Master in Artificial Intelligence

Master of Science Thesis

Human Multi-Robot Interaction based on Gesture Recognition

Author:
Gerard Canal Camprodon

Supervisor:
Cecilio Angulo Bahón
(ESAII-UPC)

Cosupervisor:
Sergio Escalera Guerrero
(MAiA-UB)

Facultat d'Informàtica de Barcelona (FIB)
Facultat de Matemàtiques (UB)
Escola Tècnica Superior d'Enginyeria (URV)

Universitat Politècnica de Catalunya (UPC)
Universitat de Barcelona (UB)
Universitat Rovira i Virgili (URV)

Defense date: February 6, 2015

January 2015

"Human-like intelligence requires human-like interactions with the world."
– Rodney Brooks

Abstract

The emergence of robot applications and its growing availability to non-technical users implies the development of new ways of interaction between this kind of electronic devices and users. Human Robot Interaction (HRI) is a research area about the study of the dynamics involved in the interaction between humans and robots. It involves several knowledge fields such as natural language processing, computer vision, machine learning, electronics and even social sciences like psychology and human communication. HRI aims at the creation of natural interfaces between human and robots which are intuitive and easy to use without previous knowledge or training.

The main goal of this Master Thesis is the development of a gestural interface to interact with robots in a similar way as humans do, allowing the user to communicate information beyond linguistic description of the task (non-verbal communication). In order to fulfill this objective, the gesture recognition application has been implemented using the Microsoft's Kinect v2 sensor. Hence, a real-time algorithm is described to deal with two kinds of gestures which are described; the static gestures and the dynamic ones, being the latter recognized using a weighted Dynamic Time Warping method. Skeletal features are used to define both kinds of gestural sequences, having each gesture its own set of specific features.

The Kinect based gesture recognition application has been implemented in a multi-robot case. So, a NAO humanoid robot is in charge to interact with the users and respond to the visual signals they produce. Moreover, a wheeled Wifibot robot carries both the sensor and the NAO robot, easing navigation when necessary. The system is currently able to recognize two gestures, one of each kind (static and dynamic). The dynamic gesture consists in a wave movement which the user salutes the robot; meanwhile the static one is a pointing to an object gesture. When performed, the robot looks for objects near the location which has been pointed, and tries to detect which is the object that the user was referring to, asking him or her about it, if needed. When the object requested by the user is recognized, the robot goes down the wheeled platform, approaches to it and shows it to the user.

A broad set of user tests have been carried out demonstrating that the system is, indeed, a natural approach to human robot interaction, with a fast response and easy to use, showing high gesture recognition rates. Possible applications of this kind of systems to household environments are also discussed.

Resum (Catalan)

L'emergent nombre d'aplicacions relacionades amb la robòtica i la seva creixent disponibilitat per part d'usuaris no tècnics implica el desenvolupament de noves formes d'interacció entre aquests tipus de dispositius i els usuaris. La Interacció Persona Robot (IPR) és una àrea de recerca que estudia les dinàmiques involucrades en la interacció entre humans i robots, i que inclou diferents àrees de coneixement tals com el processament del llenguatge natural, la visió per computador, l'aprenentatge automàtic, l'electrònica i fins i tot ciències socials com la psicologia i la comunicació humana. L'IPR té com a objectiu la creació d'interfícies naturals entre persones i robots, que siguin intuïtives i fàcils d'usar sense cap coneixement o entrenament previs.

El principal objectiu d'aquest Treball Final de Màster és el desenvolupament d'una interfície gestual per tal d'interactuar amb robots d'una forma similar a la que empren els humans, permetent a l'usuari comunicar informació més enllà d'una descripció verbal de la tasca.

Per tal d'acomplir aquest objectiu, l'aplicació de reconeixement de gestos s'ha implementat utilitzant el sensor Microsoft Kinect v2. Per consegüent, s'ha descrit un algorisme en temps real que tracta amb dos tipus de gestos diferents: els gestos estàtics i els dinàmics, i un mètode de *Dynamic Time Warping* ponderat s'ha emprat per reconèixer els gestos dinàmics. Ambdós tipus de gestos s'han definit mitjançant característiques esquelètiques, on cada gest en té el seu propi conjunt.

L'aplicació de reconeixement de gestos basada en Kinect s'ha implementat en un cas multi-robot. Un robot *humanoide* NAO és l'encarregat d'interactuar amb els usuaris i respondre als estímuls visuals que produeixen. A més, un robot amb rodes Wifibot transporta tant el sensor com el NAO, facilitant-ne així la navegació. El sistema és capaç de reconèixer un gest de cada tipus. El gest dinàmic consisteix en una salutació amb el braç, i l'estàtic rau en senyalar a un objecte. Quan és realitzat, el robot cerca objectes a prop de la zona senyalada i intenta detectar quin era l'objecte que l'usuari indicava, preguntant-li al respecte si és cal. Un cop es reconeix quin era l'objecte referenciat, el robot baixa de la plataforma i s'hi acosta per mostrar-lo.

Diverses proves d'usuari s'han dut a terme, demostrant que el sistema és una aproximació natural a la interacció persona robot, amb una ràpida resposta i facilitat d'ús, amb altes taxes de reconeixement. Possibles aplicacions d'aquests tipus de sistemes en entorns domèstics són també comentades.

Resumen **(Spanish)**

El emergente número de aplicaciones robóticas i su creciente disponibilidad por parte de usuarios no técnicos implica el desarrollo de nuevas formas de interacción entre estos tipos de dispositivos y los usuarios. La Interacción Persona Robot (IPR) es un área de investigación relacionada con el estudio de las dinámicas involucradas en la interacción entre personas y robots. Incluye áreas de conocimiento diversas tales cómo el procesamiento del lenguaje natural, la visión por computador, el aprendizaje automático, la electrónica e incluso ciencias sociales como la psicología o la comunicación humana. La IPR pretende crear interfaces naturales entre robots i personas que sean intuitivas y fáciles de usar sin previo conocimiento o entrenamiento.

El principal objetivo de este Trabajo Final de Máster es el desarrollo de una interfaz gestual para interactuar con robots de una forma similar a la que utilizan los humanos, permitiendo al usuario comunicar información más allá de las descripciones verbales.

Para cumplir este objetivo, un sensor Microsoft Kinect v2 se ha utilizado para implementar la aplicación de reconocimiento de gestos. Por consiguiente, se describe un algoritmo en tiempo real para tratar dos tipos de gestos: los estáticos y los dinámicos, y un método de *Dynamic Time Warping* ponderado se utiliza para reconocer los gestos dinámicos. Ambos tipos de gestos se han definido por medio de características esqueléticas, de las cuales cada gesto tiene su propio conjunto.

La aplicación de reconocimiento de gestos basada en Kinect se ha implementado en un caso multi-robot. Un robot humanoide NAO se encarga de interactuar con los usuarios y responder a los estímulos visuales que producen. Además, un robot con ruedas Wifibot transporta tanto el sensor como el NAO, facilitando así su navegación. El sistema es capaz de reconocer un gesto de cada tipo. El gesto dinámico consiste en un saludo con el brazo, mientras que el estático radica en señalar un objeto. Cuando éste es realizado, el robot busca objetos en la zona apuntada e intenta detectar a cuál se refería el usuario, preguntándole si hace falta. Cuando el objeto es reconocido, el robot baja de la plataforma, se acerca a él y lo muestra al usuario.

Varias pruebas de usuario se han llevado a cabo, demostrando que el sistema es una aproximación natural a la interacción persona robot, con una rápida respuesta y fácil uso, con altas tasas de reconocimiento. Posibles aplicaciones de estos tipos de sistemas en entornos domésticos son también comentadas.

Acknowledgements

I would like to thank my supervisors, Dr. Cecilio Angulo and Dr. Sergio Escalera, for his enormous support, perfect guidance, strong encouragement, and for being always there since the beginning. I must also thank my father, Josep Maria, for his great help in the handicraft part of the project and to Joan Guasch for the initial ideas, as well as to Dr. Marta Díaz for his guidelines in the design of the user tests.

A special thanks to all my family, friends and the people from the HuPBA group, who have given me an unconditional support throughout all this time.

Finally, I want to acknowledge all the volunteers who spent some time to participate in the user tests, and who gave me a precious feedback about the system.

Contents

List of Figures

List of Algorithms

List of Tables

1 Introduction

Robots are here and they have come to stay [14]. Research in robotic systems began many years ago, and there is a long way to go. Scientists have made robots able to navigate in different kinds of environments, walk, talk and also understand spoken language. Sensing capabilities have been also included in order to achieve the previous tasks and sense the environment that surrounds them to detect objects, persons and obstacles. Moreover, reasoning methodologies have also been applied, allowing robots to design plans to achieve their objectives. We have even created robots to human resemblance, so they are called humanoid robots.

Now that robots are able to complete thousands of interesting tasks for us, it is time to make them understand humans as well as communicate with. This is the main aim of Human Robot Interaction (HRI), to design interfaces and situtations to better operate and interact with robots. Speech and textual interfaces are widely used in this field, but as many psychologists claim, approximately more than the 60% of human communication is performed through non-verbal cues [7, 6]. So, humans tend to interact with themselves via gestures as an important element of communication. We usually wave to our acquaintances, or point at an object to refer to them instead of describing all the scene. In many cases, gestures are more efficient to be performed and be understood, hence it is really interesting to include these abilities into robotics systems, and specially humanoid robots. Many research groups are currently working in gestural interfaces for robots. A major objective is that they should be natural and intuitive for people. Hence, they can be used with minor training, just as they would do with another human being. Moreover, such gesture recognition and understanding skills must be run in real time, as large processing time is resulting in users frustration as they do not know what is really happening.

This work introduces a gesture based Human Robot Interaction system which allows the user to communicate with a robot using nonverbal cues. Those can be a dynamic movement such as a wave with the arm or a static position to, for instance, point at an object on the scene. Two robots are involved in the system, who cooperate in order to fulfill a visual order from the user, such as pointing to an object in the ground to make the robot fetch it. Then, the robotic system's response to this visual stimulus would be to drive together to the pointed location to see clearly which object was referred, asking to the user if there is not a clear decision. Once disambiguated, one robot separates

from the other in order to finish the task of getting the object.

