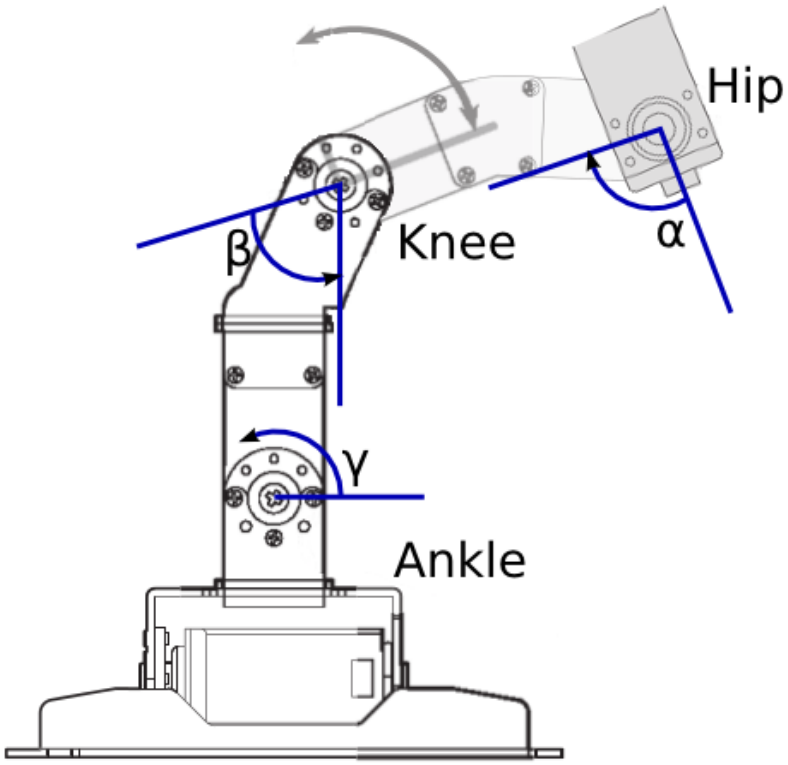


Figure 3.5: Main joint angles involved in balance.

The pose feedback equation is:

$$\hat{\xi}_t = \frac{\hat{\xi}_{t-2} + \hat{\xi}_{t-1} + \xi_t}{3} \tag{3.15}$$

where

- ξ_t is the raw value given by the skeleton tracker for a certain joint;
- $\hat{\xi}_t$ is the computed value at the instant t for the considered joint;
- $\hat{\xi}_{t-1}$ is the computed value at the instant $t-1$ for the considered joint;
- $\hat{\xi}_{t-2}$ is the computed value at the instant $t-2$ for the considered joint.

Moreover, the movements of the right and left side of the robot body are coordinated in order to make easier for the robot to grasp the object. In this way, we also increase the robot stability and its precision during the motion.

Finally, the proportions between lower body angles are computed according to the Equation 3.14. These refined data are used as input to the algorithm checking if the CoM projection on the ground is inside the stability area.

Again, the whole system has been tested with many users and different objects[2] on both real and simulated environment. The applied refinements significantly improved the performance and the users easily reached the goal. It is worth to notice that a slight delay is introduced by USB connection between the system and the real robot, nevertheless no delay is present in the simulated model, that works at 30 fps. We also can make real and simulated robot work together, so humans can take advantage of the information provided by the virtual model.

3.3 Human to manipulator re-targeting

3.3.1 Motion evaluation

In this section, the motion re-targeting system has to consider two different articulated models: the human body and the manipulator. The robot used is the Comau Smart5 SiX (Section B.2). The robot measures and operating area are showed in Figure 3.6. A novel virtual model of this robot has been developed in order to avoid damages on the real robot, so the re-targeting was tested on the virtual model. This model is suitable for different Open Source 3D simulators. In particular, we tested it in Gazebo [45] and V-REP [28].

Recent works in computer graphics highlighted that re-targeting from the whole human body to a non-humanoid avatar are feasible also for complex models [95] [83]. In this work, the motion captured from the 15 skeleton joints are remapped to 3 robot joints related to the "rough" robot motion: *Axis1*, *Axis2*, *Axis3*. We did not take into account the 3 remaining axes (*Axis4*, *Axis5*, *Axis6*), which control the "fine-grained" end effector motion. A vision based hand gesture interface [33] could be used to get a more effective control of the end effector, especially if combined with recent manipulation techniques from uncertain depth data [99].

[2]Few videos of some tests realized: `https://www.youtube.com/watch?v=LJyXT6gAyo8` `https://www.youtube.com/watch?v=AOIkVLn3Kng` `https://www.youtube.com/watch?v=GS9A4prXfpI`

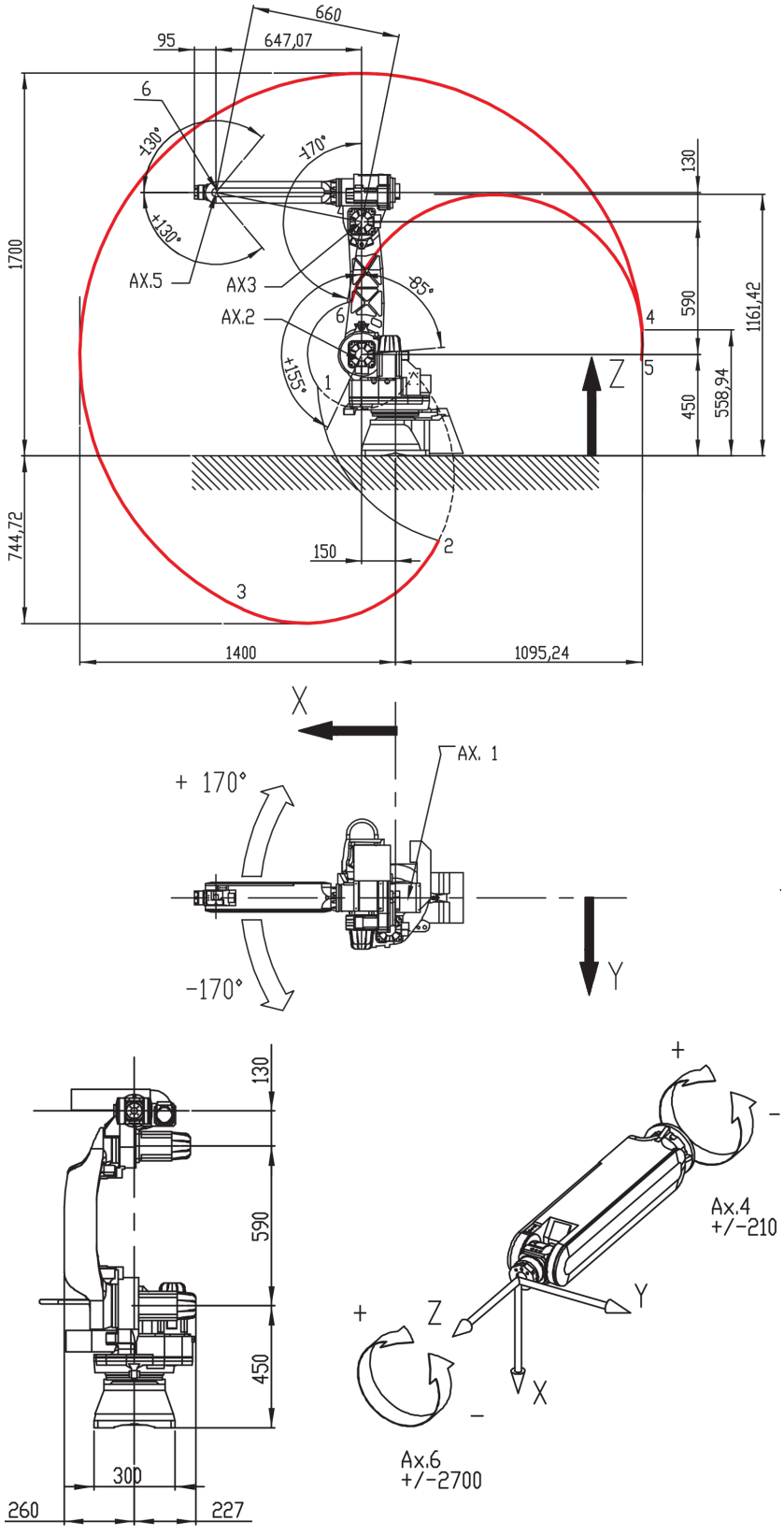


Figure 3.6: Comau Smart5 SiX operating area (red line). The overall dimensions are also reported.

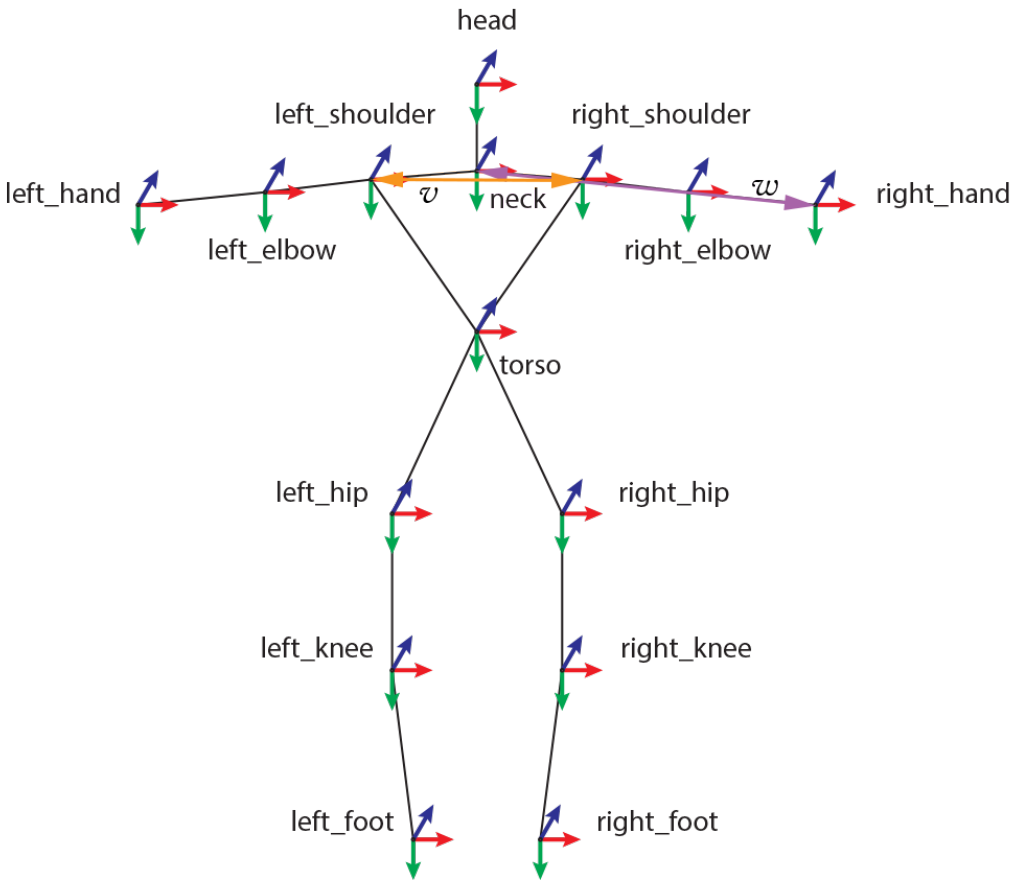


Figure 3.7: Vectors calculated starting from skeleton joints.

The *Axis1* rotation angle α is the projection $\mathscr{P}$ onto the XY plane of the robot coordinate system of the vector $v(neck, r\_hand)$ starting from the neck skeleton joint and ending in the right hand skeleton joint (Figure 3.7). In Equation 3.16 the analytic formula is reported.

$$\alpha = \begin{cases} \arccos\left(\frac{x_v}{\mathscr{P}_{XY}(v)}\right) & if \quad y_v \geq 0, \\ -\arccos\left(\frac{x_v}{\mathscr{P}_{XY}(v)}\right) & if \quad y_v < 0 \end{cases} \tag{3.16}$$

where x_v and y_v are the x and y components of the considered vector v.

The rotation β around the *Axis2* is referred to the relative motion of the demonstrator with respect to his initial position. In order to be robust to the sensor noise we considered the position of the segment $w(l\_shoulder, r\_shoulder)$ from left shoulder skeleton joint to right shoulder skeleton joint (Figure 3.7), in such way also the orientation of the vector w can easily be included in the formula (Equation 3.17).

$$\beta = \frac{\pi}{2} - \arccos\frac{\mathscr{P}_X\left(M_R^{-1} \cdot d_w\right)}{l_{Axis2}}; \tag{3.17}$$

$$M_R = \begin{bmatrix} y_w/\|w\| & x_w/\|w\| & 0 \\ -x_w/\|w\| & y_w/\|w\| & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

where

- M_R is the rotation matrix between initial and current w segment (Equation 3.18);
- d_w is the distance between initial and current position of the w mean point $p_m \in w$;
- $\mathscr{P}_X$ denotes the projection onto X;
- l_{Axis2} is the length of the robot part involved in the motion (namely *Axis2* - *Axis1*).

The *Axis3* rotation γ really represents the human arm and it depends from both v and w. Equation 3.19 explains how to calculate the desired angle γ starting from skeleton joints.

$$\gamma = \begin{cases} \beta - \frac{\pi}{2} - \arccos \frac{\sqrt{x_v^2 + y_v^2}}{\|v\|} & if \quad z_v \geq 0, \\ \beta - \frac{\pi}{2} + \arccos \frac{\sqrt{x_v^2 + y_v^2}}{\|v\|} & if \quad z_v < 0 \end{cases} \tag{3.19}$$

where the notation is the same used in the previous equations.

Different tests on several people have been performed to guarantee that the interaction is natural. In a first session, the users were asked to freely move, while they were looking at the virtual robot environment. The aim was to understand if the robot motion was the same expected from the users. A second test was designed to compare an actor and a robot execution. The users involved in this session had to guess the correspondence between the two performances. Thanks to the described tests we were able to apply the proper corrections and modifications, so that the algorithm has assumed the formulation presented in this work.

3.4 Summary

In this chapter, information extracted from human natural motion were reprojected to the robot joints space using both humanoids and manipulators. Tests highlighted that naive users are able to control the whole humanoid robot, with particular attention to the upper body motion and showing good potentiality in tasks involving robot stability. Moreover, a novel motion re-targeting technique was proposed to control a manipulator by using the whole body. Some other tests on the described methods will be presented in Chapter 4, where the data provided using these algorithms will be used as input of different Robot Learning from Demonstration frameworks.

As future work we will test both the motion re-targeting systems in more challenging scenarios to better exploit their potentials.

Chapter 4

Visual Robot Learning by Demonstration

As we already said, the aim of the Robot Learning from Demonstration (RLfD) is to easily teach new skills to a robot. Instead of using a so called “programming language”, this paradigm makes the robot learn by means of demonstrations.

Collecting demonstrations from human beings can be particularly effective in case of anthropomorphic robots, like humanoids or manipulators. A key point of the RLfD process is to extract useful information from the great amount of data gathered through human demonstrations. In this regard, it is very important to find out the constraints that control the motion and modeling them by using a proper model.

The model we used in developing this work is based on a probabilistic approach. Such a kind of approach allows us to catch the variability in the performances coming from several subjects acting in different ways to obtain the same goal. In fact, when we describe a task, we usually focus our attention on the result we would like to obtain, omitting the details on how to do it. For example, if we can ask someone ”Could you please bring me that pen?” we will not say anything about the movement this person should do, or about the grasping point on the pen.

Even in the simplest task, it is easy to find a great variety of different solutions. In this chapter we briefly describe some related works (Section 4.1). Section 4.2 and Section 4.3 introduce respectively Gaussian Mixture Model (GMM)

and Gaussian Mixture Regression (GMR). Section 4.4 describes how to selected the number of Gaussian components to initialize the GMM, while Section 4.5 explains how GMM/GMR can be applied to compose a framework suitable to model natural demonstrations for a manipulator in an industrial environment. In Section 4.6 an alternative way to encode information from visual demonstrations is reported: the Donut Mixture Model (DMM). Section 4.7 explains how to generate an output trajectory by maximizing the DMM probability density function. Subsequently, the DMM framework is tested on a real task, namely score a basket, by means of kinesthetic demonstrations (Section 4.8) and by using RGB-D videos of human trials (Section 4.9).