The dissertation begins with a review of related work in Chapter 2. Hardware and software tools used in the system are explained in Chapter 3. The implemented methods are detailed in Chapter 4. Next, Chapter 5 discusses the results of the developed algorithms in off-line experiments, as well as when considering a set of user tests. Finally, Chapter 6 concludes the work and gives some insights on possible improvements and extensions of the proposed system.

1.1 Motivation

Imagine the case of an elder or a person with mobility difficulties in his home, sitting on the sofa, when the remote controller falls to the ground and picking it up could suppose a big effort to the person, or might even be impossible to do by themselves. But, a robot could be there in order to help them and pick the object from the ground to reach it to them, as depicted in Figure 1.1. Nevertheless, specifying which object is the one to pick in a verbal channel may be hard, specially if there is more than one object near. So, an ideal way of communicating the object to the robot could be pointing at it, just as we would to to tell the same information to another human. Solving this problem of making robots able to understand people using non-verbal cues is the main motivation of this Master Thesis.

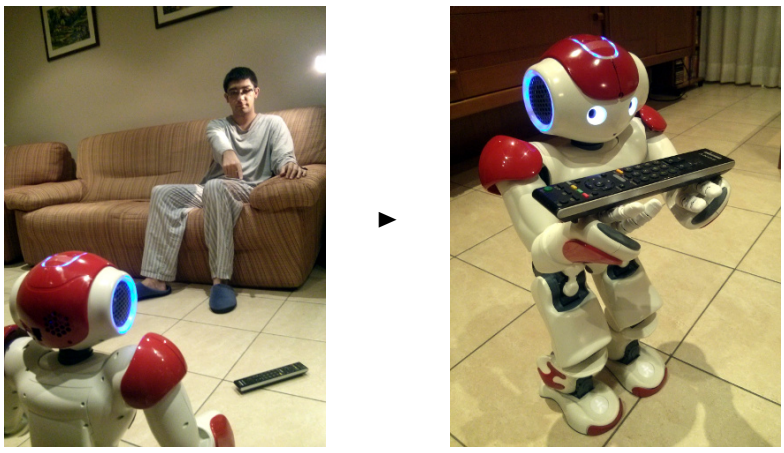


Figure 1.1: Ideal case of gesture interaction in a household environment.

To do so, a middle size humanoid robot (Aldebaran's NAO) is used as the principal robotic communicative agent. However, as the Kinect 2 sensor which is used to perform the gesture recognition is too big to be worn by the NAO, a problem emerged in order to move the sensor with the robot. This issue was solved by taking inspiration of the DARPA robotics challenge[1], which includes an experiment in which a robot must drive a car towards some goal; then exit it to finish its task by foot. In a similar way, a wheeled robot was added to the system in order to carry the sensor as well as the little humanoid, which has also to exit it in order to finish his task walking.

Furthermore, our wheeled robot is autonomous, which involves some cooperation and collaboration between robots to achieve the common goal. Instead of creating a single robotics system which can be in charge of everything, joining different smaller systems which are specialized in fulfilling a single task can make it easier and improve its efficiency and achievement rate.

1.2 Goals

The goals that have been fixed to be accomplished in this project are to build a robotics system which is *easy* to use for a common human being. The user would interact with the system using gestures, which should be *natural* for them to perform, and *intuitive*. Moreover, a *fast* response of the robot is needed in order to give feedback to the user and notify him that the gesture has been understood, so it must be a *real time* system.

In order to achieve the previous objectives, some sub goals need to be also accomplished. The Kinect based gesture recognition algorithm must be able to deal with both *static* and *dynamic gestures* and the user pointing location has to be *accurately estimated*.

Moreover, both robots have to *work together* in order to approach a given location, and also to *disambiguate* the pointed object in case of doubt. Finally, the NAO has to clearly show which was the referred object.

Several topics have been considered to complete these goals, including: object segmentation, 3D scene analysis, human detection and behaviour analysis, and robotics.

[1] theroboticschallenge.org

2 State of the art

Several contributions related to this Master Thesis can be found in the literature. Human Robot Interaction (HRI) is an active research field from many different points of view: from making humans understand the robot states through verbal and non verbal communication to doing it the other way around, making the robot understand humans. As this work is focused on interaction based on gestures, this section will present a review of the available work in this field.

To begin with, some general gesture recognition methods will be briefly commented. In [19], an algorithm for semantic motion retrieval in large data sets is proposed. To achieve it, gesture classification is performed by a method called Bag-Of-Motion-Features, which extracts wavelet spatio-temporal features and expresses them in frequency domain, transforming them into a Bag-Of-Words representation for efficient computation. A real time gesture recognition method using artificial neural networks is presented in [26]. The recognition is performed in both hands, using a hand independent representation which is obtained from salient motion features extracted from depth data. The gestures are represented as a sequence of such motion patterns. Then, Self Organizing Maps (SOM) are used to cluster the motion data in an unsupervised manner. Experiments on HRI data to operate a robot with gestures showed good performance with high recognition rates. Dynamic Time Warping approaches, as the one used in this work, are also widely used in gesture recognition. A gesture recognition method developed in [5] is applied on data coming from accelerometers and gyroscopes in real time; [16] applies the method to RGB and depth data using a probability approach, and [33, 2] apply DTW in weighted skeletal features obtained from a depth sensor.

Many publications focus on HRI applications of gesture recognition. One of the first works in this topic can be found in [45], in which both, person and arm tracking in color images was performed. Two recognition methods were compared, one template-based approach and an artificial neural network, both combined with a Viterbi algorithm. An approach to moving gesture's recognition is presented in [18], where a Kinect sensor is used to recognize gestures while the robot is moving. The method tracks the face of the person in order to perform background subtraction and then joint positions are estimated by means of a Voronoi diagram. A generated motion context is used to train a Multi-Layer Perceptron (MLP) classifier in order to recognize

similar gestures to the ones proposed here. A low cost RGB-D sensor is used in [32] to perform dynamic gesture recognition by skeleton tracking. The recognition method uses a Finite State Machine which encode the temporal signature of the gesture. An adaptive method was developed in [15] for identifying the person which is performing the gestures. The goal was to learn from gestures and therefore adapt the system to the specific person, being the same gesture performed by two different persons understood in different ways, even having the opposite meaning. Another Kinect application to gesture recognition with Hidden Markov Models (HMMs) and skeletal data is presented in [11], in which the user performs gestures to control the robot and it responds with voice or a message in the display. Deep neural networks have also been used to recognize gestures, as done in [3], aiming to recognize gestures in real time with minimal preprocessing in RGB images. They show high classification rates working online, the application being a robot that gives speech feedback. User defined gestures can be added in a semi supervised way to the system from [4], which contributes a non-parametric stochastic segmentation algorithm, the Change Point Model. This procedure does not need to be supplied with the gesture's starting and ending points, making the user able to create its own gestures to control a robot and thus being highly customizable without the need of explicit user learning or adaptation.

Elderly assistance is another interesting field in which service robotics is applied, and gesture interaction may be really useful in such case. A Kinect based approach to recognize calling gestures is proposed in [47]. This approach use a skeleton based recognition system to detect when the user is standing up, and an octree one when the skeleton is not properly tracked. Erroneous skeletons are filtered by face detection in order to determine whether the data is actually a person or a false positive. An application to object handling to the user is implemented and tested with different elder users.

Besides, some contributions are only concerned with hand gestures. The hand gesture recognition system introduced in [40] performs gesture classification in each arm independently, using two artificial neural networks, as well as HMMs, to perform arm tracking. Their trajectories are used as the input to the classifier. Another hand gesture decomposition application to HRI is proposed in [43], in which a color segmentation algorithm is used to find skin regions and a cascade of Adaboost classifiers is used for the hand posture. The method was validated in a museum robot guide. An RGB-D camera is used in [46] for hand gesture recognition. Human segmentation is performed by background subtraction and hand tracking is then calculated

from both color and depth information. Some static gestures are employed to indicate the start and end of the gesture to the system, such as opening and closing the hand. The trajectory followed in the meantime is then used to recognize the gesture by applying an HMM, as it was similarly done in [17]. A tour-guide robot able to understand gestures and speech feedback is introduced in [1], which tracks the user using depth information and performs the recognition with a Finite State Machine gesture modeling. Hand tracking approaches such as the one in [30] as well as the related work being developed in Microsoft Research with a Kinect v2 sensor [38], which shows impressive results, may imply a great improvement in the recognition of hand gestures, with applications to sign language recognition – which is another application of gesture recognition systems –.

Cooperation tasks is another research topic, as in the case of the current work, when the user cooperates with the robot to achieve a given task, for instance approaching the desired object. The system proposed in [13] detects a person with a color camera to recognize the face and laser range finders to find his legs. Then, the person perform gestures to make the robot guide him or to carry a load together. The method uses invariant Hu moments to describe the gestures and a probability based approach is used to classify them. Conversely, there are publications like [34] which make the study in the reverse way: how can robots make humans cooperate with them when the robots are those executing a gesture. It is also related to this work, as our robot performs some gestures that the human has to interpret, and posterior gestures of the human could be a result of the robots one. A similar study is conducted in [24], in which they evaluate the effect of the robot utterances when they are accompanied by gestures such as the robot looking to the person when he speaks or pointing in the direction of an object.

Spatial and directional gestures are widely used by humans to refer to elements that surround them. For instance, this work implements a 'pointing at' gesture to refer to an object on the ground. Such gestures, also known as deictic gestures, have been broadly studied. Pointing gesture recognition and direction estimation is performed in [27] by means of a cascade of HMMs and a particle filter to recognize the gesture in stereo images to which hand and face tracking is applied to capture the pointing direction. A similar HMM approach is used in [23] to recognize pointing gestures. A ROS-based robot is used in [44] to detect pointing gestures by means of a Haarlet-based hand gesture recognition system, extract the pointing direction and translate it to movement goals in a map. A tracking system is presented in [20] which recognizes the pointing gesture so that a person can tell the robot where is

another person who wants to interact with it. Finger segmentation is performed to compute the angle to which the robot has to turn its head. A research about how people refer to objects in the world is carried out in [21]. This deictic interaction comes from both speech and gesture channels. Spatial information from objects is extracted in form of features such as distance to the hand or its direction relative to the object. A K-SVD algorithm is trained to perform the classification. Human Augmented Mapping is studied in Elin Anna Topp's Ph.D. Thesis [42], in which human concepts are integrated to the robot map. Such concepts are obtained from user inputs such as pointing to a place or showing an object. The pointed location on a wall is obtained in the system of [31], which uses geometry analysis to identify shoulders and elbows to understand gestures and obtain the direction. Some constraints in the study include high illumination environments and user wearing half sleeves to better segment him.

Pointing gestures are used to refer to objects in [8] using a time-of-flight camera to get depth information. They use the line between the person's eyes and their hand as the pointing direction. Knowledge about possible object locations is exploited in [28] in order to discern between which object might be pointed, using the Dempster–Shafer theory of evidence to join information from the head pose and the pointing hand's orientation.

But deictic gestures can also be applied the other way around, as when our NAO robot points to an object it has found. Studies on making robots refer to objects via gestures have also been performed, such as the one presented in [41], in which realistic, cluttered environments containing many visually salient targets are considered. A similar study is carried out in [37]: six deictic gestures are implemented in a NAO robot to evaluate their communicative effectiveness. Involved gestures include pointing, touching and exhibiting objects.

Moreover, gesture based interaction has also been used in multi-robot environments. A human-swarm interaction scenario based on hand gestures is considered in [22]. A novel incremental machine learning approach is also proposed. This method allows the robot swarm learn and recognize the gestures in a distributed and decentralized manner.

Furthermore, there also exist applications in which the NAO robot drives. Naocar, shown in Figure 2.1, was first made by students of the *École Informatique Epitech* and then sold by RobotsLab[1], an educational robots provider

[1] robotslab.com

based in San Francisco. In it, the robot is the one who drives the vehicle rather than the vehicle working on its own, as in the case of the present work.

Figure 2.1: Naocar, NAO robot driving a BMW Z4 car from RobotsLab.

3 Resources

Several resources have been used for this work, both hardware and software. The hardware ones include the robots, two laptops and a RGB-depth sensor. The software ones are frameworks, libraries and utilities which ease the programming of the hardware components.

3.1 Microsoft's Kinect v2

In order to perform the gesture recognition, a RGB-depth sensor has been used. The chosen one is the version two of the Kinect sensor, which is manufactured and sold by Microsoft. The Kinect is a widely known sensor, and was introduced in 2010 for the Xbox 360 game console, being available to Windows users in 2012. Since then, several applications have been implemented with it, many of them in the robotics area.

The second version of the sensor was released in late 2013 and was included with the Xbox One console as a single bundle. A public Windows beta version was released in July 2014, being the final public release in late September 2014. It includes an infrared sensor, a depth one and a high definition RGB camera along with a microphone array. The sensor is shown in Figure 3.1.

Figure 3.1: Microsoft Kinect v2 sensor for Windows.

Furthermore, it incorporates several improvements with respect to the previous version. First of all, it has better resolution for both the color (1080p) and the depth cameras, and can obtain depth information from up to eight metres, while the previous one only reached about four and a half metres. It also has a wider field of view (60 degrees vertical and 70 horizontal) and works over USB 3.0, which implies a better bandwidth to transmit (extra)

data. But not only the hardware's power and precision were improved, also the software and its SDK. For instance, the new version is able to track up to six people at the same time, while the old version could track only two of them. Also, the new SDK gives skeletal information with twenty-five joints of each person, while the old sensor only provided twenty of them. Figure 3.2 displays the body joints that the Kinect 2 provides. Face analysis capabilities have been improved in this second version too, enabling the application to create a mesh of more than one thousand points for a more accurate representation of a person's face. Given that the sensor requires a powerful computer and the SDK works only in Windows 8 and 8.1, an extra laptop was needed to process the incoming data. C++ is the programming language which has been used with the SDK.

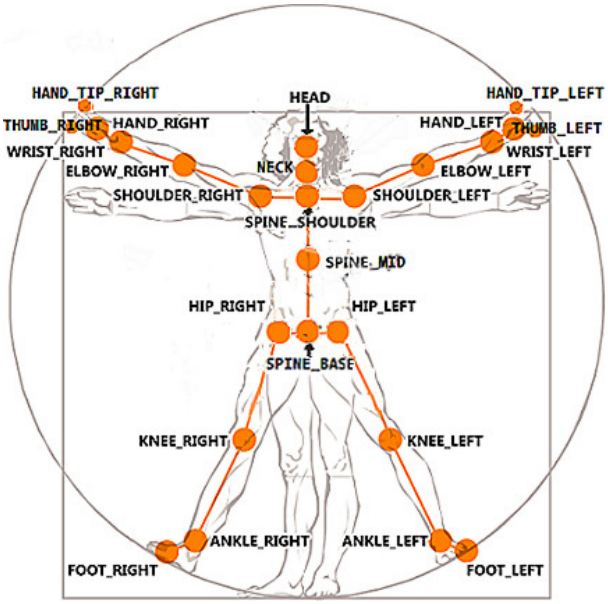


Figure 3.2: Skeleton joint positions relative to the human body in the Kinect 2. Extracted from the Kinect for Windows SDK 2.0 documentation.

3.2 Robots

The robot platforms perform the most important part of this work, and are the main component to perform the interaction with the user. Two robots have been employed: a humanoid robot, Aldebaran's NAO, and a wheeled

platform which helps the movement of the NAO and brings the sensors and part of the computing power, the Wifibot platform.

3.2.1 NAO

NAO is a humanoid robot created and developed in the French company Aldebaran Robotics[1]. Its development began in 2005, and it later replaced Sony's Aibo dog robot as the official robotic platform for the Robocup[2] Soccer League in 2010. Its cute and human-like appearance makes the NAO a very friendly robot and, thus, a great candidate for a Human Robot Interaction task.

The version of the robot which is used in this work is a v3.2, the red NAO from the UPC ESAII department, which is named Naomi. A size diagram and a picture of the robot are shown in Figure 3.3. This NAO version was released in 2009 and was the second generation in the robot evolution. Even though there were no impediments to perform its work, the robot's age and limited computing capabilities arouse some issues which needed to be handled.

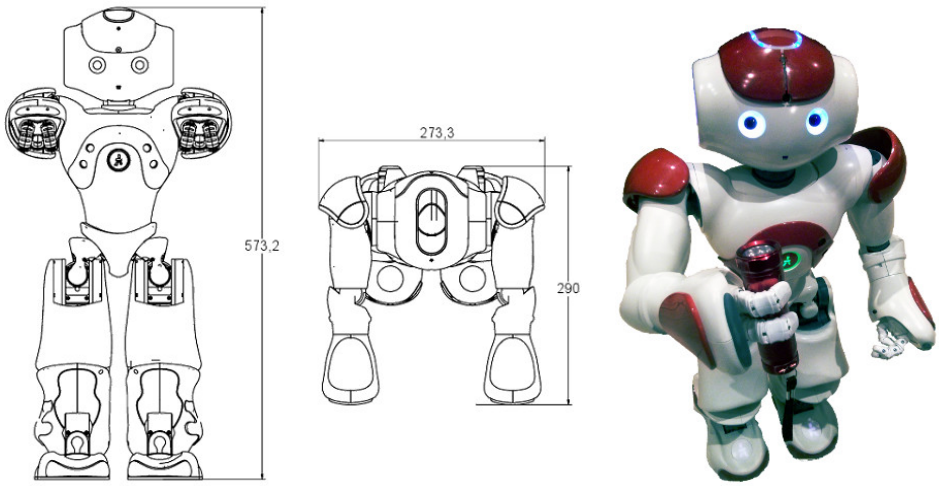


Figure 3.3: NAO robot dimensions, extracted from Aldebaran's documentation (left). ESAII department's robot Naomi used in this work (right).

[1] aldebaran.com
[2] International robot competition with many different league modalities, robocup.org.

The robot includes a complete set of useful hardware elements: networking capabilities such as WiFi and Ethernet, speakers, LEDs in the eyes and ears (which increment its expressiveness), infrared emitters, sonars, tactile sensors, two cameras, force sensing resistors, gyroscopes, accelerometers and a 45 minutes life battery. It has an AMD Geode processor at 550 MHz and runs the NaoQi operating system (a Linux based one). A hardware diagram of the robot is shown in Figure 3.4.

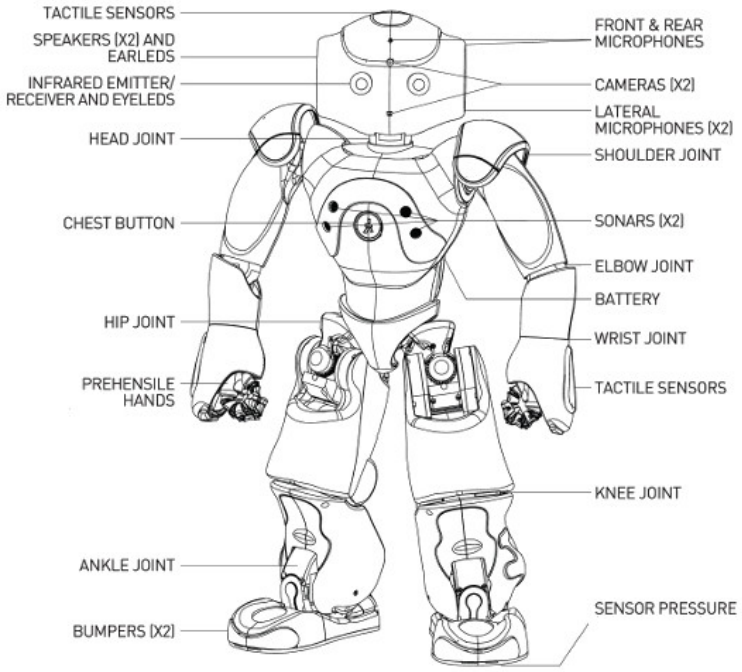


Figure 3.4: NAO robot hardware diagram. Extracted from Aldebaran's NAO Software 1.14.5 documentation.

The software included with the robot is the NaoQi API and the Choregraphe framework, which allow to program the robot easily. The NaoQi software includes basic utilities such as joint control, a walking algorithm, speech capabilities (text-to-speech and speech recognition), along with some computer vision algorithms to perform face tracking.