4.1 Related Works

The idea of learning a task by using examples appeared in the early 90's in order to define a computer program without any knowledge of programming languages [18] [86]. The idea was to observe the commands inputed by an end-user and reiterate the sequence in similar situations considering that the graphical user interface strategies are almost similar in all the developed applications.

When the paradigm was introduces in robotics [22], the problem assumed physical characteristics: operators do not have the programming skills to code the robot motion into tasks they implicitly know how to achieve. First attempts focus on teleoperating the robot to perform the task and then apply the same strategy used in the computer programming field [48].

The introduction of machine learning techniques leads the research to take account of real world variabilities by generalizing robot behaviors, object positions and orientations, structure. Several levels of abstraction and segmentation have been used on both trajectory level [89] and symbolic level [65]. The trajectory level includes continuous signals like joint positions or orientations, motor current or torques changing over time, while the symbolic level extracts discrete features using a set of defined rules in order to hierarchically organize them.

Several researchers in robotics proposed a way to generalize a skill, in fact the robot can not correctly reproduce a task in a unknown situation without this capability. Nicolescu et al. [72] represented a skill by using a graphical model:

the nodes identifies the robot behaviors, while the arcs connecting the graph are the common subsequences used to generalize the task. More recently Koppula et al. [46] proposed graphical model to represent human behaviors, understand their intentions and make the robot choose the right way to interact. Schaal et al. [81] used motion primitives to encode learning data. Similarly, Alissandrakis et al. [5] subdivided a motion into pre-defined postures, positions and configurations depending on the granularity at the symbolic level. Akgun et al. [3] extracted keyframes to correctly model a skill. Pardowitz et al [75] organized skills into a incremental sequence of objects to arrange in order to set up a table: each object, such as plates or cups, represents a hierarchical level to be set before going to the next one.

Different techniques based on statistics have been proposed to extract only the essential constraints characterizing a task across multiple demonstrations. In several works Calinon and Billard proposed a generic methodology to represent in a probabilistic manner continuous data, like human gestures or robot motions. In [14], they encoded the trajectories draw by an actor performing a gestures by using a Hidden Markov Model (HMM): a sequence of states is extracted from the observed trajectories to compose the HMM. To overcome reproduction problems due to Hidden Markov Model, in [15] they used a Gaussian Mixture Model (GMM) to encode multiple trajectories, while a smooth trajectory can be extracted by means of a Gaussian Mixture Regression (GMR). In [16], they proposed an Hidden Markov Model/Gaussian Mixture Regression technique to maintain the characteristic of HMM and otain a easy way reproduce human demonstrations in a multiple constraints environment. Finally, they also explored the use of binary signals encoding them in a Bernoulli Mixture Model (BMM) [17]. The use of a probabilistic framework allows them to autonomously and incrementally learn by classifying them by an efficient and compact representation. The information modeled can be then used to evaluate reproduction attempts performed by using robot self-experimentation by extracting the fundamental constraints that characterize different tasks. The same constraints can be used to reproduce the skill in an unknown situation.

All the cited works are based on examples that are usable by a robot. There are two main conditions imposed to achieve this kind of examples: demonstrators

are constrain to modalities understandable by the robot and the number of demonstrations has to be large enough to guarantee the desired generality. In this thesis, we relaxed the first constraint in order to extract information directly from human natural demonstrations. An initial work based on a GMM/GMR framework are tested on high repetitively industrial tasks, but to obtain similar results in very dynamic environment like service robotics we adopted a more flexible model able to infer key constrains even from failed demonstration [35] [34]: the Donut Mixture Model.

4.2 Gaussian Mixture Model

Gaussian Mixture Model is a probability manner to encode a set of N data, in this work temporal components of continuous trajectories coming from natural human demonstrations. In particular, the total number of data will be $N = nT$, where n is the number of collected demonstrations, and T is the number of observations acquired from the sensor at a constant frame rate.

The aim is to obtain the best sum of K Gaussian components which approximate the input dataset $\xi = \left\{\xi_j\right\}_{j=1,\ldots,N}$, $\xi_j \in \mathbb{R}^D$, with D the dimensionality of the problem. Each demonstration consists on a sequence of robot joint angles $\xi_s \in \mathbb{R}^{D-1}$ recorded in a specific temporal instant $\xi_t \in \mathbb{R}$. The resulting probability density function is described in Equation 4.1.

$$p\left(\xi_j\right) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\xi_j; \mu_k, \Sigma_k\right) \tag{4.1}$$

where

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

The Expectation-Maximization (EM) [21] algorithm is iteratively used to estimate the optimal mixture parameters $\theta = (\pi_k, \mu_k, \Sigma_k)$. The algorithm can be separated in two cyclic phases: expectation and maximization. The iteration stops

when the increase of the log-likelihood $\mathscr{L} = \sum_{j=1}^{N} \log\left(p\left(\xi_j|\theta\right)\right)$ at each iteration becomes smaller than a defined threshold ε, given by $\frac{\mathscr{L}(t+1)}{\mathscr{L}(t)} < \varepsilon$.

The expectation step (E-step) is described in Equation 4.2.

$$p_{k,j}(t+1) = \frac{\pi_k(t)\mathscr{N}(\xi_j;\mu_k(t),\Sigma_k(t))}{\sum_{i=1}^{K}\pi_i(t)\mathscr{N}(\xi_j;\mu_i(t),\Sigma_i(t))} \tag{4.2}$$

While the maximization step (M-step) is reported in Equation 4.3

$$\pi_k(t+1) = \frac{1}{N}\sum_{i=1}^{N} p_{k,j}(t+1) \tag{4.3}$$

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

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

The algorithm optimize the parameters of the K Gaussian components by maintaining a monotone increasing likelihood during the local search of the maximum.

4.3 Gaussian Mixture Regression

The aim of Gaussian Mixture Regression is to retrieve a smooth generalized version of the signal encoded in the associated Gaussian Mixture Model. In this work, we calculated the conditional expectation of robot joint angles $\hat{\xi}_s$, starting from the consecutive temporal values ξ_t known a priori. As we already said, a Gaussian component k is defined by the parameters (π_k, μ_k, Σ_k), with:

$$\mu_k = \left\{\mu_{t,k}\,\mu_{s,k}\right\} \qquad \Sigma_k = \begin{bmatrix} \Sigma_{t,k} & \Sigma_{ts,k} \\ \Sigma_{st,k} & \Sigma_{s,k} \end{bmatrix} \tag{4.4}$$

The conditional expectation and its covariance can be estimated respectively using Equation 4.5 and 4.6.

$$\hat{\xi}_s = E\left[\xi_s \,|\xi_t\right] = \sum_{k=1}^{K} \beta_k \hat{\xi}_{s,k} \tag{4.5}$$

$$\hat{\Sigma}_s = Cov\left[\xi_s \,|\xi_t\right] = \sum_{k=1}^{K} \beta_k^2 \hat{\Sigma}_{s,k} \tag{4.6}$$

where

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

Thus, the generalized form of the motions $\hat{\xi} = \left\{\xi_t, \hat{\xi}_s\right\}$ is generated by keeping in memory only the means and covariance matrices of the Gaussian components calculated through the GMM.

4.4 Number of Mixture

In order to calculate the appropriate number of Gaussian components K to initialize the GMM, two criteria have been considered: the Akaike Information Criterion (AIC) [2] and the Bayesian Information Criterion (BIC) [82].

Both AIC and BIC are largely used to estimate the optimal number K of Gaussian components in the Mixture. The score assigned at each considered k for the Akaike Information Criterion is described in Equation 4.7, while Equation 4.8 represents the Bayesian Information Criterion.

$$S_{AIC} = -2\mathscr{L} + 2n_p \tag{4.7}$$

$$S_{BIC} = -2\mathscr{L} + n_p \log N \tag{4.8}$$

where for both equations

- $\mathscr{L} = \sum_{j=1}^{N} \log\left(p\left(\xi_j|\theta\right)\right)$ is the log-likelihood for the considered model θ;
- $n_p = (K-1) + K(D + {}^1\!/_2 D(D+1))$ 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 aims to avoid data overfitting to keep the model general.

4.5 Industrial environment

Some tests have been performed to validate the framework on real acquisition data and challenging scenarios. In the experiments, an industrial robot have been used, namely a simulated Comau Smart5 SiX (Section B.2). The motion of the whole actors body demonstrating the tasks are remapped on three joints on the manipulator as described in Section 3.3. The tasks tested consist of two different scenarios.

In the first experiment the robot learns how to reach an object at a specific position in a set of several similar items stacked one near each other. The scenario (Figure 4.1) is composed by three stacks of boxes placed in front of the demonstrator who pushes a specific box. Three demonstrators repeated the task reaching three different boxes, 15 times per box. The first selected box is the one on the top of the central stack, the second box is the one on the bottom of the right stack, and the third box is the one in the middle of the left stack. The robot has to reproduce the three subtasks reaching the right object each time.

In the second scenario the demonstrators are asked to move a box along a linear path from the beginning to the end of a small table (Figure 4.2); the covered distance is 45 cm. As in the previous experiment, three demonstrators repeated the task 15 times. The robot should be able to correctly reproduce the task showed by the actors in a similar simulated environment (Figure 4.3) . The virtual box is equivalent to the real one less than the color; instead, the table is larger in order to avoid singularities in the simulator when the box reach the end of the table.

The experiments were performed on the Comau Smart5 SiX simulated in Gazebo. No experiments are actually been performed on the real robot, but we

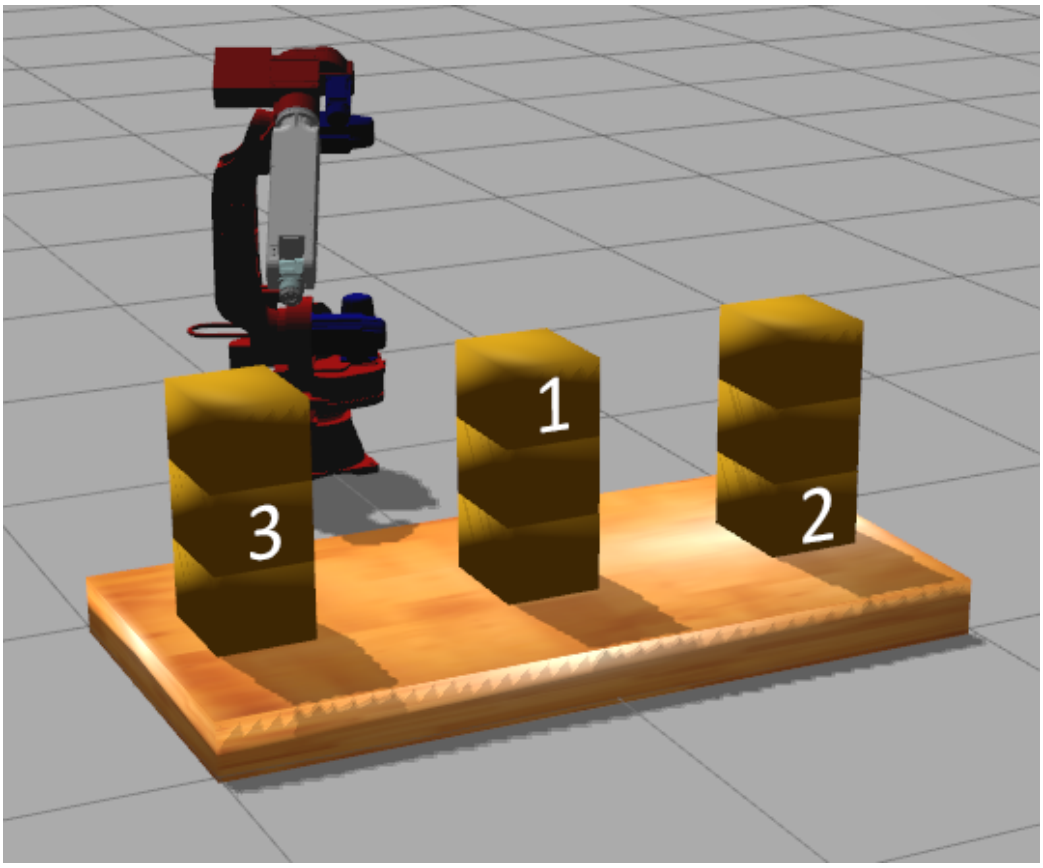


Figure 4.1: Overview of the first experimental scenario in the simulated world. The three subtasks performed by the demonstrators are numbered in the corresponding order.

Figure 4.2: An actor performing the movement of the box requested in the second task.

are working on testing our system also in the real world. The demonstrations were manually segmented and, subsequently, resampled using a spline interpolation technique to $T = 1750$ data-points. All the collected examples are properly re-targeted to the robot motion system, but only 107 out of 180 led to a qualitatively correct task once performed by the robot. In the first task the wrong box was pushed, while in the second one the box was not properly moved.

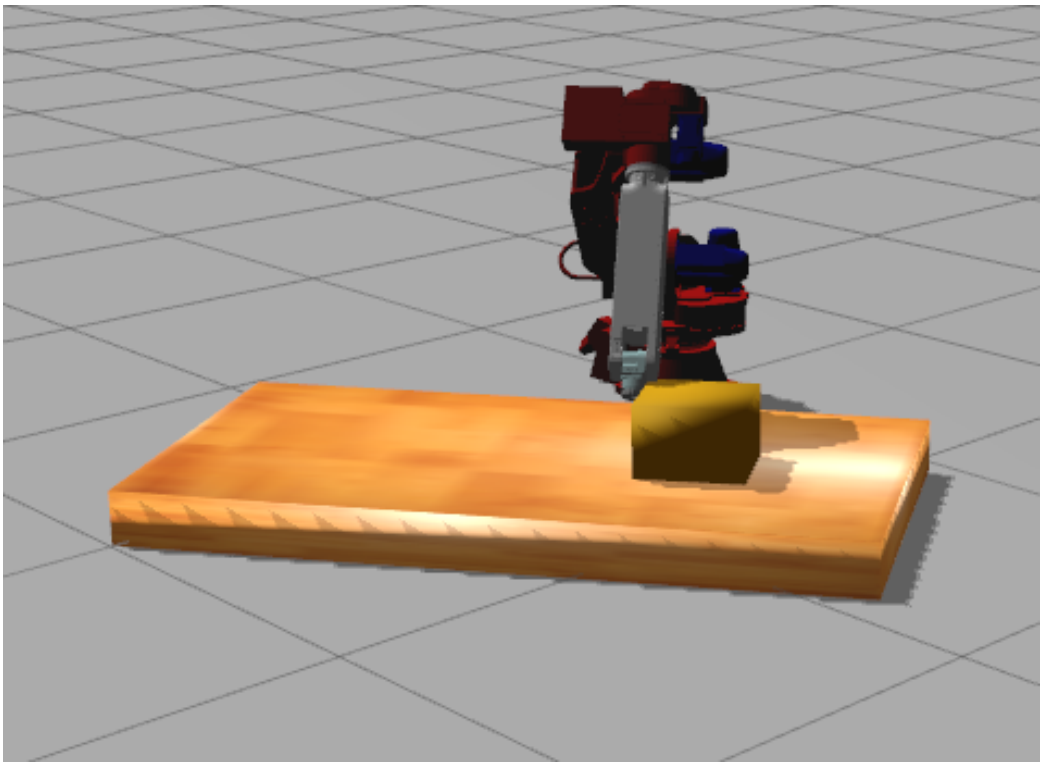


Figure 4.3: Overview of the second simulated scenario.

This experiment aims to test the precision and the repeatability of each task using the data collected through our motion re-targeting system. These features are very important to achieve the high performances required in an industrial environment.

We tested the repeatability reproducing the first task 20 times for each selected box, while the second task is simulated 25 times. Finally, the robot is able to correctly reproduce both a task 85 times. From the first experiment we do not have an accurate measure of the precision, thus we used the covered distance in the

second scenario to test this aspect. The data collected from the 25 attempts vary from 44.286 to 46.542 cm, giving a resulting mean of 45.011 ± 0.402 cm. This is a good simulation result, in fact the variability is due mainly to two factors: the initial conditions in the simulation environment and the sensor noise. Such precision can be feasible if we are working with a big gripper and large plates, but it is not so good as expected in a soldering task.