3.2.2 Wifibot

The platform selected to carry the Kinect sensor and the NAO was a Wifibot robot, more concretely the Wifibot lab v3, from Nexter Robotics[3]. It consists on a wheeled platform with an integrated WiFi access point and an on board computer with an Intel Atom processor. It has currently an Ubuntu 12.04 Operating System installed. Its four 12 volt wheel motors make it suitable to support all the extra weight. It also has two sonar sensors available, even though they are not used in the present work. Figure 3.5 shows the used Wifibot platform.

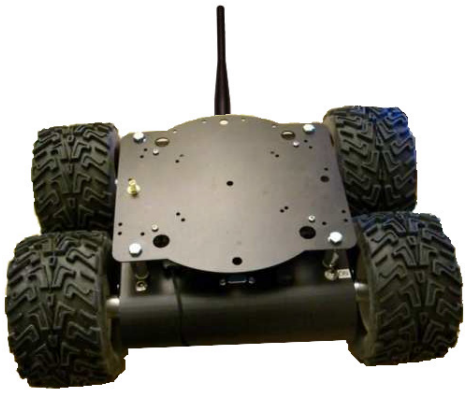


Figure 3.5: Wifibot lab v3.

Some modifications were needed in order to attach the Kinect sensor to it and carry the NAO, without losing the visual field of the camera. First of all, some strategies were developed in order to carry the NAO and making it able to go down of the platform. The final design had the robot sat on the platform itself, with its legs hanging. Some hand supports were added to avoid it fall down forward, and the sitting area was covered by a rubber sheet in order to increase adherence. Then, an elevated surface was added on the back part and attached with two bars, having the Kinect on top and over NAO's head. This construction allowed some extra space to place the Kinect's laptop. Finally, to ensure the safety of the robot and the laptop, the bars were wrapped with plastic foam, avoiding any possible bump. In Figure 3.6 a picture of the final Wifibot version is shown along with another one with the NAO on top of it.

[3] wifibot.com

Figure 3.6: Wifibot with the mounted support for the Kinect, the laptop and NAO's hand supports (left). NAO seated on the Wifibot (right).

3.3 ROS: Robot Operating System

The libraries, tools and utilities from the Robot Operating System (ROS) framework for robot software development have been used to program the robots described above. This framework was born in 2007 at Willow Garage[4], and it is currently maintained by the Open Source Robotics Foundation (OSRF)[5].

ROS provides operating system like tools, facilitating the execution of different applications in a distributed way. It allows the developer to focus on the high level programming rather than on how to interconnect all the systems, by making such interconnection between processes very easy. The framework creates the processes as a graph, in which each of them is a *node*. Furthermore, ROS utilities are implemented in different languages such as C++ and Python, allowing the developer to implement the code in the language that suits him/her better, without paying attention to the language

[4] willowgarage.com
[5] osrfoundation.org

in which the other software utilities she/he uses are written.

In order to make the nodes communicate, a *Master* service much be running so that each process can register to it and send information to other nodes through this service. Such communication is performed via message passing, being that messages objects defined in terms of basic programming types in a file with extension `.msg`, which is then compiled to create classes for different programming languages, allowing in this way the multi-language characteristic of the framework.

There are several ways in which nodes can communicate. For instance, one node can publish a message to a *topic*, and some other ones read from it, being that topic similar to a Linux pipe.

Another method of communication are *services*. In this case, a node provides a method which can be called by other nodes, like a Remote Procedure Call (RPC). Service petitions are defined in a similar way as the topic messages, but they specify the petition and the response.

Finally, there is a special kind of services which are called *actions*. Their main difference with the services is that a service executes its entire method and returns an answer, while an action sends feedback messages about how is the process going, and they can be stopped. Usually, actions are used for long procedures while services execute simple methods. Notice that there is no need to have all the process in the same machine, being it possible to have a node in one computer calling to a service or writing to a topic which is located in another computer. In fact, this is used for the communication of the main laptop with the Wifibot. Figure 3.7 shows the connection possibilities for ROS nodes.

The Indigo Igloo version of ROS is the one which has been used in this work.

3.3.1 SMACH

SMACH is a powerful library to write Finite State Machines (FSMs) in Python. It is included in the ROS distribution and provides specific utilities to interact with the ROS ecosystem. It has been used in order to develop the behaviour of the robots during their interaction task.

It allows to write hierarchical FSMs. Hence, it is possible to include some machine which was developed to make a robot do a simple task inside of a

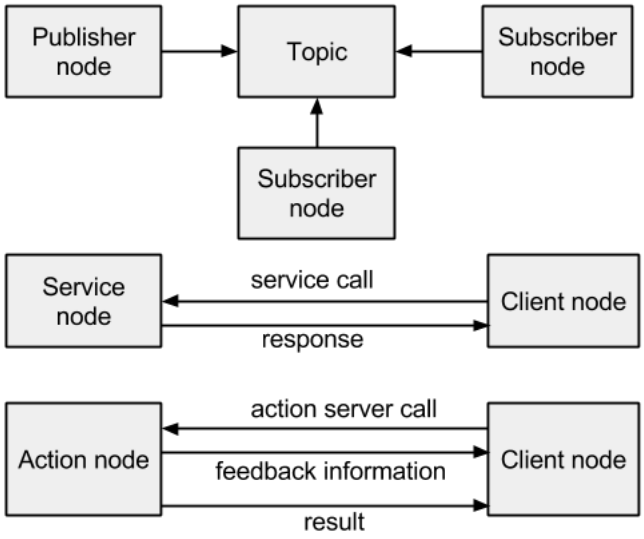


Figure 3.7: Diagram of ROS node types and communications.

more complex behaviour. This allows to completely reuse already created states, simplifying the overall work.

SMACH allows the user to define several elements of the states and finite state machines in order to control the flow of the application. For instance, each state must define the possible outcomes it has, and to which state should it transition for each of the defined outcomes. Data sharing to communicate sates is also possible.

Moreover, the library also includes some tools to visualize the state machines along with the data and execution flows inside them.

3.4 PCL: Point Cloud Library

The Point Cloud Library (PCL)[6] is an open source library whose development started in 2010 at Willow Garage. It is written in a modular way in the C++ language, and includes algorithms to process depth information organized in Point Clouds. Among several applications which involves geometric 3D processing, it is widely used in robotics and computer vision for

[6] pointclouds.org

tasks such as segmentation, feature estimation, model fitting or surface reconstruction. It also provides tools to visualize the 3D point clouds in an interactive way.

The version used in this work is the 1.8.1 one, compiled from source in Windows 8.1. It is used for tasks such as object detection and ground plane segmentation.

4 HRI System

This chapter is devoted to describe in detail the Human Robot Interaction system designed and implemented for this Master Thesis[1]. It will begin with a general overview of the whole system, then going deeper into the computer vision part and it will finally focus on the mobile robotics platforms.

4.1 Overview and architecture

The implemented system gathers all the resources as explained in Chapter 3 Resources and connects them to conform the complex application at hands. A graphical representation of the whole system is shown in Figure 4.1.

The Wifibot robotic platform has its own on board computer, which runs Ubuntu 12.04 with ROS Fuerte installed. This on board computer is only managing drivers for motors and capturing information from infra-red sensors. This robot has been endowed with a Kinect camera attached on top. Since drivers for this device are currently only working on Windows platforms, the Wifibot is also carrying a laptop running Windows 8.1 where the Kinect sensor is connected to obtain and process its data (it performs the gesture recognition and 3D cluster segmentation[2]).

The NAO robot, running an OpenNAO operating system, is connected to an external laptop using the NAO packages for ROS, which act as a wrapper to the Aldebaran's NAOqi API. This laptop acts a server connecting both robotics platfroms. It runs Ubuntu 14.04 and contains the installation of the ROS Indigo robotic meta-operating system, controlling both robots. Moreover, the secondary laptop managing the Kinect camera is also connected to the main one via the WiFi network using the rosserial_windows package.

4.1.1 The Human Robot Interaction procedure

The user interacts with the system by using gestures. For the experiment, two gestures have been considered: the "Salute" or "Wave" gesture and a "Point At" one. When the salute is performed, the robot is expected to wave

[1] The implemented code is publicly available and can be found in the Github's HR2I repository (github.com/gerardcanal/HR2I).

[2] See Chapter 4.2 Computer Vision for details about these methods.

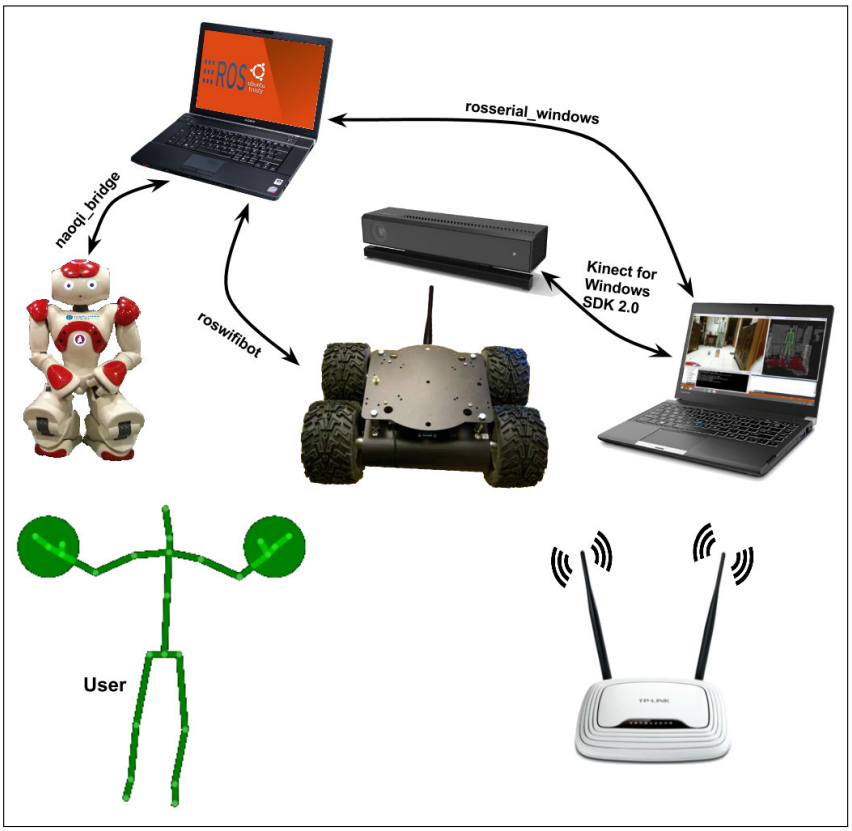


Figure 4.1: System architecture.

back at the user. In case the gesture performed is a point at some object, the robot should navigate to the object location – provided that the user has pointed at an object and neither to an empty space nor a non-ground position – and make clear that it has recognized which object was the user selecting. In case of ambiguity about the selected object, the robot will ask the user some questions related to the size or position of the detected objects, so the user can clarify, using spoken language, which object was the desired one.

Given that the defined setup and methodology correspond to a proof of concept, only a few gestures have been integrated into the system. Even though, it has been implemented in a way in which it is really easy to extend it to handle a wider set of gestures and more varied ones. Figure 4.2 shows the flow of execution in an example case.

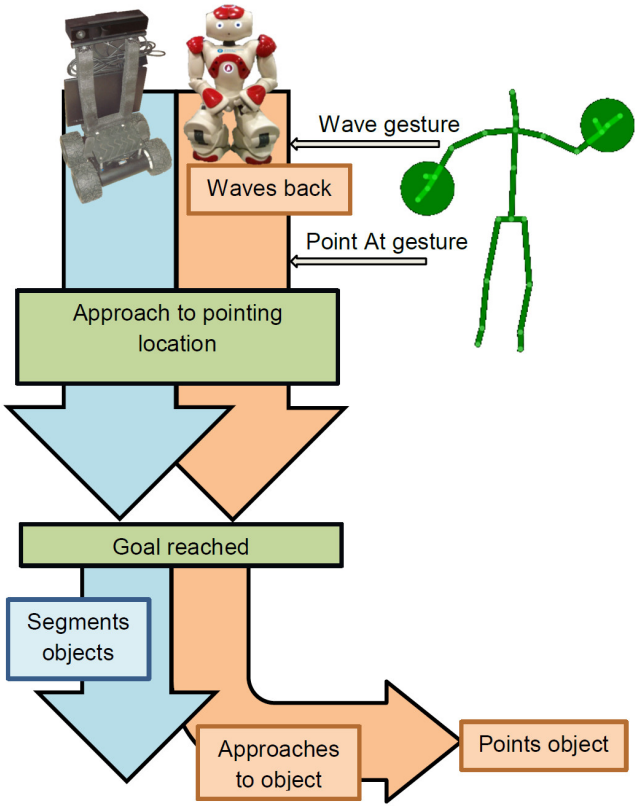


Figure 4.2: Example case of the system's application flow.

4.1.2 System's Graphical User Interface

A Graphical User Interface (GUI) for the system has been implemented to present the information obtained from the Kinect sensor. It is composed of two windows and a console.

The left-hand window shows the RGB video output of the Kinect and depicts the skeleton of the person, when detected. It can be also shown the depth information from the sensor.

The window in the right-hand shows a graphical representation of the system's information. It displays the PCL point cloud of the scene, which can be navigated, and the floor plane is shown in red. Furthermore, the skeleton of the person is also showed in the screen, the segmented objects are drawn in different colors and an arrow is displayed to show the direction of the point-

ing gesture, along with a blue sphere representing the estimated pointing position, as shown in Figure 4.7. Some set-up information is needed in order to segment the floor plane (named plane coefficients) which are loaded at the system start up. However, the system asks for user intervention whether no plane can be found in the scene, or if it is the first time that the system is executed. Such intervention consists in clicking three points of the ground in order to recompute the floor plane information. Due to these processing options, images on both windows are not completely synchronized in time.

The console window prints messages about system's state and related data such as the recognized gesture or the coordinates of the estimated pointing location. An example of the GUI can be seen in Figure 4.3.

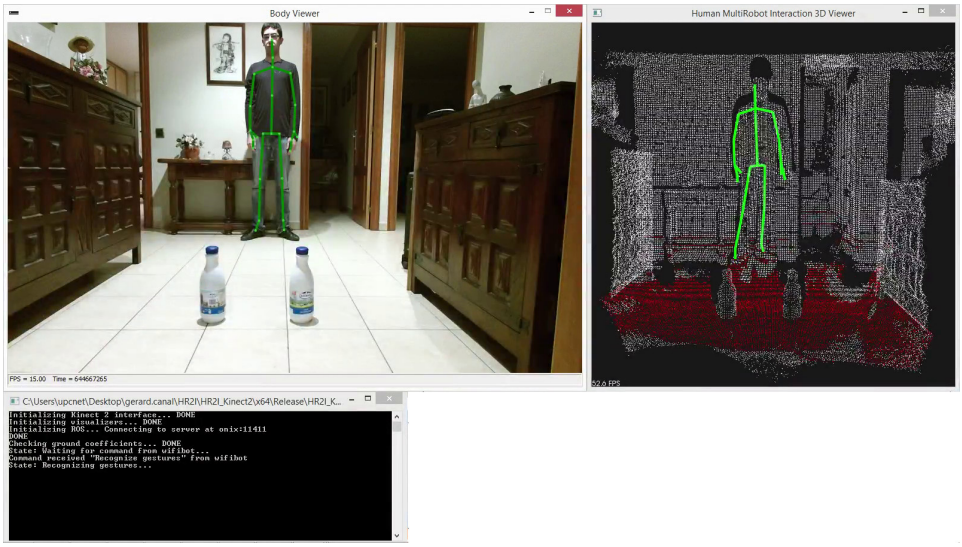


Figure 4.3: System's Graphical User Interface.

4.2 Computer Vision

One of the main parts of the system corresponds to the computer vision module. It takes care of the gesture recognition, which is the basis of the interaction system proposed in this current work.

Furthermore, it is also used in the object segmentation part, which is the method that gives the robot information about which are the possible object candidates that the user is referring to while doing some *deictic gestures*.

4.2.1 Real time online gesture recognition

This section focuses on the methodology used in order to perform gesture recognition. Given the high time constraints of the system, the recognition must be done in real time. A system in which the user performs the gesture and sees the reaction of its gesture after a long time is not much robust nor reliable, and would give raise to the user repeating the gesture a lot of times in case the robot has not seen or understood him, resulting in a non pleasant and confusing interaction.

To achieve a natural, human-like interaction, the system must be able to understand human-like gestures taking roughly the same time that would take to another human. In the proposed system, the robot reaction to a user gesture can be seen just after its execution has been finished.

The gesture recognition system has been implemented in a multi-threaded way[3] in order to speed up the process and be able to reach the real time constraint. A single threaded application would have also worked, but in a much more slower way which would probably imply the loss of the online and real time conditions, resulting in an unacceptable delay between the gesture and the response. Moreover, the recognition time would grow exponential when the number of possible gestures was increased. The algorithm takes into account the number of threads which are available in order to avoid that this part of the whole system monopolises all the assigned CPU time, given that the interface already two threads. With this information, it divides the gesture between threads in a way that each thread takes care of the processing of some gestures. The ideal situation of the algorithm is the one in which each thread only processes the information about a single gesture, but this assumption would not be realistic in a domestic computer, in which the number of logical cores of the processors is usually between four and eight.

4.2.1.1 Proposed gestures

A taxonomy for gestures is proposed for this system. The available gestures have been divided into two types: static gestures and dynamic gestures.

[3]The OpenMP API for shared-memory parallelism (openmp.org) has been used to handle the multi threading.

- A *static gesture* is defined by a static position without any movement of the user in neither the whole nor a limb of their body.
- In contrast, a *dynamic gesture* is defined by the movement the user performs with a part of the body as, for instance, the right arm.

Currently, the system supports a gesture of each type: the "Wave" gesture which is used to express a greeting, and the "point at" one, which is a deictic gesture that intends to draw the attention of the other interlocutor to some point of the space or to make a reference to any element present in the scene.

4.2.1.2 Skeletal extracted features per gesture

The gesture recognition has been performed by extracting body tracking information from the Kinect v2 sensor. Using the Kinect for windows SDK v2.0, skeletal information can be extracted from the depth images of the sensor. Such skeletal data is obtained using a method which *may* be based in the one proposed by Shotton et al. in [39], which quickly and accurately predicts the 3D position of body joints from a single depth image.

The algorithm uses an intermediate body part representation which consists on the definition of several localized body part labels which cover the whole body. This allows the authors to use a classification algorithm to solve the problem, by using the depth and labeled body part images. Then they define the features used to train the classifier. Those are simple depth comparison features, which are depth and translation invariant. This features are very efficient to compute as they do not involve any preprocessing and only the access to three pixels and five arithmetic operations are needed. As said, the features are used to classify the depth pixels into body regions. The classification algorithm they use is a randomized decision forest, which is an ensemble learning method consisting on a set of decision trees. In each leaf of the trees a probability distribution of the class given the image and the pixel is stored, being the final class obtained by averaging the distributions of all the trees of the forest. The training of each tree is performed on a different set of images, and uses the information gain metric. Once the forest has been trained, the algorithm infers the body region of each pixel of the image. That information is then gathered to generate a proposal for the joint position, which make up the skeleton. The proposals are obtained by the use of a local mode-finding approach based on mean shift with a weighted Gaussian kernel instead of just accumulating the global 3D centers of each part, as the former

was depth invariant and exhibited significant accuracy improvements. This algorithm has been used for a long time with the first Kinect sensor, first in Xbox 360 game consoles and then in desktop computer applications, showing a great performance. Figure 4.4 shows an example of skeleton obtained with the Kinect SDK.

In the gesture recognition approach that is defined in this thesis a different feature space representation is defined for each gesture. There is no need that all the gestures are defined by the same parameters or set of features, given that some gesture may be better characterized by using some specific information of a limb, for instance. The following paragraphs describe the features used for each gesture. Note that as the skeletal information is obtained in real world coordinates, all the used features are scale and body orientation invariant. Such properties avoid the need of feature preprocessing and normalization, speeding up the process.

Wave gesture's features
The wave gesture is performed by moving the forearm near and far, while keeping the upper arm in an horizontal position with the floor. Figure 4.4 illustrates the gesture.

To characterize the gesture, only two features have been used:

- θ_1**:** Euclidean distance between the Neck joint and the Hand joint[4].
- θ_2**:** Angle in the Elbow joint, which is the angle between the vector from the Elbow joint to the Shoulder, and the vector from the Elbow to the Hand joints.