The results (Figure 4.4) showed that using our GMM/GMR framework the tasks are well learned with a low number of demonstrations (11 examples in the worst case). We test single actor and multi-actors learning: even if the tasks can be correctly performed in both methods, the single actor approach needs less examples that the multi-actors one. The path generated by the framework avoids rapid accelerations between timesteps and large oscillations in robot velocity. The resulting trend is very smooth and it is easy to notice that the sensor noise is removed, in fact the vertical straight-lines peaks present in the blue demonstrations are completely absent in the final red path.

In this work, we did not look at velocity, acceleration or force of the robot during the performance. This is a directly consequence of using positions to control the robot. We chose on purpose this modality to better focus on precision and repeatability, even omitting other important parameters.

4.6 Donut Mixture Model

In spite of the good results showed in an high repetitively industrial setting, the GMM/GMR framework lack of flexibility in more dynamic environment. In fact, the same framework does not work as well in service robotics, where the actors motion are not constricted from the constraint intrinsic in tasks. The Donut Mixture Model was tested to overcome the limitations showed by the GMM/GMR framework.

As the GMM, the Donut Mixture Model (DMM) is a probability manner to encode a set of data, in this work demonstrations. The input dataset $X = \{x_n\}_{n=1}^{N}$ is a set of N trajectories $x_n = \{\xi_{n,t}\dot{\xi}_{n,t}\}_{t=1}^{T_n}$, of possibly different dimensions T_n. Each trajectory is composed by joint positions $\xi = \{\xi_j\}_{j=1}^{D}$ and velocities $\dot{\xi} = f_\theta(\xi) = \{\dot{\xi}_j\}_{j=1}^{D}$, where D is the number of joints involved in the trajectory. The

DMM is used to approximate the non-linear function $\dot{\xi} = f_\theta(\xi)$.

A Donut component in the mixture model is defined as the difference between two Gaussian components:

$$\mathscr{D}(x|\mu_\alpha, \mu_\beta, \Sigma_\alpha, \Sigma_\beta, \gamma) = \gamma\mathscr{N}(x|\mu_\alpha, \Sigma_\alpha) - (\gamma - 1)\mathscr{N}(x|\mu_\beta, \Sigma_\beta) \tag{4.9}$$

where

- $\mathscr{D}$ is the Donut distribution;
- x is a trajectory;
- $\mathscr{N}$ is a standard normal distribution;
- μ_α and Σ_α are respectively the mean vector and the covariance matrix of the first Gaussian;
- μ_β and Σ_β are respectively the mean vector and the covariance matrix of the second Gaussian;
- $\gamma > 1$.

The aim of a Donut distribution is looking for a good trajectory in the covariance space between the two Gaussian distributions. In order to explore this space the distribution has to avoid the best approximation of the collected demonstrations obtained from a Gaussian distribution. Therefore the mean of the two Gaussians should be the same, $\mu_\alpha = \mu_\beta$. In order to keep constant the shape of the covariance, a parameterization is applied to the Donut covariance matrices, such that $\Sigma_\alpha = \Sigma/r_\alpha^2$ and $\Sigma_\beta = \Sigma/r_\beta^2$.

The Donut distribution changes as follow:

$$\mathscr{D}(x|\mu, \Sigma, r_\alpha, r_\beta, \gamma) = \gamma\mathscr{N}(x|\mu, \Sigma/r_\alpha^2) - (\gamma - 1)\mathscr{N}(x|\mu, \Sigma/r_\beta^2) \tag{4.10}$$

In order to calculate the Gaussian distributions, a Gaussian Mixture Model is computed from the input data. A weighted version of the Expectation-Maximization (EM) [70] process is used to iteratively adjust the (π_k, μ_k, Σ_k) parameters of the K Gaussians

composing the mixture, namely the priors vector, the mean vector and the covariance matrix. The Bayesian Information Criterion (BIC) [82] was used, as before, to estimate the optimal number of Gaussian components K in the Mixture. The overall state of the GMM is described by $\theta = \{K, \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}\}$.

The probability that a joint assumes a certain velocity given its position is

$$p(\dot{\xi}|\xi) = \sum_{k=1}^{K} \tilde{\pi}_k \mathscr{D}(\dot{\xi}|\tilde{\mu}_k, \tilde{\Sigma}_k, \varepsilon) \tag{4.11}$$

$$\tilde{\mu}_k = \tilde{\mu}_k(\xi) = \mu_k(\dot{\xi}) + \Sigma_k(\dot{\xi}, \xi)\Sigma_k^{-1}(\xi, \xi)(\xi - \mu_k(\xi))$$

$$\tilde{\Sigma}_k = \Sigma_k(\dot{\xi}, \dot{\xi}) + \Sigma_k(\dot{\xi}, \xi)\Sigma_k^{-1}(\xi, \xi)\Sigma_k(\xi, \dot{\xi})$$

$$\tilde{\pi}_k = \tilde{\pi}_k(\xi) = \frac{\pi_k \mathscr{N}(\xi; \mu_k(\xi), \Sigma_k(\xi, \xi))}{\sum_{k=1}^{K} \pi_k \mathscr{N}(\xi; \mu_k(\xi), \Sigma_k(\xi, \xi))}$$

$$\varepsilon = 1 - \frac{1}{1 + \|\tilde{V}[\dot{\xi}, \xi, \theta]\|}$$

$$\tilde{V}[\dot{\xi}|\xi, \theta] = -\tilde{E}[\dot{\xi}|\xi, \theta]\tilde{E}[\dot{\xi}|\xi, \theta]^{\top} + \sum_{k=1}^{K} \pi_k(\mu_k \mu_k^{\top} + \Sigma_k)$$

where

- γ is set to 2, so in Equation 4.10 the first Gaussian is doubled and than the second Gaussian is subtracted;
- $\tilde{\mu}_k$ is the k-th mean vector derived from the estimated GMM;
- $\tilde{\Sigma}_k$ is the k-th covariance matrix derived from the estimated GMM;
- $\tilde{\pi}_k$ is the k-th mean vector derived from the estimated GMM, with $\sum_{k=1}^{K} \pi_k = 1$;
- ε is the (exploration) exploratory factor, with $0 \leq \varepsilon \leq 1$, so we have maximum exploration (high variability) for $\varepsilon \to 1$, while the minimum exploration (low variability) is reached for $\varepsilon \to 0$;
- $\tilde{V}[\dot{\xi}|\xi, \theta]$ is the overall variance of the estimated GMM connecting the variability generated by human demonstrators to the system exploration.

A novel trajectory can be generated from the computed DMM. If this new trajectory fails in reaching the task, the model should be updated reiteratively in order to get success. A first technique consists of recomputing the model using the EM process on the entire dataset plus the generated trajectory. This method grows in time and memory proportionally to the input dataset. An alternative uses a sample and merge approach to maintain constant both time and memory.

4.7 Density Function Maximization

The DMM probability density function have to be maximized in order to generate the new trajectory. Differently from GMM, DMM has no analytical solution to the problem, so an optimization technique should be used. Grollman [35] proposed gradient ascendant (steepest gradient) to find a maximum around an initial guess. The result could be both global or local maxima depending on the starting point.

During this work, we analyzed different optimization techniques and several implementations comparing them on different aspects like convergence, number of iterations and starting point. Optimization methods we tested can be divided into three categories: Quasi-Newton methods, Conjugate Gradient methods and Simplex-based methods.

4.7.1 Quasi-Newton methods

These methods find a stationary point of a function by using a Newton-based method. They look at the gradient and the Hessian matrix of second derivatives in order to converge to a stationary point (where the gradient is 0) by assuming that the function can be locally approximated as a quadratic in the region around the optimum. In particular, quasi-Newton methods approximates the inverted Hessian matrix by mean of successive gradient vectors, so that it does not need to be computed directly.

Davidon-Fletcher-Powell method

The DavidonFletcherPowell (DFP) algorithm [20] [26] was the one of the first quasi-Newton method created and the first one that generalize the secant method to a multidimensional problem.

Given $f : \mathbb{R}^n \to \mathbb{R}$ with $x_0 \in \mathbb{R}^n$ starting point in the f domain, $H_0 = I$ identity matrix, initial approximation of the Hessian matrix and $d_0 = H_0 \nabla f(x_0)$. At each iteration k the point x_k and the matrix H_k are updated using the formulas:

$$x_k = x_{k-1} + \Delta x_{k-1} \tag{4.12}$$

$$H_k = H_{k-1} + \frac{\Delta x_{k-1} \Delta x_{k-1}^\top}{\Delta x_{k-1}^\top q_{k-1}} + \frac{H_{k-1} q_{k-1} q_{k-1}^\top H_{k-1}}{q_{k-1}^\top H_{k-1} q_{k-1}} \tag{4.13}$$

where

- $\Delta x_{k-1} = \lambda_{k-1} d_{k-1}$ is the Displacement Vector;
- λ_{k-1} is the value of λ that minimizes $f(x_{k-1} + \lambda d_{k-1})$;
- $d_{k-1} = H_{k-1} \nabla f(x_{k-1})$ is the direction to move to find the maxima, it is also used as stopping criterion in the case it is lower than a predefined threshold;
- $q_k = \nabla f(x_k) - \nabla f(x_{k-1})$.

This method is quite effective because it simultaneously generates conjugate directions and constructs an approximation of the inverse of the Hessian matrix.

Broyden-Fletcher-Goldfarb-Shanno method

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [12] [25] [32] [84] is one of the most powerful methods to solve unconstrained optimization problem.

Given f convex, continuously differentiable function to maximize, x_0 starting point in the f domain and B_0 initial approximation of the Hessian matrix set to any symmetric positive definite matrix (i.e. identity matrix), at each iteration k the point x_k and the matrix B_k are updated using the formulas:

$$x_{k+1} = x_k + t\Delta x \tag{4.14}$$

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k} \tag{4.15}$$

where

- t is the step size;
- $\Delta x = -B_{k-1}^{-1} \nabla f(x_{k-1})$ is the direction to move to find the maxima, it is also used as stopping criterion in the case it is lower than a predefined threshold;
- $s_k = x_k - x_{k-1}$;
- $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$.

This method assures fast convergence, it avoids calculation of second-order information, that is computationally expensive using numerical differentiation. BFGS is a general purpose method, very effective in several situations, and with a minimal variation also with large scale applications.

4.7.2 Conjugate Gradient methods

Conjugate Gradient methods are a class of algorithms for the numerical solution of systems of linear equations. They are often implemented as iterative algorithms. These methods typically require less computation than a Quasi-Newton as there is no need to compute the Hessian matrix, on the other hand they tend to converge in more iterations. Conjugate Gradient methods also require less memory to operate.

Fletcher-Reeves method

The Fletcher-Reeves algorithm [27] belongs to a group of methods called nonlinear conjugate gradient methods able to generalize the techniques used in conjugate gradient methods to nonlinear optimization.

Given a quadratic function $g(x) = \|Ax - b\|^2$ the minimum is obtained when $\nabla g(x) = 2A^\top (Ax - b) = 0$. The method searches for a minimum in a given nonlinear function $f : \mathbb{R}^n \to \mathbb{R}$ by approximating it with a quadratic near the area of interest, which is the case when the function is twice differentiable in the interval.

Considered $x_0 \in \mathbb{R}^n$ starting point in the f domain and $s_0 = -\nabla f(x_0)$ initial conjugate direction. At each iteration k the point x_k and the subsequent conjugate direction s_k are updated using the formulas:

$$x_k = x_{k-1} + \alpha_{k-1} s_{k-1} \tag{4.16}$$

$$s_k = -\nabla f(x_k) + \beta_k s_{k-1} \tag{4.17}$$

$$\beta_k = \frac{\nabla f(x_k)^\top \nabla f(x_k)}{\nabla f(x_{k-1})^\top \nabla f(x_{k-1})} \tag{4.18}$$

where

- α_{k-1} is the value of α that minimizes $f(x_{k-1} + \alpha s_{k-1})$;
- β_k is the updating factor;
- $\|\nabla f(x_k)\|$ is used as stopping criterion in the case it is lower than a predefined threshold.

Polak-Ribiere method

The Polak-Ribiere algorithm [76] belongs to the same group than Fletcher-Reeves method. This method approximate the nonlinear function $f : \mathbb{R}^n \to \mathbb{R}$ in the interval around a presumed minimum starting by an initial point $x_0 \in \mathbb{R}^n$ in the f domain and, as before, $s_0 = -\nabla f(x_0)$ initial conjugate direction. The update formula differentiates the two methods: at each iteration k the point x_k and the subsequent conjugate direction s_k are updated by using:

$$x_k = x_{k-1} + \alpha_{k-1} s_{k-1} \tag{4.19}$$

$$s_k = -\nabla f(x_k) + \beta_k s_{k-1} \tag{4.20}$$

$$\beta_k = \frac{(-\nabla f(x_k))^\top (\nabla f(x_{k-1}) - \nabla f(x_k)}{\nabla f(x_{k-1})^\top \nabla f(x_{k-1})} \tag{4.21}$$

where

- α_{k-1} is the value of α that minimizes $f(x_{k-1} + \alpha s_{k-1})$;

- β_k is the updating factor;
- $\|\nabla f(x_k)\|$ is used as stopping criterion in the case it is lower than a predefined threshold.

4.7.3 Simplex-based methods

The latter category iteratively approximates a stationary point of a n dimensional function with a special polytope of $n+1$ vertices called Simplex when the objective function varies smoothly and is unimodal. The algorithm starts from an initial Simplex. Then it calculates the objective function for each point in the Simplex. The worst performing point is replaced by a new testing position generated from the ones composing the Simplex. Depending on the situation the new point will reflect (Figure 4.5), expand (Figure 4.6), contract(Figure 4.7) or shrink (Figure 4.8)the initial Simplex. The algorithm iterates until a stopping criteria is satisfied or the maximum number of iterations is reached.

Nelder-Mead method

The NelderMead algorithm, also called Downhill Simplex or Amoeba method, is the most commonly used Simplex-based optimization technique. It is computationally quite simple, because it does not require any derivative information, which makes it suitable for problems with non-smooth functions.

The initial $n+1$ vertices composing the Simplex $S = (x_0, x_1, ..., x_n)$ are generated around a starting point $x_0 \in \mathbb{R}^n$ by satisfying two constraints:

- $x_j = x_0 + h_j e_j, j = 1, ..., n$ where h_j is the step-size of the direction represented by the versor e_j;
- S is a regular Simplex: all the edges have the same length.

At each iteration k the Simplex S and consequently the $N+1$ points $x_j, j = 0, 1, ..., n$ are updated using the following procedure. The algorithm determines

- $x_h = \text{argmax}_j(f(x_j))$ the worst performing point;
- $x_s = \text{argmax}_{j \neq h}(f(x_j))$ the second worst performing point;

- $x_l = \text{argmin}_j(f(x_j))$ the best performing point;
- $c = \frac{1}{n}\sum_{j \neq h} x_j$ the centroid of the n best points.

Depending on the four parameters:

- α the reflection parameter,
- β the expansion parameter,
- γ the contraction parameter,
- δ the shrinkage parameter;

the worst vertex x_h is replaced by a better test point lying on the line defined by x_h and c. If no better point is founded, the Simplex S is shrank towards the best vertex x_l and n new vertices are computed. The selected parameters have to satisfy the constraints $\alpha > 0, 0 < \beta < 1, \gamma > 1 \wedge \gamma > \alpha, 0 < \delta < 1$. In our implementation $\alpha = 1, \beta = \frac{1}{2}, \gamma = 2, \delta = \frac{1}{2}$.

To find the new point, the reflection point $x_r = c + (\alpha - x_h)$ is computed. If $f(x_l) \leq f(x_r) < f(x_s)$ then x_r is accepted as new point (Figure 4.5). Otherwise, if $f(x_r) < f(x_l)$ the expansion point $x_e = c + \gamma(x_r - c)$ is calculated, x_e is accepted if $f(x_e) < f(x_r)$ (Figure 4.6), if not x_r is accepted. While if $f(x_r) \geq f(x_s)$ the contraction point x_c is used: in this algorithm the formula to compute the contraction point use the better performing point between x_h and x_r. If $f(x_s) \leq f(x_r) < f(x_h)$ then $x_c = c + \beta(x_r - c)$ and an outside contraction (Figure 4.7 (a)) will be applied if $f(x_c) \leq f(x_r)$. Instead, if $f(x_r) \geq f(x_h)$ then $x_c = c + \beta(x_h - c)$ and an inside contraction (Figure 4.7 (b)) will be applied if $f(x_c) < f(x_h)$. If no one of the previous condition is satisfied the algorithm perform a shrink transformation (Figure 4.8) by computing the n new vertices $x_j = x_l + \delta(x_j x_l)$ for $j \neq l$.

4.7.4 Methods performance analysis

The algorithms described have been used to calculate the maximum in a well-know function in order to evaluate the number of iterations in which each algorithm converges to the maximum. The starting point was $x_0 = 0$. The results are showed in Table 4.1. The same algorithms are tested on the probability density

function of a randomly generated Donut Mixture Model: the collected data are reported in Table 4.2.

Table 4.3 shows more tests comparing a gradient ascendant implementation and the BFGS algorithm performing better between the algorithms we tested. Three initial points were considered on a DMM generated randomly. The convergence of the algorithm, the number of iterations, the resulting point and its corresponding value in the Donut are considered.

Table 4.1: Comparison between different implementation of the described optimization algorithms on a well-know function.

Method	**Convergence**	**Iterations**	x_0	x_{max}	$f(x_{max})$
Grad. Asc.[1]	Yes	143	0.0	-0.0222	17.126
DFP[2]	Yes	6	0.0	-0.0222	17.126
BFGS 1[1]	Yes	6	0.0	-0.0222	17.126
BFGS 2[1]	Yes	2	0.0	-0.0222	17.126
BFGS 3[2]	Yes	6	0.0	-0.0222	17.126
FR[1]	Yes	6	0.0	-0.0222	17.126
PR[1]	Yes	6	0.0	-0.0222	17.126
NM 1[1]	Yes	6	0.0	-0.0222	17.126
NM 2[1]	Yes	6	0.0	-0.0222	17.126
NM Rand[1]	Yes	6	0.0	-0.0222	17.126

The result still could be a local maximum, this is an intrinsic problem due to use of gradient as moving direction. It mainly depends from the designed starting point, for example if it is near to a local maximum. Another typical error might happen when the curvature near to the starting point is lower than the threshold and the algorithm stops at the first iteration even if the selected point is in the Donut tail.

In order to avoid these situations, we introduced two polices:

- *Start Check (SC)*: the starting point is changed if the neighborhood presents a gradient already lower than the selected threshold;

[1]This implementation was based on the GNU Scientific Library (GSL) `http://www.gnu.org/software/gsl/`

[2]This implementation was based on MATLAB Unconstrained Nonlinear Optimization

Table 4.2: Comparison between different implementation of the described optimization algorithms on the probability density function of a randomly generated Donut Mixture Model.

Method	**Convergence**	**Iterations**	x_0	x_{max}	$f(x_{max})$
Grad. Asc.[1]	No	250	0.0	-0.10989	3.51196
DFP[2]	Yes	111	0.0	-0.9353	5.24314
BFGS 1[1]	Yes	23	0.0	-0.9353	5.24314
BFGS 2[1]	Yes	7	0.0	-0.9353	5.24314
BFGS 3[2]	Yes	18	0.0	-0.9353	5.24314
FR[1]	Yes	29	0.0	-0.9353	5.24314
PR[1]	Yes	24	0.0	-0.9353	5.24314
NM 1[1]	Yes	38	0.0	-0.9353	5.24314
NM 2[1]	Yes	38	0.0	-0.9353	5.24314
NM Rand[1]	Yes	42	0.0	-0.9353	5.24314

Table 4.3: A more detailed comparison between BFGS 2[1] and Gradient Ascendant[1] optimization algorithms on the probability density function of a randomly generated Donut Mixture Model.

Method	**Convergence**	**Iterations**	x_0	x_{max}	$f(x_{max})$
Grad. Asc.[1]	No	250	-0.02	-0.0029	114.23
BFGS 2[1]	Yes	4	-0.02	-0.0029	114.23
Grad. Asc.[1]	Yes	194	0.0	-0.0029	114.23
BFGS 2[1]	Yes	2	0.0	-0.0029	114.23
Grad. Asc.[1]	No	250	0.02	-0.0029	114.23
BFGS 2[1]	Yes	5	0.02	-0.0029	114.23

- *Multi-way Search (MW)*: the research of the maximum is reiterated starting from S different initial points simultaneously. The number of starting points S can vary with the number of processors and the response time requested by the application. In this work, we set $S = K$, and the starting points are the means of the K Donut components. Finally, we considered the overall maximum, thus only the best resulting maximum is used. This policy allows us to avoid local maxima that lead to sub-optimal trajectories.

In Table 4.4, three different tests are reported in which *SC* policy is applied to non converging or local converging BFGS instances. The implementation used

is the one giving us the best performances in the previous tests. Table 4.5 shows the *MW* policy with $S = 4$ compared with two different standard BFGS iterations. For both Table 4.4 and Table 4.5 the same DMM generated for the Table 4.3 comparison was used as function to maximize.

Table 4.4: A more detailed comparison between BFGS 2^1 and Gradient Ascendant[1] optimization algorithms on the probability density function of a randomly generated Donut Mixture Model.

Method	**Conv.**	**L/G**	**Iters**	x_0	$\nabla f(x_0)$	x_{max}	$f(x_{max})$
BFGS 2^1	No	-	1	-0.25	0.0	-0.25	0.0
BFGS 2^1 + SC	Yes	G	4(8)	-0.031	1.25	-0.0029	114.23
BFGS 2^1	Yes	L	2	-0.063	-2.48 E-19	-0.0053	-1.47 E -17
BFGS 2^1 + SC	Yes	G	5(3)	-0.034	0.027	-0.0029	114.23
BFGS 2^1	Yes	L	5	-0.114	0	-0.10	-1.73 E -84
BFGS 2^1 + SC	Yes	G	3(5)	0.006	-197.3	-0.0029	114.23

Table 4.5: Comparison between standard BFGS 2^1 and the same algorithm plus the MW policy.

Method	**Conv.**	**L/G**	**Iters**	x_0	$\nabla f(x_0)$	x_{max}	$f(x_{max})$
BFGS 2^1	No	-	1	-0.07	0.0	-0.07	0.0
BFGS 2^1	Yes	G	5	-0.038	7.29 E-6	-0.0029	114.23
MW 1	Yes	L	2	-0.063	-2.48 E-19	-0.0053	-1.47 E -17
MW 2	Yes	G	3	-0.062	2597.8	-0.0029	114.23
MW 3	Yes	G	5	-0.001	-2026.36	-0.0029	114.23
MW 4	Yes	G	3(5)	0.024	0.24	-0.0029	114.23

4.8 Kinesthetic demonstrations

The developed framework was initially tested by using the kinesthetic demonstration technique. Joints positions and velocities in input to the Donut Mixture Model were collected from the robot manually moved by a human being.

The task selected was throwing a ball to score a basket and the robot used for these tests was a small humanoid: the Aldebaran NAO (Figure 4.9). The only joint

involved in the the task was the right shoulder pitch. The ball was autonomously released by opening the robot fingers when the angle overcomes a threshold. At each trial the robot arm was placed at an initial angle of -2.0857 rad, the person moved it forward until the ball was released at -1.30 rad, so that the joint angles and velocities were the only relevant information we had to acquire. At each instant t, the joint angle ξ_t came directly from the encoder, while the velocity was computed as the single-step difference between angle samples: $\dot{\xi}_t = \xi_t - \xi_{t-1}$.

The choice of a good optimization method is crucial in order to manage data provided by devices working at 20-30 Hz, like the encoders mounted on the NAO. Moreover, the number of the collected trajectories is higher with respect to other similar kinesthetic demonstration tests in which the data were recorded at 500 Hz ([34]). In Table 4.6 are reported the 30 failure demonstrating trajectories used to train the Donut Mixture Model, while Table 4.7 show the generated trajectories, the successful trial are highlighted. In Figure 4.10, the blue trajectories corresponds to the initial dataset, while the ones in red were generated by the framework.

4.9 Human observations

Finally, our approach is also tested using the skeleton data provided in the *Throw Over Head* action from the IAS-Lab Action Dataset (Section A.1) [69] [66], in which 12 actors threw a ball using a basketball approach (Figure 4.11). The arm is raised, the elbow is bent, the limb is extended, and the wrist is rotated so that the ball reaches the target; the entire movement is correctly repeated 3 times for each actor. The acquired data were refined as described in Section 3.2 and remapped to the robot joint space. From the original 36 trajectories, we removed 2 samples in which the number of skeleton joint data provided is too low to be effective in the robot movement.

These demonstrations are the input to our Learning from Demonstration framework. The aim is to make an Aldebaran NAO learn how to play basketball. The robot should be able to put the ball into a basket placed 40 cm in front of it (Figure 4.12). The results of the system are compared with the output trajectories computed using well-know GMM/GMR framework [17] suitable for robot

learning when the task is successfully performed by the demonstrators. As in the original article, the means of the Gaussian components are used to generate the trajectory from the system.

Looking at our input trajectories we notice, as we expected, that only few (11 over 34) throws enter correctly into the basket, the remaining 23 throws score no points; even if all the 34 demonstrations were successfully performed by the actors. This result is due to difference between the human and the robot DoFs, and, as we said before, we chose a Learning from Failure Demonstrations framework to model human observations mainly for this reason. It is also worth to notice that the difference is further augmented by the noise introduced by the sensor.

We divided the 34 demonstrations into three groups:

- successful and failed demonstrations (34 trajectories)
- failed demonstrations only (23 trajectories)
- successful demonstrations only (11 trajectories)

for each of these groups we generated 15 new trajectories coming from 15 different DMM, where the 1^{st} DMM models the initial dataset (34, 23 or 11 demonstrations), the 2^{nd} DMM models the initial dataset plus the previously generated trajectory, the n^{th} DMM models the initial dataset plus the $(n-1)^{th}$ trajectories generated previously. Differently than in Grollman's work, where the generation is stopped when a successful trajectory comes out, we preferred to generate all the 15 trajectories in order to observe the system behavior even with a high number of correct demonstrations.

In the first group, the input trajectories are both correct and incorrect. The generated trajectories are showed in Figure 4.13 (a), the 15 trajectories from our framework are drawn in red; in black the one generated from the GMM/GMR framework; the input trajectories are dotted in blue. The 5th, 6th, 13th and 15th trajectories scored a point, while the one from GMM/GMR did not.

The results from the second group are showed in Figure 4.13 (b). As before the 23 incorrect input trajectories are dotted in blue; in red the ones from our framework; the trajectory generated by the GMM/GMR framework is black. Since this group is composed by failed demonstrations only, we expected an improvement in

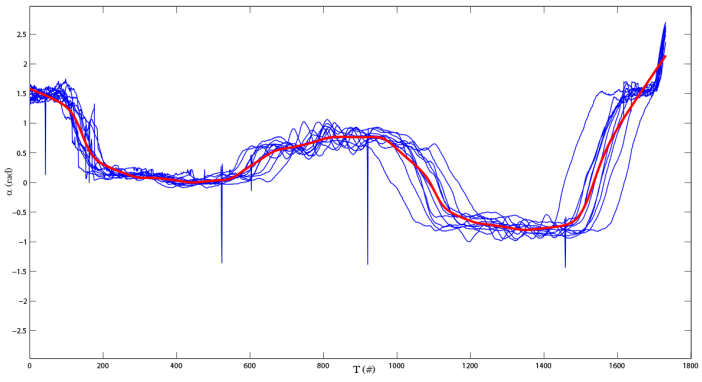
our framework performances and still a bad throw from GMM/GMR framework. As we expected, only 5 generated trajectories failed, the 2nd, the 6th, the 8th, the 13th and the one from GMM/GMR.

Figure 4.13 (c) shows the third group results. As usual the 11 successful trajectories are dotted in blue; the ones in red are generated from our framework; the GMM/GMR trajectory is in black. Differently from what we thought, our framework overcomes the GMM/GMR one even in this group. In fact, the GMM/GMR throw touches the basket border, but finally goes out. On the other hand, our framework generated only 2 failed trajectories: the 1st and the 11th.

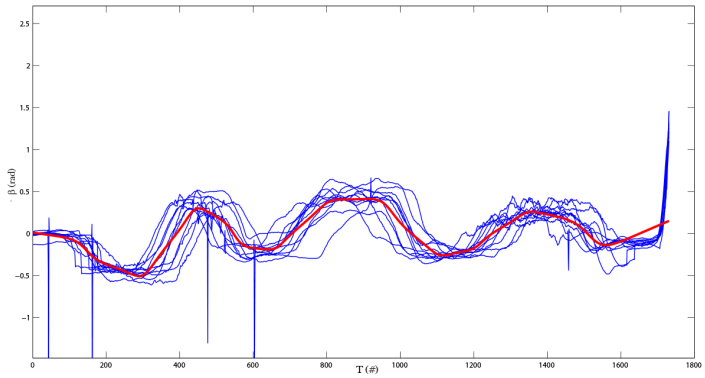
4.10 Summary

In this chapter, two different Robot Learning from Demonstration framework were developed. The former was based on Gaussian Mixture Model and Gaussian Mixture Regression, the latter used Donut Mixtures to model information. Both frameworks were tested in modeling natural demonstrations. The GMM/GMR framework was able to correctly reproduce tasks in a high repeatability industrial environment. The same framework did not work as well with different interpretations of the same task that occurred in a more dynamic service robotics environment. The DMM framework introduced the proper flexibility showed to be a able to overcome the differences between the different demonstrations.

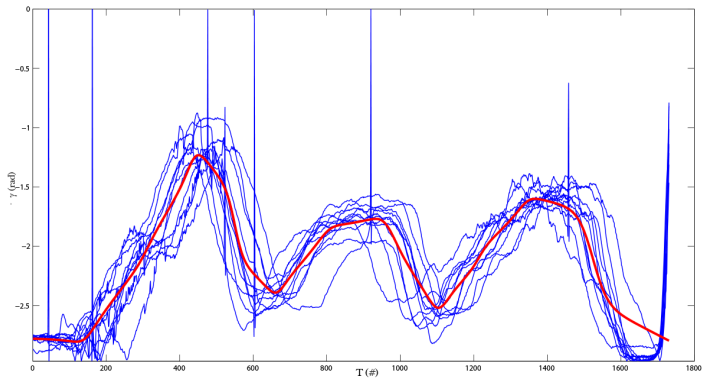
In the future, we will focus on time performances in order to make both frameworks work real-time.



(a) Results of GMM/GMR applied to *Axis2*.



(b) Results of GMM/GMR applied to *Axis2*.



(c) Results of GMM/GMR applied to *Axis3*.

Figure 4.4: Results obtained in the first task using a GMM trained with 11 examples (blue) to calculate the regression (red) through GMR for *Axis1* (a), *Axis2* (b), *Axis3* (c). The vertical straight-lines peaks are out-layers due to the sensor noise. Using the GMM/GMR model the robot avoid rapid accelerations between timesteps and large oscillations in its velocity.

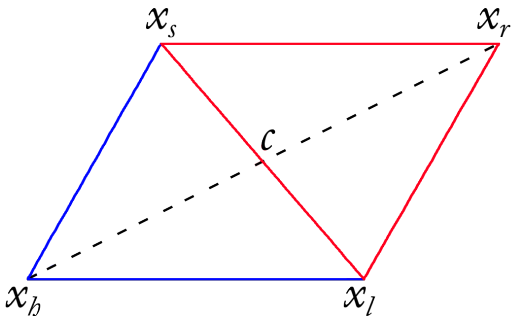


Figure 4.5: Simplex reflection operation applied to the current polygon composed by 3 vertices.

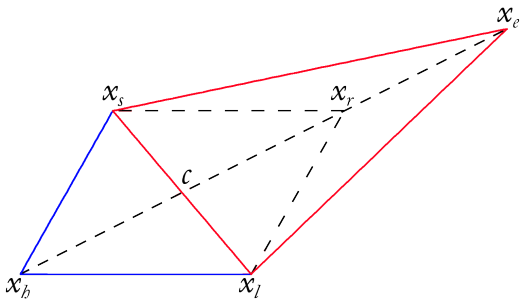


Figure 4.6: Simplex expansion operation applied to the current polygon composed by 3 vertices.