Point at gesture's features
The Point At gesture is a static gesture, so no movement is involved. The gesture is defined by the elongated arm position of the user. The features for this gesture are:

- θ_1**:** Distance between the Hand and the Hip joints (both of the same body part, be it the left or right one).
- θ_2**:** The angle in the Elbow joint, as in the wave gesture.
- θ_3**:** The position of the Hand joint.

[4]A reference of the joints can be found in Figure 3.2.

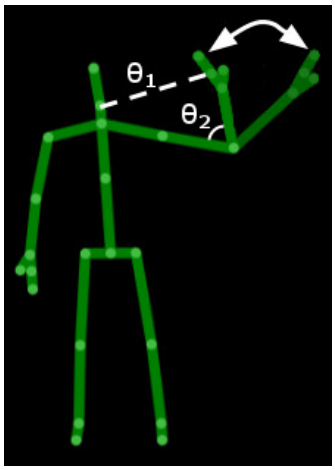


Figure 4.4: Example of skeletal wave gesture and used features.

In Figure 4.5 a skeletal representation of the gesture and its features is shown.

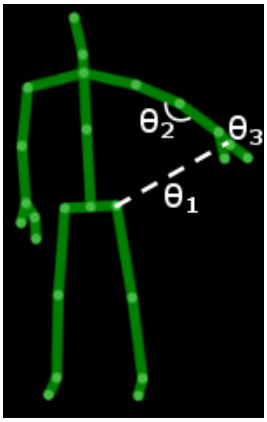


Figure 4.5: Example of skeletal pointing gesture and used features.

The features of both gestures can be extracted from both hands, but the tests have been performing by using the person's right hand.

4.2.1.3 Dynamic Time Warping (DTW)

The main gesture recognition method we use in this work is the Dynamic Time Warping (DTW). The algorithm is usually applied as a template

matching method to measure the similarity or the cost of alignment between two temporal sequences which may differ in speed or length, finding an alignment warping path. The method, originally described in [36], is also widely used to recognize gestures by detecting input sequences which are similar enough to a given reference gesture. Once detected, the whole gesture can be segmented from the input sequence by getting its warping path. Many examples of application can be found in the literature. For instance, [16] proposes a probability based DTW to recognize gestures in video streams with color and depth information. They use a Bag-of-Visual-and-Depth-Words (BoVDW) representation for the gesture information. Their approach uses the DTW to perform the segmentation of an idle gesture which is performed between gestures. Once they have the input sequence segmented, a BoVDW classification is performed by using a k-Nearest Neighbors classifier. The authors of [2] propose a robust recognition based on feature preprocessing and weighting, using as features the whole body skeleton (joint values). They use weights for the different joints and gestures to improve the discriminant capabilities of the DTW. It is a similar approach as the one in [33], from which this work is mainly based on, where the authors propose a begin-end gesture recognition system with DTW. They also use skeletal joints information as the input features to the algorithm, and weight those features (each joint) depending on its participation in a particular gesture (for instance, legs are not much important in a handshaking gesture). These weights are obtained by a training algorithm based on the ground truth of the gestures.

The method which we propose has some differences from the contribution of [33]. First, our features are not the whole skeleton but some metrics extracted from the joints of interest. This implies that the position of the non related limbs are not taken into account, avoiding the noise they would generate (as in the handshaking example). Secondly, we do not need the actual segmentation of the gesture in a begin-ends manner, as by knowing which gesture has the user performed is enough. Furthermore, different number of features are allowed for each gesture by the framework, along with a weighting on those features to add discriminant power in case of some metrics being more important, or for numerical scaling purposes (to set them to have equal importance). This method is used to detect the dynamic gestures.

The Dynamic Time Warping (DTW) is a dynamic programming algorithm. It creates a matrix to store the intermediate computations and uses them to obtain the following ones. The input data to the algorithm is a gesture model, which consists in a sequence of features corresponding to a general example of

the gesture to be recognized, and the features of the frame sequence which is obtained from the Kinect at real time (data keeps arriving and the algorithm consumes it as soon as it becomes available).

The algorithm works as follows: be the gesture reference model a sequence $R = \{r_1, \ldots, r_m\}$ and the input sequence $S = \{s_1, \ldots, s_\infty\}$, an alignment matrix $M_{m \times n}$[5] is constructed in which $M_{i,j}$ contains the distance between r_i and s_j. The input sequence S has infinite length as the system keeps getting feature frames and processing them until a gesture has been recognized. The distance metric which has been used to compute the alignment cost between two feature vectors is a weighted L_1 distance, also called taxicab distance, being it defined as

$$d_1(r, s) = \sum_{i=1}^{k} \alpha_i |r_i - s_i| \,, \tag{4.1}$$

where α_i are the positive weight constants associated with the i_{th} feature, and k is the number of features of the gesture ($k = 2$ in the case of the wave gesture).

A warping path is defined as a set of neighbouring matrix elements which define a mapping between R and S. More formally, a warping path W is $W = \{w_1, \ldots, w_T\}$, being T the length of the path, and each w_i corresponding to a matrix position $w_t = M_{i,j}$. The objective warping path is the one which minimizes the warping cost,

$$DTW(R, S) = min \left\{ \frac{1}{T} \sqrt{\sum_{t=1}^{T} M[w_t]} \right\} . \tag{4.2}$$

Note that, even though the warping path computation has been implemented in the system, it is not used in the system as the gesture segmentation is not needed.

Then, the recurrence which the dynamic programming algorithm computes to get the alignment cost is

$$M_{i,j} = d_1(r_i, s_j) + min\big\{M_{i-1,j}, M_{i-1,j-1}, M_{i,j-1}\big\}. \tag{4.3}$$

To perform the detection of the beginning and ending of the gesture in an input sequence, a segment of it which is similar enough to the model gesture

[5] n is the length of the temporal window from the input sequence which is being taken.

has to be found. Given that a perfect match is almost impossible, a test sequence is considered similar enough to a model sequence if the following condition is satisfied,

$$M_{m,k} < \mu, k \in [1, \ldots, \infty],$$

where μ is a cost threshold associated with the gesture. An example of the begin-end gesture recognition is shown in figure Figure 4.6, and the pseudo code of the algorithm in Algorithm 4.1. The code runs in a thread which is in charge of the corresponding gesture, being all the gestures processed in parallel, and keeps running until one of the threads finds a gesture in the input sequence. Once this happens, all the gesture recognition threads stop their execution to return the recognition result.

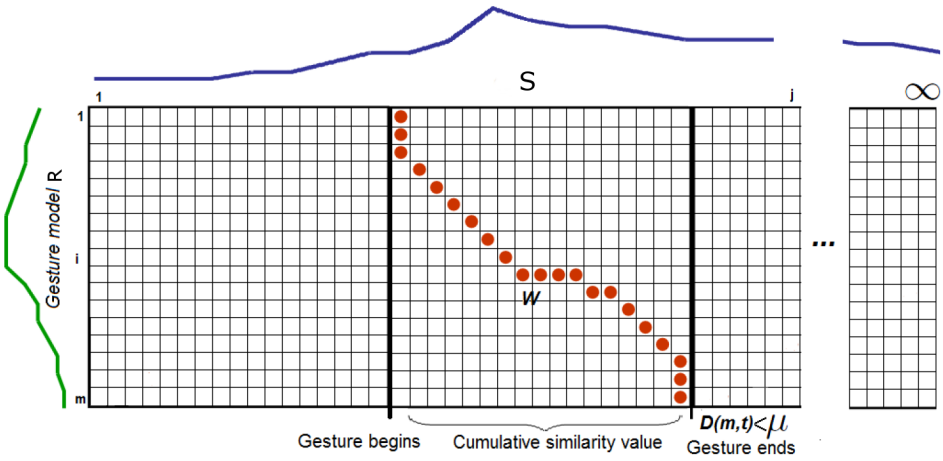


Figure 4.6: Example of begin-end of gesture recognition of a gesture model R and an infinite input sequence S using a Dynamic Time Warping matrix. Extracted from [33].

The implementation makes use of a custom defined data structure called "Sliding Matrix", which stores the matrix of the DTW. Given that the input sequence is infinite, the method must deal with an infinite matrix. As such implementation would not be possible and it is not even needed, the Sliding Matrix structure stores up to n elements, being it the length of a time window which is sufficiently large to contain the gesture. The data structure is implemented in a way in which when its capacity is filled, all the columns are slided to the left, leaving a new column to be filled. This has

Algorithm 4.1: Dynamic Time Warping algorithm for gesture recognition

Input: gId: Gesture id
Input: R: Model feature sequence of the gesture corresponding to gId
Output: Alignment cost

```
begin
    // Sliding matrix initialization
     /* Matrix of m rows and n columns initialized to infinity */
    M = Matrix(m, n, ∞)
    for j = 1 : n do
        M[0][j] = 0
    end
    // Cost computation
    t = 1
    while no gesture has been found do
        s = get_input_frame_from_Kinect()
        for i = 1 : m do
            d = d_1(R[i − 1], s)
            M[i][t] = d + min( M[i - 1][t], M[i - 1][t - 1], M[i][t - 1] )
        end
        if M[m][t] < μ_gId then
            return M[m][t]
        end
        t = t + 1
    end
    return ∞
end
```

been implemented in an efficient way, being the sliding operation of constant complexity ($O(1)$), along with element access operators. Moreover, as the gesture is not being segmented, only two columns of the matrix are needed, being the structure quite optimal in space.

To choose the different parameters of the algorithm, such as the α_i and μ of each gesture, a training method has been implemented based on different example sequences which have been manually labeled. Before this parameter selection, some tests were performed to observe the value of the different features while performing the gestures, obtaining from them a set of feasible parameter values for the α and μ values. After this, the parameter selection method consisted on using the recorded sequences, which were performed by

different users, to get the values from the set which maximized the performance in terms of overlapping[6]. Such performance was computed by testing each sequence as if it was a real input sequence, using the DTW with the current parameters and checking the obtained performance, keeping those parameters that got better results.

4.2.1.4 Static gesture recognition

Given that the static gesture recognition does not require from temporal warping but just its spatial configuration, we do not use DTW for recognize the "Point at" gesture.

The proposed solution to this problem was to adapt the recognition and make a method to handle static gestures. The method is simple: it just checks whether the input frames features are above some recognition thresholds during a certain number of frames. Another constraint is that the limbs which feature the gestures do not move much during its execution.

Consequently, the parameters involved in the static gesture recognition are the feature thresholds, the minimum number of frames the gesture has to be performed and a reference to the limb position such as the hand. The algorithm is described in Algorithm 4.2. The parameters are obtained in a similar way as it is done in the training for the dynamic gesture recognition.

4.2.1.5 Static and Dynamic Gesture Recognition Algorithm (SDGRA)

Once the system is able to recognize static gestures and dynamic gestures, both methods must be joined in a single algorithm in order to detect all the frames. This algorithm is shown in Algorithm 4.3. The actual implementation takes into account the possibility of more than one thread recognizing a gesture in the same frame and in such case it returns the one with less cost.

4.2.2 Ground plane detection and pointed point extraction

After the gestures that the user performs have been recognized, some of them may need some post processing in order to extract meaningful information.

[6]See section 5.1 Data, methods, and settings for details about the overlapping metric.

Algorithm 4.2: Algorithm for static gesture recognition

Input: gId: Gesture id
Input: P: Parameters of the gesture corresponding to gId
Output: Alignment cost `// It will be either zero or infinity`

```
begin
    consecutive_frames = 0 // Consecutive detection frames
    limb_dist = 0 // Distance that the limb has been moved
    limb_pose = (0, 0, 0) // Limb position in the previous frame
    while no gesture has been found do
        s = get_input_frame_from_Kinect()
        // Check that ∀t ∈ {0 : s.num_features}, s[t] > P.thresholds[t]
        constraints_satisfied = check_thresholds(s, P.thresholds)
        if constraints_satisfied then
            limb_dist = limb_dist + euclidean_dist(P.pose, limb_pose)
            limb_pose = P.pose
            consecutive_frames = consecutive_frames + 1
            if consecutive_frames > P.number_frames then
                if limb_dist < D_THRESHOLD then return 0
                else limb_dist = consecutive_frames = 0
            end
        end
        else limb_dist = consecutive_frames = 0
    end
    return ∞
end
```

It does not happen for the "wave" gesture, but it does for the "point at" one, in which the pointing location must be obtained.

Just three elements are needed to obtain the pointing position: the ground plane description (such as a vector which is orthogonal to it), a point from the ground plane and the pointing direction. With this, a simple geometric line-plane intersection can be computed to obtain the desired point.

To extract the ground plane, the PCL library is used. First, a depth image is obtained from the Kinect sensor, and with this a Point Cloud object is created. After that, the biggest planes of the cloud are segmented, keeping those whose normal vectors do not differ much to the reference ground plane

Algorithm 4.3: Multi threaded gesture recognition algorithm SDGRA

Input: Models: Set of dynamic gesture models
Input: P: Parameters of the gestures (both static and dynamic)
Output: Recognized gesture

```
begin
    for i = 1 : number_of_gestures do
        begin spawn_working_thread
            if is_static(i) then c = recognize_static_gesture(i,
            P.static_params[i])
            else c = recognize_dynamic_gesture(i, Models[i])
            if c < ∞ then
                stop_recognition_threads()
                recognized_gesture = i
            end
        end
    end
    return recognized_gesture
end
```

normal vector[7]. Such plane is segmented using a Random Sample Consensus (RANSAC) method [10] to generate model hypotheses. The RANSAC algorithm fits a model to observed experimental data which may contain outliers or errors in an iterative way. The plane segmentation algorithm which is being used is described in [35] and replicated in Algorithm 4.4 for exemplification purposes.

Once the plane information is obtained, a point of the plane is extracted from the plane point cloud. The last step is to obtain the line equation of the pointing direction in order to be able to get the intersection with the plane. Such line is obtained from the skeletal data, using the mean of the joint's position during ten contiguous frames from the middle-end of the gesture to make sure the correct direction is obtained and overcome possible tracking errors.

The joints of the Elbow and the Hand are used to compute the direction vector of the line, being then able to compute the final point on the ground to which the user is making reference by computing the intersection between

[7]If available. Either, user intervention is asked as explained in Section 4.1.2 System's Graphical User Interface.

Algorithm 4.4: Plane segmentation using RANSAC (from [35])

Input: P: Point Cloud (set of points to which the model will be fitted)
Output: Largest set of points corresponding to the planar model

begin
  repeat
    1. Randomly select three non-colinear unique points from P
    2. Compute the model coefficients ($ax + by + cz + d = 0$) from the points
    3. Compute distance $\forall p \in P$ to the plane model (a, b, c, d)
    4. Count the number of points $p* \in P$ whose distance d to the plane model is between $0 \leq |d| \leq |d_t|$, being d_t a user specified threshold
  until *k iterations are done*
end

the Elbow - Hand line and the floor plane. We use the Hand and Elbow joints because tests performed by using the Hand to HandTip joints direction to get the actual finger pointing direction did not improved the results but rather produced more deviated locations, due to skeleton estimation inaccuracies. Figure 4.7 shows an example of the pointing gesture result in the GUI.

Given that an intersection point is found either the user is pointing to the ground or not (as a line is infinite and does not have direction by itself), a verification on the pointing direction is done by checking the sign of the vertical component of the pointing line direction vector.

4.2.3 3D cluster segmentation for object detection

Once the "point at" gesture has been recognized and the pointing point has been located, the next step is to detect which objects are around the pointed point. As the point is to make the robot know which is the object the user is referring to, there is not much need in recognizing the objects but just detecting them, knowing there are objects there. Therefore, object recognition has not been used even though the system could be extended to handle it and actually recognize the objects and tell them by their name.

To detect the objects, the PCL has been used. First of all, a sphere of the scene of a certain radius[8] and centered in the pointed location is extracted

[8]The tests use a radius of 55 centimeters, which was suitable for the objects used.

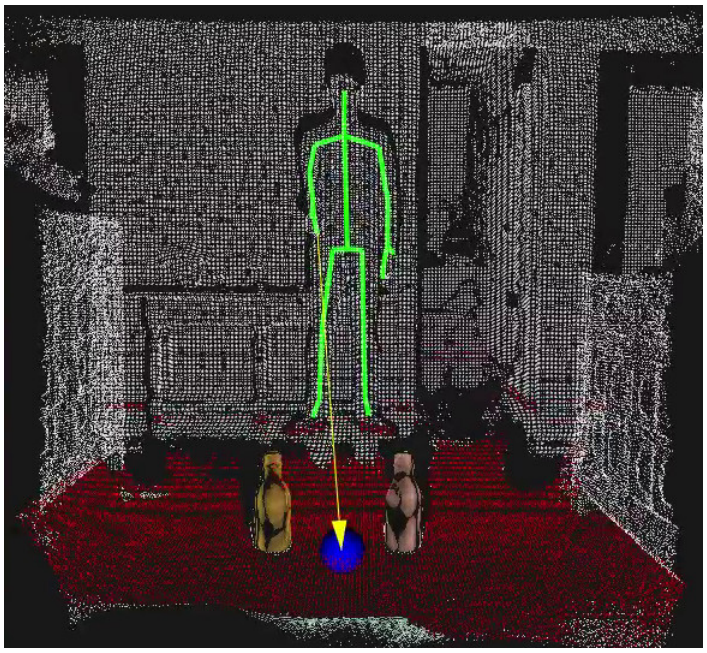


Figure 4.7: Example of pointing point estimation and surrounding objects segmentation.

from the scene point cloud. The spherical cloud contains the objects to be segmented. This extraction has been performed by constructing a PCL k-d Tree structure and performing a radius search centered in the point of interest. Later, all the planes of the spherical cloud are extracted in order to remove the floor and wall planes, in the way explained in the previous section. After that, each of the objects should be a set of points which is isolated from the others. This is a good situation to apply a clustering algorithm which joins all the neighbouring points in a single point cloud, getting as a result as point clouds as objects there are. At last, the size of the objects can be computed by means of a convex hull optimization and also the centroid of it can be computed to be used as the position of the object.

The clustering algorithm used to achieve this task is the Euclidean Cluster Extraction algorithm, which is implemented in the PCL. The algorithm makes use of the nearest neighbours idea and has some resemblance to a flood fill algorithm. The algorithm is defined in [35], and uses a k-d Tree to perform efficient nearest neighbour calculations. This clustering method is

shown as described by the author in Algorithm 4.5.

Algorithm 4.5: Euclidean Cluster Extraction (from [35])

Input: P: Point Cloud (spherical cloud from the scene in the current case)
Output: List of clusters (sets of points)

```
begin
    t = create_kdTree(P)
    C = list() // Empty list of to store the clusters
    Q = list() // Empty list of the points that need to be checked
    foreach p_i ∈ P do
        if p_i already processed then continue
        Q.add(p_i)
        foreach p_j ∈ Q do
            // Search for the set of point neighbors of p_j in a
               sphere with radius r ≤ d_th, being d_th user defined
            P_j^k = t.find_neighbors(p_j, r)
            foreach p_j^k ∈ P_j^k do
                if not Q.contains(p_j^k) then Q.add(p_j^k)
            end
        end
        C.add(Q)
        Q = list() // Reset Q to an empty list
    end
    return C
end
```

4.3 Mobile robotics: human interaction

Now that the system's vision techniques have been explained, it is time to move to the physical world. This Section introduces and explains the methodology used to endow the robot with the skills and reasoning capabilities needed to fulfill the interaction tasks, making use of the visual information (gestures and objects) obtained from the Kinect sensor.

4.3.1 Providing robots with skills

In order to implement the whole robotics application, the robot platforms must be empowered with some specific skills which they lacked. Those skills allow to develop some of the main parts of the robotics tasks, and they are essential for the success of the interaction.

4.3.1.1 Wifibot's navigation

The Wifibot's ROS driver running on the on-board computer only provides a basic interface to move the robot. The left and right velocities for both wheels can be specified, as well as the odometry and the infra-red sensors information can be read. Hence, a higher level behaviour to control the robot navigation is needed.