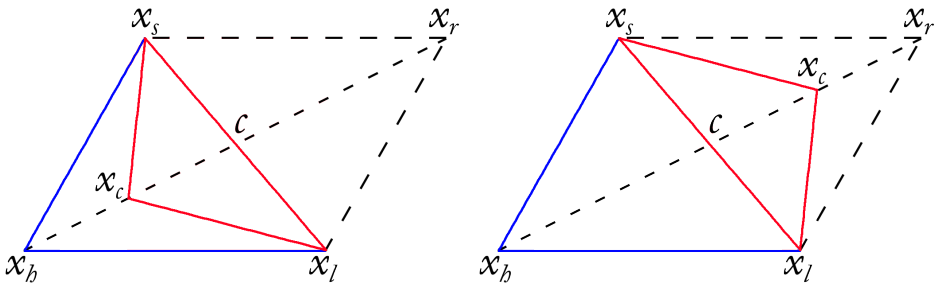


Figure 4.7: Simplex inside contraction (a) and outside contraction (b) operation applied to the current polygon composed by 3 vertices.

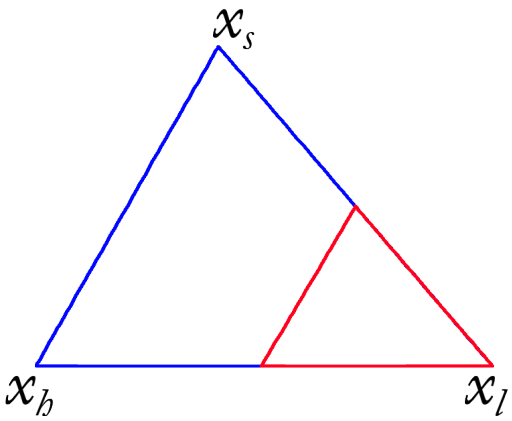


Figure 4.8: Simplex shrinkage operation applied to the current polygon composed by 3 vertices.

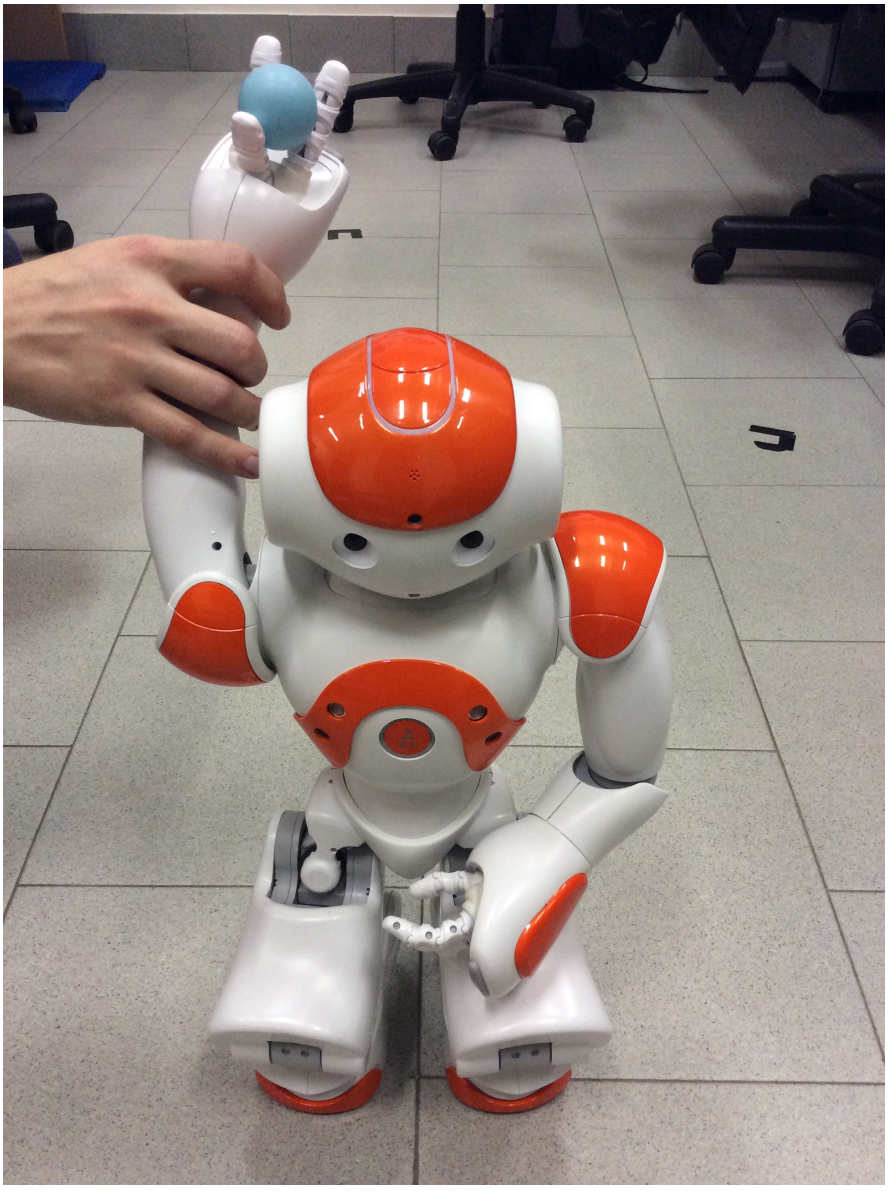


Figure 4.9: Overview of the experimental scenario: the NAO is manually moved by a human to score in a basket placed at 40 cm from it.

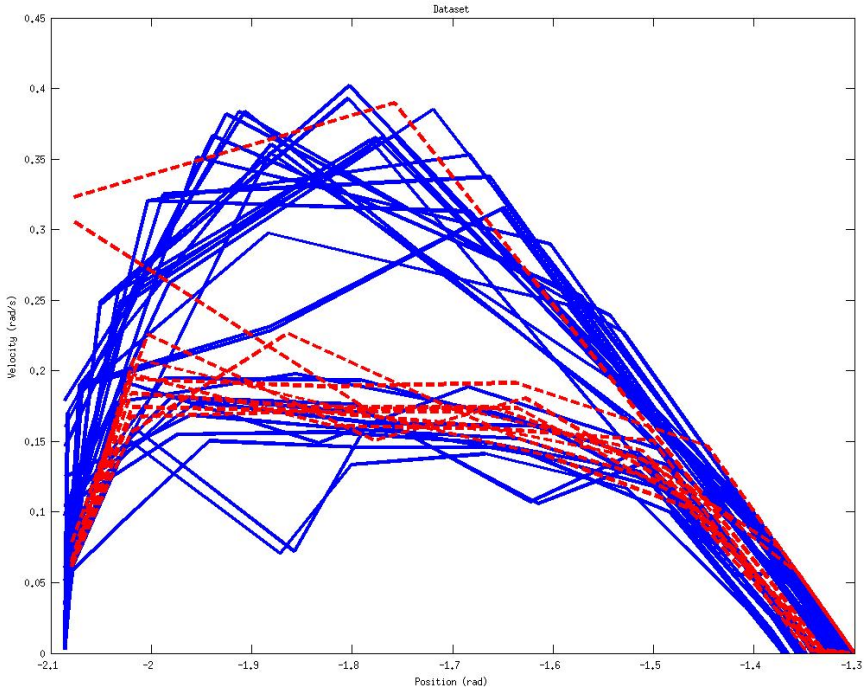


Figure 4.10: Trajectories used to generate the DMM during the kinesthetic demonstration task. The blue dotted trajectories corresponds to the initial dataset, while the ones in red were generated by the framework.

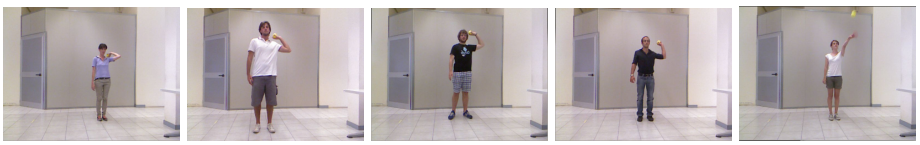


Figure 4.11: Screen-shots from the *Throw Over Head* action performed by different actors in the IAS-Lab Action Dataset.

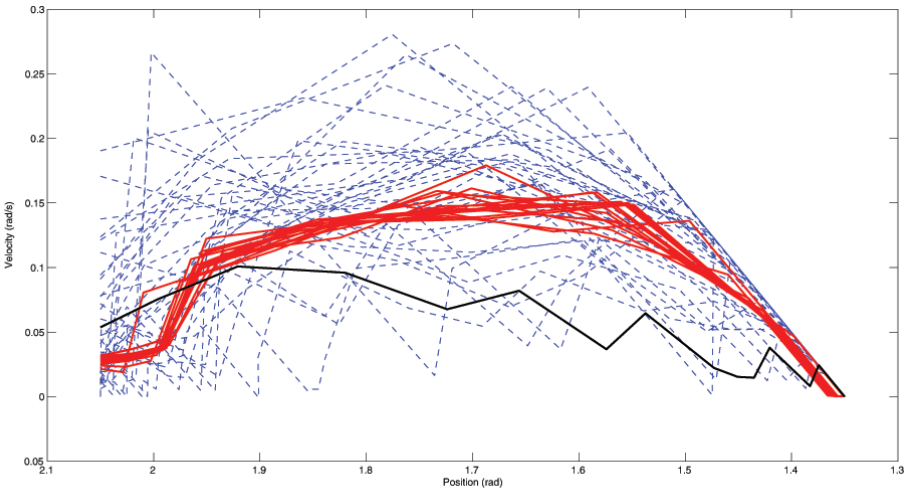
Table 4.6: The trajectories collected by using kinesthetic demonstrations.

Trail	Datapoints	Initial ξ	Final ξ	$\dot{\xi}_{max}$	Result
1	7	-2.0857	-1.34	0.29	TP
2	5	-2.0857	-1.30	0.52	TD
3	8	-2.0857	-1.36	0.22	P
4	5	-2.0857	-1.30	0.59	TD
5	5	-2.0857	-1.30	0.54	TD
6	4	-2.0857	-1.32	0.56	TD
7	8	-2.0857	-1.35	0.26	P
8	9	-2.0857	-1.36	0.21	P
9	5	-2.0857	-1.30	0.57	TD
10	7	-2.0857	-1.34	0.25	P
11	7	-2.0857	-1.34	0.30	TP
12	4	-2.0857	-1.30	0.58	D
13	5	-2.0857	-1.31	0.55	TD
14	5	-2.0857	-1.31	0.64	TD
15	7	-2.0857	-1.35	0.24	P
16	5	-2.0857	-1.30	0.61	TD
17	5	-2.0857	-1.30	0.60	TD
18	7	-2.0857	-1.33	0.27	P
19	6	-2.0857	-1.31	0.34	TD
20	7	-2.0587	-1.33	0.28	P
21	9	-2.0857	-1.37	0.20	P
22	8	-2.0857	-1.35	0.23	P
23	6	-2.0857	-1.32	0.32	TP
24	6	-2.0857	-1.32	0.33	TD
25	5	-2.0857	-1.32	0.62	D
26	5	-2.0857	-1.30	0.63	D
27	5	-2.0857	-1.31	0.51	TD
28	6	-2.0857	-1.30	0.50	TD
29	5	-2.0857	-1.31	0.64	D
30	6	-2.0857	-1.31	0.31	TP

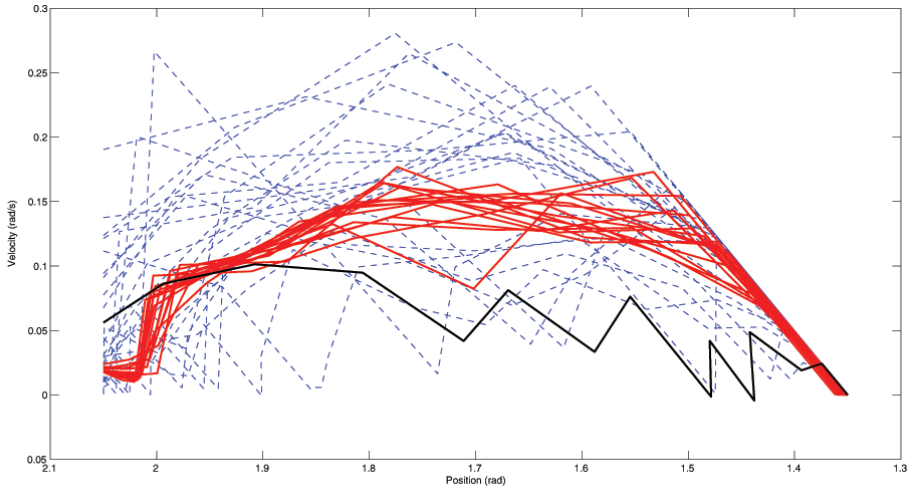
Table 4.7: The trajectories collected by using kinesthetic demonstrations.

Trail	Datapoints	Initial ξ	Final ξ	$\dot{\xi}_{max}$	Result
1	43	-2.0857	-1.30	0.38	D
2	51	-2.0857	-1.30	0.22	C
3	52	-2.0857	-1.31	0.17	TP
4	60	-2.0857	-1.30	0.19	C
5	59	-2.0857	-1.30	0.30	TP
6	70	-2.0857	-1.30	0.17	TP
7	70	-2.0857	-1.30	0.18	TP
8	56	-2.0857	-1.30	0.22	TP
9	46	-2.0587	-1.30	0.20	C
10	57	-2.0857	-1.31	0.19	C

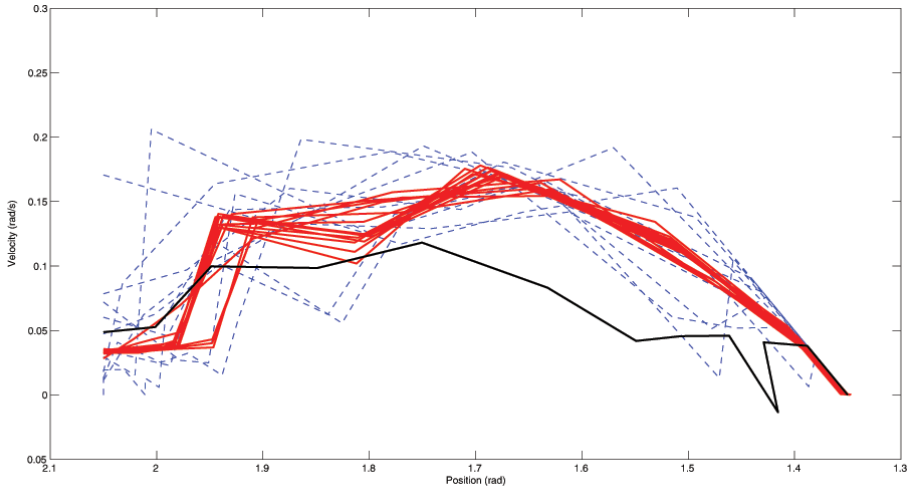
Figure 4.12: Overview of experimental scenario: the NAO is placed at 40 cm from the basket.



(a) Generated trajectories from both successful and failed demonstrations.



(b) Generated trajectories from failed demonstrations only.



(c) Generated trajectories from successful demonstrations only.

Figure 4.13: Results obtained from the demonstrations collected from human observations (blue dots) using our framework (in red) and a GMM/GMR framework (black).

Chapter 5

Conclusions

In this thesis a novel paradigm for Robot Learning from Demonstration was explored. The aim was to learn new skills from human natural observations to foster the development of service robotic. In fact, if we think at personal robots to help people in everyday life, one of the most important characteristics would the ability to learn new tasks depending on the circumstances. Recognize actions performed from people, generalize them, and reproduce similar tasks in different situations are basic bricks composing the puzzle. During this work we analyzed all these aspects using as input device a small and inexpensive RGB-D sensor, deeply tested in an home environment and able to acquire both depth and color information.

In Chapter 2, we compared two approaches to human action recognition based on RGB-D data. The former estimate joint positions and orientations from a skeleton tracking system. The latter computes 3D motion flow by exploiting joint color and depth information. Both of them have been classified with a Nearest Neighbor classifier for performing action recognition. The two algorithms were tested on a large variety of actions and actors available in a our own novel publically released action dataset which provides RGB and depth data as well as skeleton information. Particularly good results were obtained from the 3D motion flow approach that was able to outperform the skeleton based technique reaching about 85% of accuracy. As future works we plan to automatize the segmentation procedure to test action recognition on autonomous mobile robots.

In Chapter 3, we presented fast on-line procedure to re-target human motion to a small humanoid called Robovie-X by analyzing both upper limbs and lower limbs. A more accurate analysis of the upper body motion was also reported and applied on a different and more complex humanoid, namely the Aldebaran NAO. The stability issue in lower body motion re-targeting was discussed by using a toy example in which a human operator have to teleoperate the robot to pick up an object laying in front of it. Finally we proposed a novel re-targeting approach involving whole body motion to control an industrial manipulator, the Comau Smat5 SiX. All the described re-targeting techniques were designed to be used by naive people acting in natural way. As future works, we schedule to extend the stability algorithm capabilities and perform more challenging tests on both the proposed motion re-targeting systems.

In Chapter 4 we proposed different Robot Learning from Demonstration frameworks to model the data provided from the re-targeting system in a probability manner. A first framework based on Gaussian Mixture Model and Gaussian Mixture Regression was successfully used to model natural demonstrations in an industrial environment. The model also highlighted a good precision due to the intrinsic repetitive nature of the tested tasks. In a more dynamic situation, the same framework showed flexibility limitations. Thus, a different framework based on Donut Mixture Model was introduced. This model can work properly even with failed demonstrations, so it is flexible enough to failed robot attempts and different interpretations of the same task. We deeply discussed the optimization techniques used to maximize the probability density function of the Donut Mixture in order to improve the performance of the model. The use of Broyden-Fletcher-Goldfarb-Shanno optimization algorithm leaded us to obtain good results even with a few input datapoints. The Donut Mixture Model framework was tested on a simple task, namely score a basket. Both kinesthetic and natural demonstrations were used for testing. Natural demonstrations was extracted from the *IAS-Lab Action Dataset*by using the action *Throw Over Head* Results showed that the Donut Mixture Model framework was able to outperform the GMM/GMR framework in all the considered configurations. As future work, we plan to try out the model with more complex task in order to verify the scalability of the system.

This thesis contributed to Robot Learning from Demonstration by proposing

a generic probabilistic framework able to let users act in a natural way. The effectiveness of the proposed system open new interesting possibilities by relaxing the restrictive constrain that forced demonstrators to robot understandable modalities. Moreover, consumer RGB-D sensors, like the ones used in this thesis, are going to be more and more precise and resolute. Consequently, we expect to obtain a corresponding improvement also in the techniques proposed in this work.

Appendices

Appendix A

RGB-D Datasets

A.1 IAS-Lab Action dataset

As we already said, the rapid dissemination of inexpensive RGB-D sensors boosted the research on several field, included action recognition. At the same time a new need arose: the acquisition of new datasets in which the RGB stream is aligned with the depth stream. Currently, the following action recognition 3D datasets have been released: RGBD-HuDaAct Database [97], Indoor Activity Database [87], MSR-Action3D Dataset [54], MSR-DailyActivity3D Dataset [91], LIRIS Human Activities Dataset [92] and Berkeley MHAD [74]. All these datasets are targeted to recognition tasks in indoor environments. The first two are thought for personal or service robotics applications, while the two from MSR are also targeted to gaming and human-computer interaction. The LIRIS dataset concerns actions performed from both single persons and groups, acquired in different scenarios and changing the point of view. The last one was acquired using a multimodal system (mocap, video, depth, acceleration, audio) to provide a very controlled set of actions to test algorithms across multiple modalities.

A.1.1 *IAS-Lab Action Dataset*

Two key features of a good dataset are size and variability. Moreover, it should allow to compare as many different algorithms as possible. For the RGB-D action recognition task, that means that there should be enough different actions, many

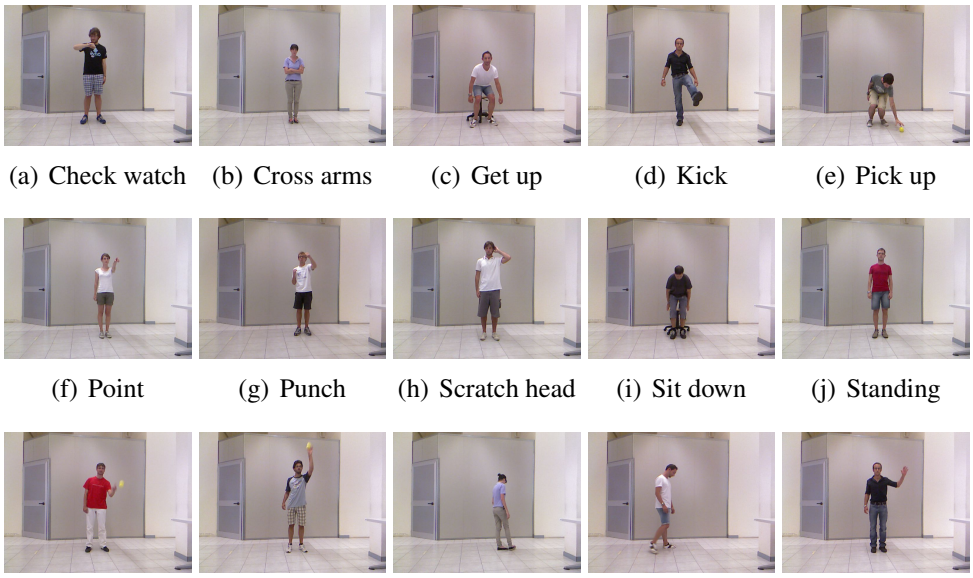
Table A.1: Datasets for 3D Human Action Recognition.

	#actions	**#people**	**#samples**	**RGB**	**skel**
[97]	6	1	198	yes	no
[87]	12	4	48	yes	yes
[54]	20	10	567	no[2]	yes
[91]	16	10	320	no	yes
[92][3]	10	21	461	yes[4]	no
[74]	11	12	660	yes	yes[5]
Ours	15	12	540	yes	yes

different people performing them and RGB and depth synchronization and registration. Moreover, the 3D skeleton of the actors should be saved, given that it is easy available and many recent techniques rely on it. Hovewer, we noticed the lack of a dataset having all these features, thus we acquired the *IAS-Lab Action Dataset*[1], which contains 15 different actions performed by 12 different people. Each person repeats each action three times, thus leading to 540 video samples. All these samples are provided as ROS `bags` containing synchronized and registered RGB images, depth images and point clouds and ROS `tf` for every skeleton joint as they are estimated by the NiTE middleware. Unlike [74], we preferred NiTE's skeletal tracker to a motion capture technology in order to test our algorithms on data that could be easily available on a mobile robot and, unlike [92], we asked the subjects to perform well defined actions, because, beyond a certain level, variability could bias the evaluation of an algorithm performance.

In Table A.1, the *IAS-Lab Action Dataset* is compared to the already mentioned datasets, while in Figure A.1 an example image for every action is reported.

[1] `http://robotics.dei.unipd.it/actions`.
[2] The RGB images are provided, but they are not synchronized with the depth images.
[3] Only the set provided with depth information was considered.
[4] The RGB information has been converted to grayscale.
[5] Obtained from motion capture data.



(a) Check watch (b) Cross arms (c) Get up (d) Kick (e) Pick up

(f) Point (g) Punch (h) Scratch head (i) Sit down (j) Standing

(k) Throw from bottom up (l) Throw over head (m) Turn around (n) Walk (o) Wave

Figure A.1: Examples of images for the 15 actions present in the dataset.

Appendix B

Robot models

B.1 Virtual robots in ROS

In this work, the Unified Robot Description Format (URDF) has been used to create an accurate model to simulate the physical robot in a ROS [80] environment.

The URDF is the ROS standard way to represent a robot model. This format can be read from the ROS Visualizer (RViz) and any simulator integrated in ROS should be able to import this kind of models. Tn this thesis we analyzed two specific simulator: Gazebo and VRep.

B.1.1 Gazebo

Gazebo [45] is an Open Source 3D simulator which is capable of simulating a wide population of robots, sensors and objects. The project started in 2002 at the University of Southern California. It was designed to help researchers working on robotic vehicles in outdoor environments, but it manages also indoor situations. Gazebo has historically been used as a research and development tool to rapid prototyping, locomotion, robot competition, person simulation, and regression testing. It nominally provides multiple physics engines including ODE [41] and Bullet [40] (not completely supported also in the last software release) and general parameters such as accuracy and performance are exposed to better suite to user needs. Gazebo relies on OGRE [42] to render 3D graphics and improve re-

alism generating correct lighting and shadows using state of the art GPU shaders.

Several objects can be loaded ranging from simple shapes like cubes or spheres to complex models like buildings or animals. Each object has his own attributes: mass, velocity, friction and several other properties to push physic and aspect as realistic as possible. Many robot models are provided in a community-supported database, and everyone can create his own model defining a physical entity with dynamic, kinematic, and visual properties. Gazebo can also generate information from different kind of sensors: laser range finder, 2D and RGB-D cameras, contact sensors, inertial measurement units (IMUs) and radio-frequency identifications (RFIDs). The use of sensors is very important: in this way robots can to act in different ways depending on data read in the simulated world. Custom plug-ins can be developed to make the robot model interact with the world. Plug-ins provide direct control over all robot aspects and manage data collected by sensors.

Many simulation parameters can be directly controlled also by a QT-based graphical interface. Gazebo is compatible with several Linux distributions and a native interface to ROS [80] and Player [29] is provided in order to integrate different kind of robots. In this way, it is not necessary to use the API to develop a specific interface for each robot or sensor, but any device working with ROS or Player can be simulated. The supported programming language are C, C++, Java and Python.

B.1.2 VRep

The 3D robot simulator V-REP [28] has been developed to perform simulation of factory automation systems. V-REP is available in 4 licenses: Player (free), Pro Edu (free for educational), Pro Eval (not for commercial use) and Pro (commercial use) and it is Open Source for not commercial use. The first public release was in March 2010, in August 2012 started the ROS integration and at the beginning of 2013 it became Open Source. V-REP offers fast prototyping and verification, fast algorithm development, robotics related education, remote monitoring, hardware control, safety monitoring, and product presentation. V-REP can rather be seen as a hybrid simulator that combines kinematics and dynamics in order to obtain the best performance for various simulation scenarios. It is based on two physics

engine: Bullet [40] and ODE [41] (the same as Gazebo) and user is free to switch from one to the other at any time.

The integrated development environment is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution. Controllers can be written in C/C++, Python, Java, Lua, Matlab or Urbi.

V-REP is compatible with Windows, Linux and Mac OS X. Documentation is very good and cover all aspect of the simulator. A lot of tutorials and examples allow users to learn quickly how use it.

B.1.3 Comparison

In Table B.1, a selection of simulators parameters are summarized. The following features are compared:

1. Available licenses(License);
2. Operating system (OS); it describes which OS is supported by the robotic software.
3. Simulator type; it describes if the robotic software provides either a 2-D or 3-D simulation environment.
4. Programming language; it describes the language supported by the robotic software.
5. Year of origin; it specifics the year of commencement of the simulator;
6. Collision Detection;
7. Sensors; it describes the supported sensors. Only the most requested sensors are reported in the table due to space limitations.
8. Graphical User Interface; it describes if it is possible to modify objects and the environment during run-time and/or program functions in an development environment. The graphical user interface does not include windows that open to display the simulation.

9. Portability; “Yes” means that the code written for a simulation is portable to a real robotic platform.

10. Scalability “High” means that the simulator does not require a lot a resource for running complex simulation and that it can simulate more robot simultaneously, “Medium” means that the simulator in some case required a lot resource and “Low” specifics that the simulator can’t simulate multiple robot at the same time;.

11. Real Time; it specifics the number of allowed operation during the simulation. “High” means that you can do a very large number of operation in real time, “Low” means that only a few operation can do in real time.

12. Interfaces; it describes the facility of integration the simulator with another system.

13. Documentation level provided with the simulator (Documentation); it can be “High” or “Low”. “High” means the documentation only provides descriptions of the functions in the robotic software libraries; “Low” means the documentation provides the code for the functions in the robotic software libraries.

14. Tutorial; it describes if examples and a step-by-step guide are provided. “Yes” means that a well defined guide with examples is available; “limited” means a guide exists, but not enough details and examples are provided. Finally, “No” means there is not a useful tutorial and/or examples.

15. Debugging/Logging; describes if the simulator debugging, fault tolerances, play-back, and logging features are provided by the robotic software.

B.2 Comau Smart5 SiX

The robot used is a Comau Smart5 SiX (Figure B.1). It is a small manipulator with 6 DoF particularly suitable for all operations that require low payload, fast

Table B.1: Summary of the main features of Gazebo and V-Rep simulators.

	Gazebo	**V-Rep**
License	Open Source	Open Source for not commercial use
OS	Linux	Linux, Win, MacOSX
Programming Language	C++/Python/Java	C/C++/Python/Java/Matlab/ Urbi
Year of origin	2002	2010
Collision detection	Yes	Yes
Sensors	Laser Range Finders, 2D and RGB-D Cameras, Contact Sensors, IMU, RFID	Proximity, Vision, Force
GUI	Sufficient	Good
Portability	Yes	Yes
Scalability	Medium	High
Real time	Low	High
Interfaces	ROS (Excellent), Player (Excellent)	ROS (Good)
Documentation	Low	High
Tutorial	Yes	High
Debugging/ Logging	Yes	Yes

movement, and a high degree of repeatability. In Table B.2, the main characteristics of the robot are listed[1].

Figure B.1: The small manipulator Comau Smart5 SiX.

From these technical data a URDF model (Figure B.2 (a)) has been built. Information not provided by Comau, like mass and inertial matrices, has been estimated from the original 3D robot model by approximating the different parts to parallelepipeds and applying formulas in Equation B.1 and B.2:

[1] A more exhaustive description of the robot and its technical specifications can be found at `http://www.comau.com`

Table B.2: Summary of the main Comau Smart5 SiX features.

	Lock lower angle	**Lock upper angle**	**Speed**
Axis 1	-170^o	170^o	140^o/s
Axis 2	-85^o	155^o	160^o/s
Axis 3	-170^o	0^o	170^o/s
Axis 4	-210^o	210^o	450^o/s
Axis 5	-130^o	130^o	375^o/s
Axis 6	-2700^o	2700^o	550^o/s

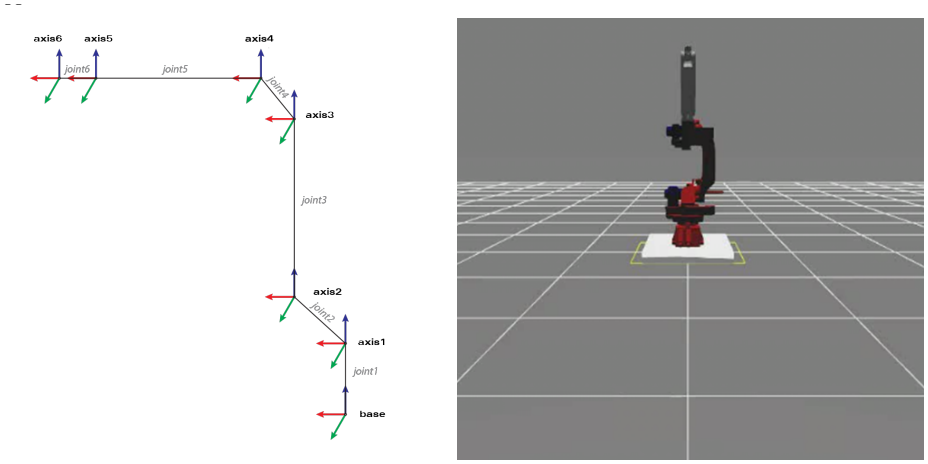
$$m = w \cdot h \cdot d \cdot SW_{steel}; \quad I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{B.1}$$

$$\begin{aligned} I_{xx} &= 1/12 \quad m(h^2 + d^2) \\ I_{yy} &= 1/12 \quad m(w^2 + d^2) \\ I_{zz} &= 1/12 \quad m(w^2 + h^2) \end{aligned} \tag{B.2}$$

where:

- m is the parallelepiped mass;
- w, h, and d are respectively the parallelepiped width, height, and depth;
- SW_{steel} is the steel specific weight approximated to $8 \cdot 10^{-3}$ Kg/m^3;
- I is the parallelepiped inertial matrix.

The final model has been integrated in both Gazebo and V-REP, but only the first one (Figure B.2 (b)) has been used as experimental environment due to the Gazebo robot plug-in can be better customized for our purposes.



(a) (b)

Figure B.2: The Comau Smart5 SiX robot represented through URDF links and joints (a), and simulated in Gazebo (b).

B.3 Vstone Robovie-X

The Robovie-X (Figure B.3) is a small bipedal robot emerged from the creative labs of KumoTek Robotics and Vstone Corporation. The robot is designed to offer an entertaining, low-cost alternative to high-end robotics, while stylishly combining the latest in sleek design and cutting-edge robotics engineering.

In this work we used the standard model,it tall 35 cm and it comes with 17 degrees of freedom (5 per leg, 3 per arm, 1 for the head) and a 60 MHz processor with 512 kB ROM and 64 kB RAM to store pre-defined movements and manage the motor control. The VS-S092J servos it features, give the Robovie-X high motion performances, making it capable of fast walking, dancing, fighting, playing soccer.

In fact, the X in the name stands for flexible because of the large range of optional accessories that allows you to expand the basic model. A standard software called RobovieMaker2 for Microsoft Windows is also supplied with the Robovie-X. This software allows you to interface with the robot by calibrating and controlling it. It also provides a simple graphical overview of the installed motion, but there is no APIs available at the moment.

For this reason we developed a Linux C++ driver for the Robovie-X able to

Figure B.3: The small humanoid Vstone Robovie-X.

communicate directly with the robot by means of a USB cable connected to the internal control board. The driver also has a ROS wrapper that integrate Robovie-X with the well-known robotics framework. A URDF model (Figure B.4)) has also been built in order to simulate the Robovie-X in a virtual environment. The technical data to compose the virtual model are extracted from an already virtualized similar robot.

B.4 Aldebaran NAO

The Aldebaran NAO H25 v 4.0 (Figure B.5) is a small humanoid robot with 25 DoF and an integrated Intel Atom CPU @ 1.6GHz. NAO comes with two gyroscopes, an accelerometer, a feedback provided from all its joints and pressure sensors on its feet. It can communicate with external systems using a Wi-Fi connection. A complete software suite and a SDK[2] package are also provided to fully program the humanoid platform and interact with the NAOqi application programming interface (API).

[2] *http://www.aldebaran-robotics.com/documentation/dev/sdk.html*

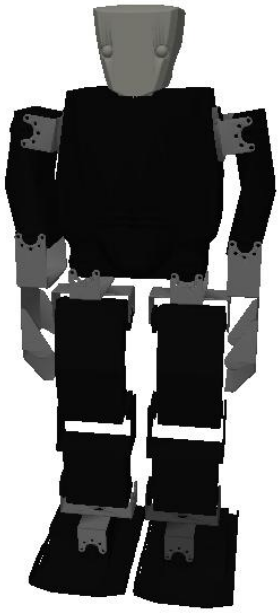


Figure B.4: The Robovie-X simulated in Gazebo.

We based our work on the documentation[3] by Aldebaran Robotics. The virtual model is not manually composed, but we auto-generate it starting from some `xacro` scripts in order to better structure our model. The `xacro` language is based on XML and it allows us to define basic macros to create more complex URDF models.

Four basic macros has been created to build the complete model:

- Joint: defines the model joints;
- Link: defines the model links;
- Visual: defines the model visual meshes;
- Collision: defines the model collision meshes.

this division is very important in order to easily update the model to new robot versions or attach NAO accessories to the main body.

All joints are defined using a single macro setting an effort limit of $100N$, a velocity limit of $5rad/s$, a damping factor of $0\frac{N \cdot m \cdot s}{rad}$ and a friction factor of $25N \cdot m$.

[3] *NAO documentation* http://www.aldebaran-robotics.com/documentation/

For each link composing NAO, we defined the inertial matrices starting from the ones declared from Aldebaran and referring them to the CoM (Center of Mass) using the *Huygens-Steiner* or *Parallel axis* Theorem. In Equation B.3 the used formula is explained

$$\Gamma_{CoM} = \Gamma_{NAO} + m\left\{\left\|t_{CoM,NAO}\right\|^2 I - t_{CoM,NAO} t_{CoM,NAO}^T\right\} \tag{B.3}$$

were:

- Γ_{CoM} is the calculated inertia matrix referred to the Center of Mass;
- Γ_{NAO} is the original inertia matrix given by Aldebaran Robotics;
- m is the mass of the considered NAO part;
- $t_{CoM,NAO}$ is relative translation between the Center of Mass and the original reference system.

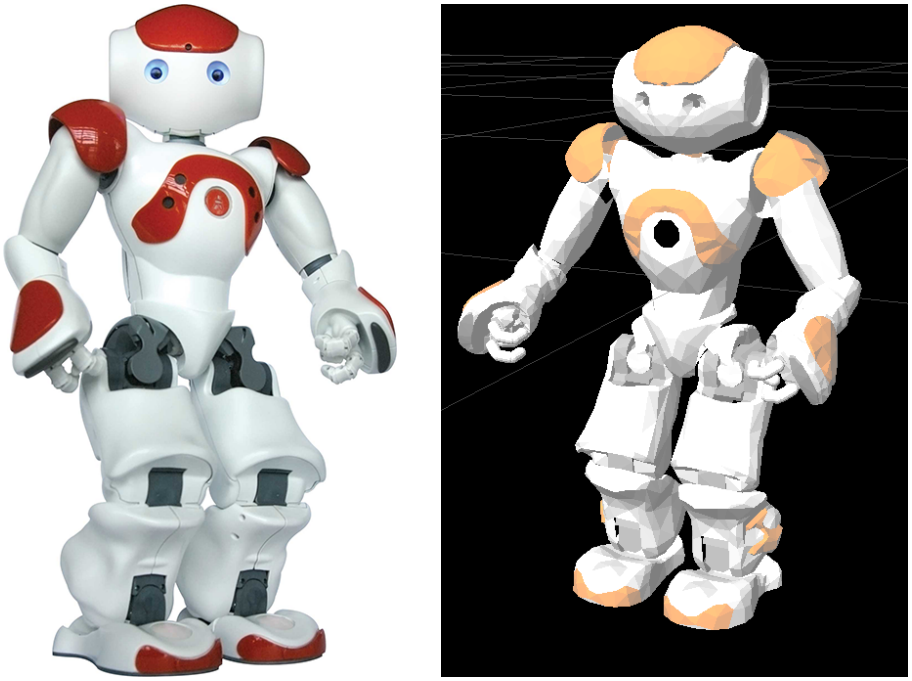
An URDF model already exists in the NAO ROS package from Freiburg University, but inertial matrices are not correctly defined, no realistic meshes has been provided and the xacro scripting only take account of visual and structure separation. The resulting model is very hard to read and modify, consequently we start to building our new model from zero. It is worth to notice that, with no proper inertial matrices, the model coming with the actual ROS package is not suitable for simulation and so no comparison with our model is possible.

Both Gazebo and V-REP are able to import URDF models, we only have to develop the specific plugin. Model and plugins will be released as a Open Source software[4]. In Figure B.5 the physical and the simulated[5] robot are shown.

In order to evaluate our model characteristics on the selected simulators, we tested two concrete, challenging robot tasks, namely walking straight and turning around. The system developed is able to generate suitable movements based on standard motion provided by Aldebaran. The same joint movements could be used

[4]ROS model and plugins for Alderan NAO are available respectively at
`https://github.com/iaslab-unipd/nao_description`,
`https://github.com/iaslab-unipd/nao_gazebo_plugin`,
`https://github.com/iaslab-unipd/nao_vrep_plugin`

[5]NAO users can download official meshes from the Aldebaran Robotics site. A script is available to properly split in parts the provided 3D mesh file.



(a) NAO physical robot. (b) NAO URDF model.

Figure B.5: Aldebaran NAO: physical robot (a) and his URDF model (b).

as input for both real and simulated robot. Thus, the tasks were also performed using a real NAO platform in order to better compare the results coming from the simulators.

The first test was based on the standard walking straight command included in Aldebaran NAO Drivers. The robot walked 3 different distances (0.5, 1, and 3 meters) at 3 different velocities (40%, 80%, and 100% of the maximum effort given by NAO motors). Every single trail was repeated 7 times. The real robot, V-REP simulator with both ODE and Bullet engines, and Gazebo simulator were tested. The robot were stopped when the distance theoretically walked reach the goal, the joint positions feedback coming from robot motors were considered to better approximate the measure. On the other hand, the algorithm did not take advantage of any feedback to refine the movements.

The space walked by simulated and real robots was analyzed with respect the expected distance in the *xy* ground plan. Figure B.6 shows the distances measured along the *x* axis, where *x* is the walking direction; while Figure B.7 describes *y* as the lateral deviation with respect the straight trajectory.

The typical trajectory followed by the robot is a curve progressively deviating from the straight direction for both simulation and reality. All the tested modalities are able to correctly perform the walk with no falls. Analyzing the collected data, it is easy to see that the real robot exceeded the goals, while simulators usually underestimated the distance.

Looking at the requested task, ODE engine, in particular Gazebo, has performed a great work. The distance walked is about 70-80% of the goal, the deviation is minimal, and the variability between the attempts and the tested velocities is low. On the other hand, the performances are quite poor compared with the real robot trajectory. The difference along the *x* axis starts from 20-25% to walk 0.5 m at 40% of the maximum effort, and reach 35-40% to walk 3.0 m at 80% of the maximum effort. The gap is even higher along the *y* axis: the deviation performed by the real robot is up 2 m, while the ODE simulations return substantially zero drift.

The results from Bullet engine are deeply different. Along *x* axes, the walked distance is 60-65% less then expected at low velocities, while it consistently grows augmenting the speed. The lateral deviation follows the same trend, and it is

aligned to the to the real robot drift, at high velocities.

It is also worth to notice that Gazebo presents a great repeatability, while V-REP shows a great variability with both ODE and Bullet engines.

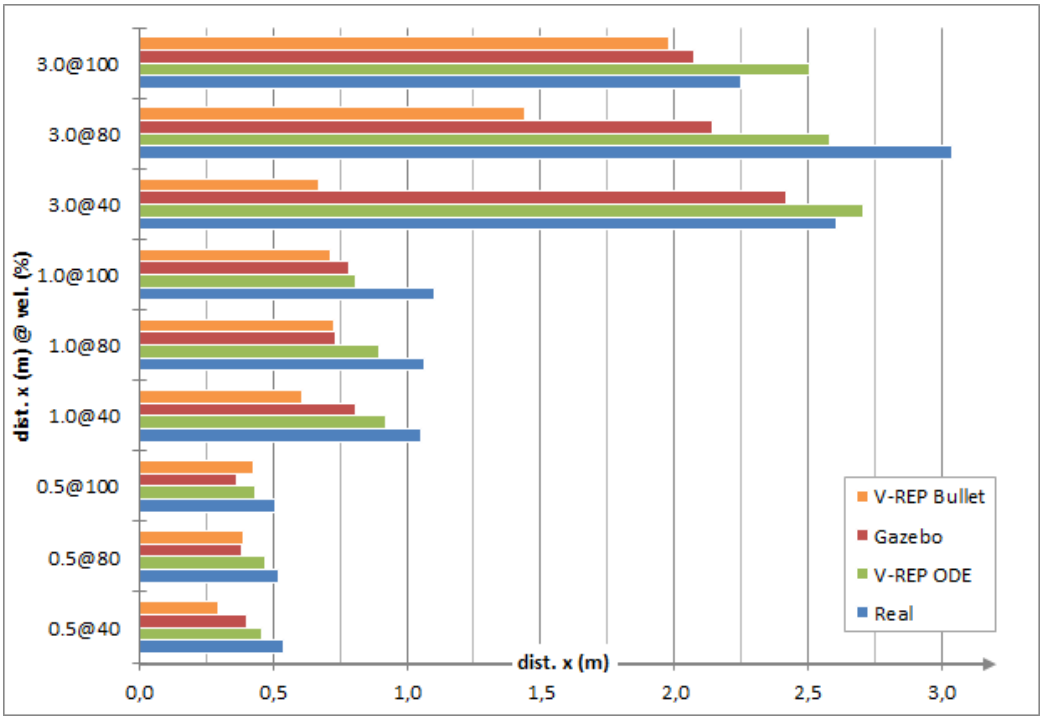


Figure B.6: Results of the task straight walk along x direction. This is not the real distance walked by the robot, but the projection along x.

The second test was based on a standard motion implemented in Aldebaran NAO Drivers to let the robot perform the turning around task. NAO turned of 4 distinct angles (90° and 360° counterclockwise, 180° and 270° clockwise) at 3 different velocities (again 40%, 80%, and 100% of the maximum effort given by NAO motors). As before, each trail were performed 7 times using the four tested modalities: real robot, V-REP (ODE and Bullet) and Gazebo. Again, the robot were stopped when the desired turning angle has been reached. The feedback coming from robot joint positions helped us in the measure approximation, but the motion did not depend from this feedback.

The angle turned by simulated and real robots was analyzed with respect the expected rotation. Figure B.8 shows the absolute value of the robot rotation

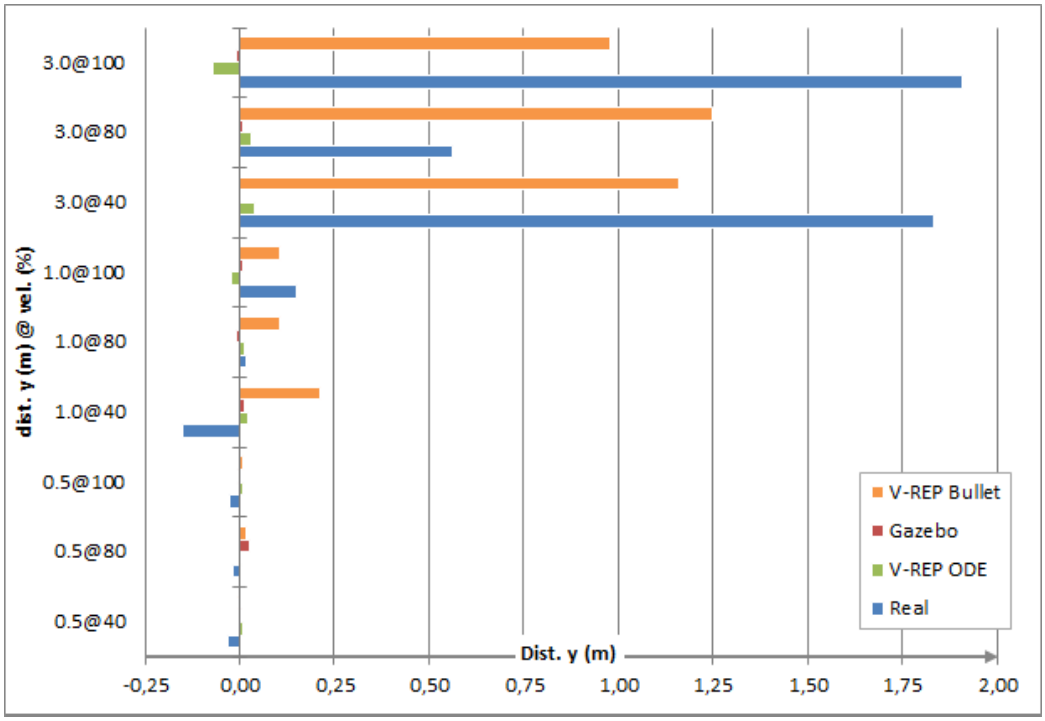


Figure B.7: Results of the task straight walk along y direction. A positive value corresponds to a right deviation of the robot, a negative value corresponds to a left deviation of the robot.

around itself.

The robot turned with no falls and a minimal deviation from its theoretical rotation center in all the considered modalities, so we did not further investigate this parameter. Again, the real robot usually overcame the goals, while simulators highly underestimated the desired rotation.

ODE engine has not performed a great work as in the first task. The angle turned is usually less than 50% with respect to the goal, on the other hand the variability between the attempts and the tested velocities is still low for both Gazebo and V-REP.

The results from Bullet engine are quite similar, the rotation performed is more effective than ODE engine and the angle is 60-65% less then expected. at low velocities, while it consistently grows augmenting the speed. The variability decreases with respect the walking task reaching the ODE engine accuracy.

Gazebo has maintain the great repeatability showed in the first task, V-REP has performed better than walking task, but a certain variability still persists.

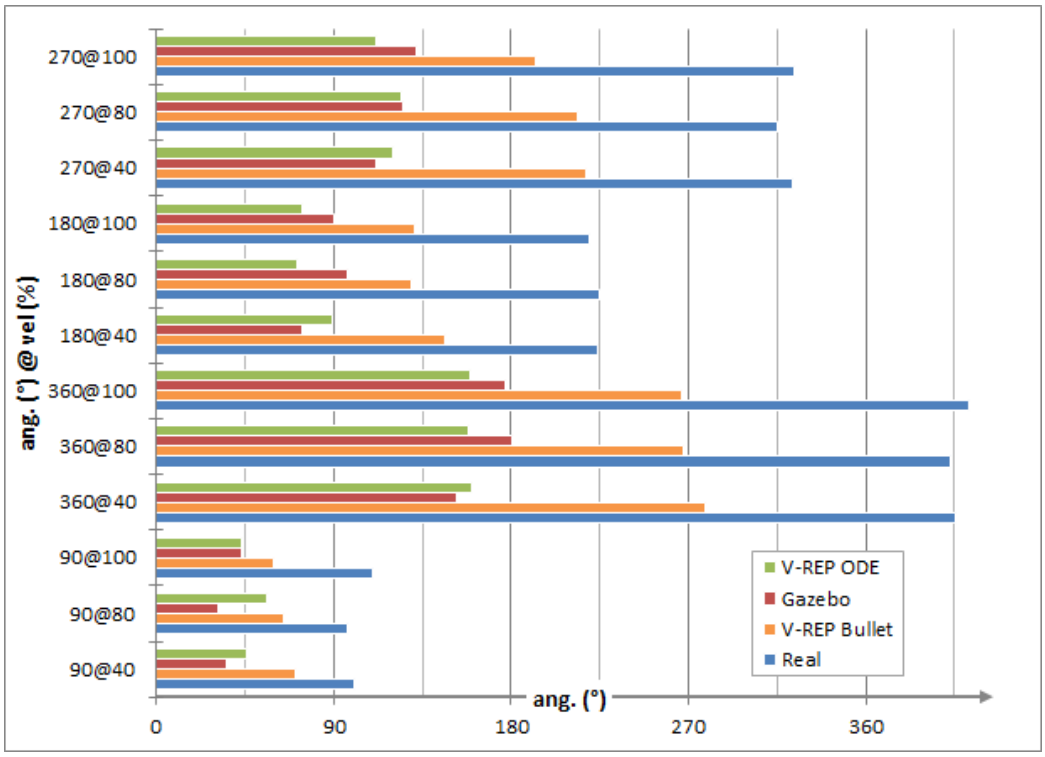


Figure B.8: Result of the task turn around. The absolute value of rotation angle is showed. At 90° and 360° the robot has turned counterclockwise, while at 180° and 270° the robot has turned clockwise.

Bibliography

[1] http://www.microsoft.com/en-us/kinectforwindows/.

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

[3] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz. Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 4(4):343–355, 2012.

[4] S. Ali and M. Shah. Human action recognition in videos using kinematic features and multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):288 –303, feb. 2010.

[5] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Correspondence mapping induced state and action metrics for robotic imitation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):299–307, 2007.

[6] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[7] G. Ballin, M. Munaro, and E. Menegatti. Human Action Recognition from RGB-D Frames Based on Real-Time 3D Optical Flow Estimation. In A. Chella, R. Pirrone, R. Sorbello, and K. R. Jóhannsdóttir, editors, *Biologically Inspired Cognitive Architectures 2012*, pages 65–74. Springer Berlin Heidelberg, 2012.

[8] F. Basso, M. Munaro, S. Michieletto, E. Pagello, and E. Menegatti. Fast and robust multi-people tracking from rgb-d data for a mobile robot. In *12th Intelligent Autonomous Systems Conference (IAS-12)*, pages 265–276. Springer, Jeju Island, Korea, June 2013.

[9] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, pages 1371–1394. Springer, Secaucus, NJ, USA, 2008.

[10] A. Bisson, A. Busatto, S. Michieletto, and E. Menegatti. Stabilize humanoid robot teleoperated by a rgb-d sensor. In *Proceedings of the Workshop Popularize Artificial Intelligence (PAI2013)*, pages 97–102, 2013.

[11] V. Bloom, D. Makris, and V. Argyriou. G3d: A gaming action dataset and real time action recognition evaluation framework. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 7 –12, june 2012.

[12] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

[13] S. Calinon and A. Billard. Stochastic gesture production and recognition model for a humanoid robot. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2769–2774. IEEE, 2004.

[14] S. Calinon and A. Billard. Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm. In *Proceedings of the 22nd international conference on Machine learning*, pages 105–112. ACM, 2005.

[15] S. Calinon and A. Billard. A probabilistic programming by demonstration framework handling constraints in joint space and task space. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 367–372. IEEE, 2008.

[16] S. Calinon, F. D'halluin, D. G. Caldwell, and A. G. Billard. Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 582–588. IEEE, 2009.

[17] 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.

[18] E. Cypher and D. C. Halbert. *Watch what I do: programming by demonstration*. The MIT Press, 1993.

[19] B. Dariush, M. Gienger, A. Arumbakkam, Y. Zhu, B. Jian, K. Fujimura, and C. Goerick. Online transfer of human motion to humanoids. *International Journal of Humanoid Robotics*, 6(02):265–289, 2009.

[20] W. C. Davidon. Variable metric method for minimization. *SIAM Journal on Optimization*, 1(1):1–17, 1991.

[21] 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 (Methodological)*, pages 1–38, 1977.

[22] R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2):109–116, 2004.

[23] G. Du, P. Zhang, J. Mai, and Z. Li. Markerless kinect-based hand tracking for robot teleoperation. *INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS*, 9, 2012.

[24] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 726 –733 vol.2, oct. 2003.

[25] R. Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970.

[26] R. Fletcher and M. J. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168, 1963.

[27] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.

[28] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira. Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. In *Proceedings of the Second international conference on Simulation, modeling, and programming for autonomous robots*, SIMPAR'10, pages 51–62, Berlin, Heidelberg, 2010. Springer-Verlag.

[29] B. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th international conference on advanced robotics*, volume 1, pages 317–323, 2003.

[30] S. Ghidoni, S. Anzalone, M. Munaro, S. Michieletto, and E. Menegatti. A distributed perception infrastructure for robot assisted living. *To appear in Robotics and Automous Systems (RAS) Journal*.

[31] M. Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42. ACM, 1998.

[32] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

[33] R. Gopalan and B. Dariush. Toward a vision based hand gesture interface for robotic grasping. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1452–1459. IEEE, 2009.

[34] D. H. Grollman and A. Billard. Donut as i do: Learning from failed demonstrations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3804–3809. IEEE, 2011.

[35] D. H. Grollman and A. G. Billard. Robot learning from failed demonstrations. *International Journal of Social Robotics*, 4(4):331–342, 2012.

[36] J. Han, L. Shao, D. Xu, and J. Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, 2013.

[37] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *Robotics, IEEE Transactions on*, 24(6):1463–1467, 2008.

[38] M. Holte and T. Moeslund. View invariant gesture recognition using 3d motion primitives. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 797 –800, 31 2008-april 4 2008.

[39] M. Holte, T. Moeslund, N. Nikolaidis, and I. Pitas. 3d human action recognition for multi-view camera systems. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*, pages 342 –349, may 2011.

[40] http://bulletphysics.org/. Bullet Physics Library. [online].

[41] http://www.ode.org/. Open Dynamics Engine. [online].

[42] http://www.ogre3d.org/. OGRE 3D. [online].

[43] http://www.primesense.com/solutions/nite middleware. Nite middleware [online].

[44] A. Kar. Skeletal tracking using microsoft kinect. *Methodology*, 1:1–11, 2010.

[45] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154 vol.3, Sept.-2 Oct.

[46] H. S. Koppula and A. Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. ICML, 2013.

[47] M. Korner and J. Denzler. Analyzing the subspaces obtained by dimensionality reduction for human action recognition from 3d data. In *Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on*, pages 130 –135, sept. 2012.

[48] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *Robotics and Automation, IEEE Transactions on*, 10(6):799–822, 1994.

[49] I. Laptev and T. Lindeberg. Space-time interest points. In *Proc. Ninth IEEE Int Computer Vision Conf*, pages 432–439, 2003.

[50] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition CVPR 2008*, pages 1–8, 2008.

[51] J. Lee, J. Chai, P. S. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 491–500. ACM, 2002.

[52] J. Lei, X. Ren, and D. Fox. Fine-grained kitchen activity recognition using rgb-d. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 208–211, New York, NY, USA, 2012. ACM.

[53] A. León, E. F. Morales, L. Altamirano, and J. R. Ruiz. Teaching a robot to perform task through imitation and on-line feedback. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 549–556. Springer, 2011.

[54] W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. In *IEEE International Workshop on CVPR for Human Communicative Behavior Analysis (in conjunction with CVPR 2010), San Francisco, CA,*, pages 9 –14, June 2010.

[55] R. Liu, S. Z. Li, X. Yuan, and R. He. Online Determination of Track Loss Using Template Inverse Matching. In *The Eighth International Workshop*

on Visual Surveillance - VS2008, Marseille, France, 2008. Graeme Jones and Tieniu Tan and Steve Maybank and Dimitrios Makris.

[56] B. Lukas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI '81*, pages 674–679, 1981.

[57] M. M. Marinho, A. A. Geraldes, A. P. Bó, and G. A. Borges. Manipulator control based on the dual quaternion framework for intuitive teleoperation using kinect. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, pages 319–324. IEEE, 2012.

[58] S. Michieletto, N. Chessa, and E. Menegatti. Learning how to approach industrial robot tasks from natural demonstrations. In *Advanced Robotics and its Social Impacts (ARSO), 2013 IEEE Workshop on*, pages 255–260. IEEE, 2013.

[59] S. Michieletto, S. Ghidoni, E. Pagello, M. Moro, and E. Menegatti. Why teach robotics using ros. *To appear in Journal of Automation, Mobile Robotics & Intelligent Sysyems (JAMRIS)*.

[60] S. Michieletto and E. Menegatti. Human action recognition oriented to humanoid robots action reproduction. In *Proceedings of the Workshop Popularize Artificial Intelligence (PAI2012)*, pages 35–40, 2012.

[61] S. Michieletto, A. Rizzi, and E. Menegatti. Robot learning by observing humans activities and modeling failures. In *IROS workshops: Cognitive Robotics Systems (CRS2013), IEEE (Nov 2013)*, 2013.

[62] S. Michieletto, D. Zanin, and E. Menegatti. In Z. X. David Al-Dabass, Alessandra Orsoni, editor, *European Modelling Symposium (EMS2013)*, pages 448–453, Manchester, UK.

[63] Y. Ming, Q. Ruan, and A. Hauptmann. Activity recognition from rgb-d camera with 3d local spatio-temporal features. In *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, pages 344 –349, july 2012.

[64] K. Miura, M. Morisawa, S. Nakaoka, F. Kanehiro, K. Harada, K. Kaneko, and S. Kajita. Robot motion remix based on motion capture data towards human-like locomotion of humanoid robots. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 596–603. IEEE, 2009.

[65] S. Muench, J. Kreuziger, M. Kaiser, and R. Dillman. Robot programming by demonstration (rpd)-using machine learning and user interaction methods for the development of easy and comfortable robot programming systems. In *Proceedings of the International Symposium on Industrial Robots*, volume 25, pages 685–685. INTERNATIONAL FEDERATION OF ROBOTICS, & ROBOTIC INDUSTRIES, 1994.

[66] M. Munaro, G. Ballin, S. Michieletto, and E. Menegatti. 3d flow estimation for human action recognition from colored point clouds. *Biologically Inspired Cognitive Architectures*, 2013.

[67] M. Munaro, F. Basso, and E. Menegatti. Tracking people withing groups with rgb-d data. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS), Vilamoura (Portugal)*, 2012.

[68] M. Munaro, F. Basso, S. Michieletto, E. Pagello, and E. Menegatti. A software architecture for rgb-d people tracking based on ros framework for a mobile robot. In *Frontiers of Intelligent Autonomous Systems*, pages 53–68. Springer, 2013.

[69] M. Munaro, S. Michieletto, and E. Menegatti. An evaluation of 3D motion flow and 3D pose estimation for human action recognition. In *RSS Workshops: RGB-D: Advanced Reasoning with Depth Cameras*, 2013.

[70] R. M. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[71] B. Ni, G. Wang, and P. Moulin. Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *Computer Vision Workshops (ICCV*

Workshops), 2011 IEEE International Conference on, pages 1147 –1153, nov. 2011.

[72] M. N. Nicolescu and M. J. Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the second international joint conference on Autonomous agents and multi-agent systems*, pages 241–248. ACM, 2003.

[73] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy. Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 8 –13, june 2012.

[74] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy. Berkeley MHAD: A comprehensive multimodal human action database. In *Proceedings of IEEE Workshop on Applications of Computer Vision (WACV)*, Jan. 2013.

[75] M. Pardowitz, S. Knoop, R. Dillmann, and R. Zollner. Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):322–332, 2007.

[76] E. Polak and G. Ribiere. Note sur la convergence de méthodes de directions conjuguées. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 3(R1):35–43, 1969.

[77] N. S. Pollard, J. K. Hodgins, M. J. Riley, and C. G. Atkeson. Adapting human motion for the control of a humanoid robot. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1390–1397. IEEE, 2002.

[78] M. Popa, A. Koc, L. Rothkrantz, C. Shan, and P. Wiggers. Kinect sensing of shopping related actions. In undefined, K. Van Laerhoven, and J. Gelissen, editors, *Constructing Ambient Intelligence: AmI 2011 Workshops*, Amsterdam, Netherlands, 11 2011.

[79] G. Pozzato, S. Michieletto, and E. Menegatti. Towards smart robots: rock-paper-scissors gaming versus human players. In *Proceedings of the Workshop Popularize Artificial Intelligence (PAI2013)*, pages 89–95, 2013.

[80] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[81] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Robotics Research*, pages 561–572. Springer, 2005.

[82] G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[83] Y. Seol, C. OSullivan, and J. Lee. Creature features: Online motion puppetry for non-human characters. In *Proceedings of the 2013 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2013.

[84] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

[85] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[86] D. C. Smith, A. Cypher, and J. Spohrer. Kidsim: programming agents without a programming language. *Communications of the ACM*, 37(7):54–67, 1994.

[87] J. Sung, C. Ponce, B. Selman, and A. Saxena. Unstructured human activity detection from rgbd images. In *ICRA*, 2012.

[88] H. L. U. Thuc, P. V. Tuan, and J.-N. Hwang. An effective 3d geometric relational feature descriptor for human action recognition. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on*, pages 1 –6, 27 2012-march 1 2012.

[89] A. Ude. Trajectory generation from noisy positions of object features for teaching robot paths. *Robotics and Autonomous Systems*, 11(2):113–127, 1993.

[90] M. Vukobratović and J. Stepanenko. On the stability of anthropomorphic systems. *Mathematical Biosciences*, 15(1):1–37, 1972.

[91] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012), Providence, Rhode Island,*, June 2012.

[92] C. Wolf, J. Mille, E. Lombardi, O. Celiktutan, M. Jiu, M. Baccouche, E. Dellandra, C.-E. Bichot, C. Garcia, and B. Sankur. The LIRIS Human activities dataset and the ICPR 2012 human activities recognition and localization competition. Technical Report RR-LIRIS-2012-004, LIRIS UMR 5205 CNRS/INSA de Lyon/Universit Claude Bernard Lyon 1/Universit Lumire Lyon 2/cole Centrale de Lyon, Mar. 2012.

[93] L. Xia, C.-C. Chen, and J. Aggarwal. View invariant human action recognition using histograms of 3d joints. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 20 –27, june 2012.

[94] Y. Yacoob and M. Black. Parameterized modeling and recognition of activities. In *Computer Vision, 1998. Sixth International Conference on*, pages 120 –127, jan 1998.

[95] K. Yamane, Y. Ariki, and J. Hodgins. Animating non-humanoid characters with human motion data. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 169–178, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[96] X. Yang and Y. Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *IEEE Workshop on CVPR for Human Activity Understanding from 3D Data*, 2012.

[97] H. Zhang and L. E. Parker. 4-dimensional local spatio-temporal features for human activity recognition. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2044 –2049, sept. 2011.

[98] Y. Zhao, Z. Liu, L. Yang, and H. Cheng. Combing rgb and depth map features for human activity recognition. In *Signal Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1 –4, dec. 2012.

[99] C. Zito, M. Kopicki, R. Stolkin, and J. L. Wyatt. Sequential re-planning for dextrous grasping under object-pose uncertainty. In *RSS 2013 Workshop: Manipulation with Uncertain Models*, 2013.