This behaviour will allow to move the platform towards the goal position, that pointed at by the user. Given that a free path is assumed, without any obstacles, a simple navigation towards a goal skill has been implemented. To do so, a standard PID controller [25] approach has been used, in which the control signal is the angle towards the goal and the process variable is the distance to the goal. This kind of standard go-to-goal behaviour can be found in several courses, books and papers of introduction to mobile robotics control, such as [12].

As far as the Wifibot geometry is considering two rotating axis, the right and the left one, a differential drive robot model has been employed, even though our robot is equipped with four wheels. To simplify it even more, the linear speed is assumed to be constant, and only the angular speed is modified as control output. The robot model used is

$$\begin{cases} \dot{x} = \dfrac{R}{2}(v_r + v_l)\cos\phi \\ \dot{y} = \dfrac{R}{2}(v_r + v_l)\sin\phi \\ \dot{\phi} = \dfrac{R}{L}(v_r - v_l) \end{cases}, \tag{4.4}$$

where R is the wheel radius and L the distance between the right and left wheels.

However, it is easier to work with robot velocities rather than left and right

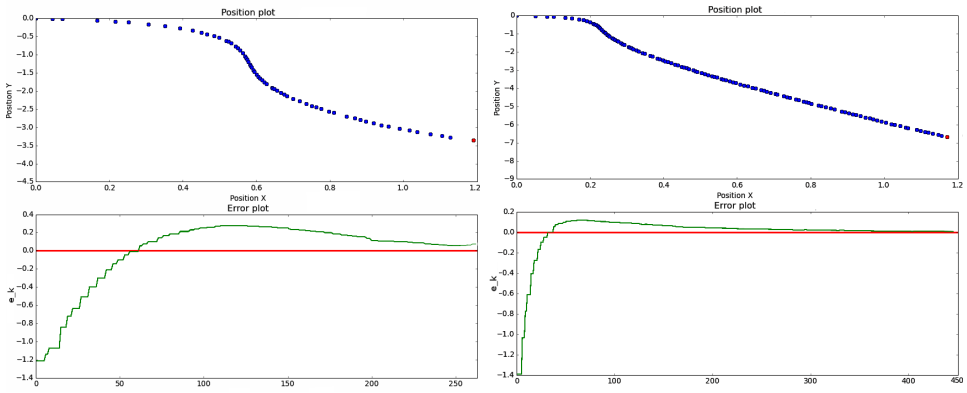


Figure 4.8: Examples of the robot's position change towards a goal (upper plots) and the angular error optimization (lower plots).

wheels ones, so the model used to design the controller is

$$\begin{cases} \dot{x} = v\cos\phi \\ \dot{y} = v\sin\phi \\ \dot{\phi} = \omega \end{cases} . \tag{4.5}$$

This odometry information is used to compute the distance and angle to the goal position.

The problem to be solved is shown in Figure 4.9. Given that the linear velocity is fixed and constant, the only variable to be controlled is the heading of the robot, which is measured by the angle ω. Let ϕ be the current heading of the robot and (x, y) its current pose, the goal orientation ϕ_g to a goal position (x_g, y_g) will be calculated as $\phi_g = \arctan\frac{y_g - y}{x_g - x}$. Then, the error to be minimized by the feedback control algorithm is $e = \phi_g - \phi$, from which the angular velocity can be obtained as $\omega = PID(e)$.

Consequently, the algorithm gets the data from the ROS's odometry topic, and each time a new update is received the angle error is computed to obtain the new angular velocity ω from which the left and right velocities are obtained and send as a command to move the wheels. Once the distance to the goal and the angular error are low enough, the robot is stopped as it has reached the goal position (x_g, y_g). Figure 4.8 shows two examples of the error optimization plot and its corresponding robot's position change towards the goal.

This feedback controller has been implemented as a ROS service which can

be called by sending a goal position and it returns once the robot has reached it.

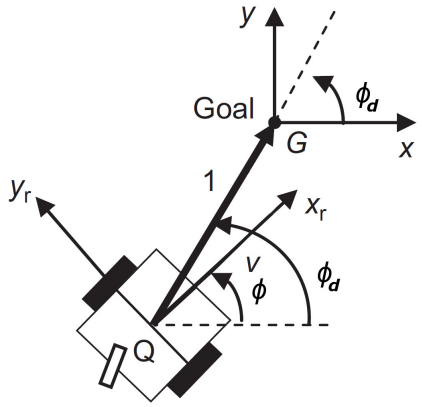


Figure 4.9: Go to goal problem in a differential drive robot. Extracted from [12].

Then, the right and left velocities can be obtained easily by

$$\begin{aligned} v_r &= \frac{2v + \omega L}{2R} \\ v_l &= \frac{2v - \omega L}{2R} \end{aligned} \quad . \tag{4.6}$$

This model can be used to compute the odometry which is provided by the driver (in fact it is computed by using the previous model), gathering the information of the wheel encoders. Only this information is employed, so the pose estimation is not very precise – as the wheels may slip –.

4.3.1.2 NAO's "going down the Wifibot" movement

Another big issue to be solved was to make the NAO robot go down of the Wifibot. Given that NAO moves with the platform by sitting on it, there was a moment in which they had to separate in order to fulfill the given task.

Such handicap was solved by creating a "going down" movement. It was designed in an iterative way. The first attempts were to move the body forward in order to generate an unbalance that translated NAO's weight to the front meanwhile it pushed himself with the knee joints (moving the legs backward). This strategy resulted in the first successes, but it some

times ended in a failure in which NAO's safety checks detected a possible fall, making him relax all the motors and lay safely in the ground, but making it impossible for the task to be completed. After some analyses on that movement, an improvement was added by also using the arms to push him forward taking advantage of the handles in the Wifibot. This tactical action improved significantly the effectiveness of the movement, which was optimized by sitting the robot a little bit more to the edge. Hence, the unbalanced movement makes the robot put its foot on the ground, and the arms help him get the correct orientation. This last version has shown a near perfect performance.

Figure 4.10: Example of NAO's going down sequence. First seated beginning to move the weight forward, then with the arms in the front and pushing with the legs to fall, and finally in the floor pushing himself with the arms.

Unfortunately, designing a movement to make the NAO go back up the Wifibot has been not possible. A little behaviour was programmed to help the user sit the NAO on top. Therefore, NAO asks the user to lift him up, it puts its joints in the sitting position and once placed on the Wifibot, the user touch its head in order to make the robot put its arms in a safety position.

4.3.2 Finite State Machines (FSMs)

The state machine paradigm has been used to program the whole application. The system can be in a finite number of states in which some conditions are held and a specific task is encapsulated. Once a state has finished its execution, the machine transits to one state or another one depending on the outcome, generating the application flow.

The considered approach has been to use states which perform a single and

little task, which are then used to build simple state machines that use the task for a more complex behaviour, which at its time is included into another state machine, giving rise to a complex hierarchical state machine that executes the whole application flow. An example of this state machine is shown in Annex A State Machine diagrams. Several states which are specific to this Human-Robot interaction application have been developed, but a lot of already made states from the NAO@UPC[9] repository were used, as far as many of them were contributed by the author.

For the Kinect controller application, a single state machine with three states has been implemented: the *wait state* waits until a command is received from the server laptop, and depending on it make the transition to either the *gesture recognition state* or the *object segmentation state*, both of them transiting back to the *wait state*.

4.3.3 Robot interaction

The overall objective of the methods introduced is to generate a single gesture based interaction of a human with a robot. Since NAO is a humanoid robot, it can interact with a human person both via gestures and voice, as people knows how to interpret those gestures in a similar form to those made by a human. However, this similarity also involves some prejudices as users may expect the robot to act and behave *as* a real human. So, one aside objective of the interaction is to avoid user frustration due to unexpected actions or unintelligible gestures from the robot side. In addition, successful but also funny interaction will increase engagement for the user and enjoy more the situation.

To accomplish all of these goals, different elements have been used. To begin with, the response gestures are as human-like as possible, from the wave gesture which is similar to the one the user has to perform to the gesture it does to reference an object, in which NAO shows which was the object the user pointed at with his hand, but also accompanying it with a head movement. The movement to go down of the Wifibot is also quite natural, despite being it more similar to the one a clumsy person would do.

Secondly, the robot verbalizes almost everything it is doing or is going to do.

[9]Repository of NAO utilities and SMACH states developed for the Humabot 2014 challenge which was held in Madrid during the 2014 IEEE-RAS International Conference on Humanoid Robots. The code is found in the GitHub's NAO-UPC repository.

This communication allows the user to be informed and enlivens the process. To make it even more pleasant, a set of messages are available for the robot for every utterance, from which it chooses one of them at random, avoiding repeated sentences. Hence, even though there exist some states which are repeated in the process, the robot says something different every time. For instance, when NAO has not recognized any gesture after a given amount of time, it says something like "I did not see you moving. Are you there?"; when the timeout is reached again, it could say "Please, do a gesture", among other sentences. Furthermore, utterances have been designed to be a little bit funny so the robot does not look like a cold machine. An example would be that when the user points at an object but the robot is not riding the Wifibot (since it has already gone down); it could say "I can not get there without my cool Wifibot transport". When the users points at a non-ground place, NAO could tell the user that "I can not fly! Please point at the ground". However, if the user repeats the same action, the robot will say something different, maybe in a more severe tone.

Finally, the eye LEDs are also used to send information to the user, although it has only been used in a specific case. This is the moment in which NAO is listening to the user answer its question. To indicate it is listening, the robot emits a beep sound, and puts its eye LEDs in a blue circular animation, which means it is waiting. When a sound is being recorded, the light turns yellow, and when a recorded signal is being processed, the eyes are in still white with the blue circular animation in the background, meaning that it is still listening while processing the previous information. If the processed signal is a successful recognition, the eyes blink in green, while if it was an erroneous one they blink in red. A final beep signal tells the user that it has stopped listening, after it understood what the user said. This is what the default speech recognition behaviour does, and it was kept because we thought it was informative and intuitive enough. The eye's LED lights execution flow is depicted in Figure 4.11.

Even though, not all the process could be as human like. For instance, when the object segmentation is performed, the robot tilts its head back in order to move it away from the Kinect's field of view. This action results in the NAO "looking for objects" in the ground but looking to the ceiling, which may seem incoherent for the user. Moreover, the robot does not follow the user's face with its gaze and it may be quite impersonal, although in most of the movements the robot moves the head up so it seems like it is looking to the user.

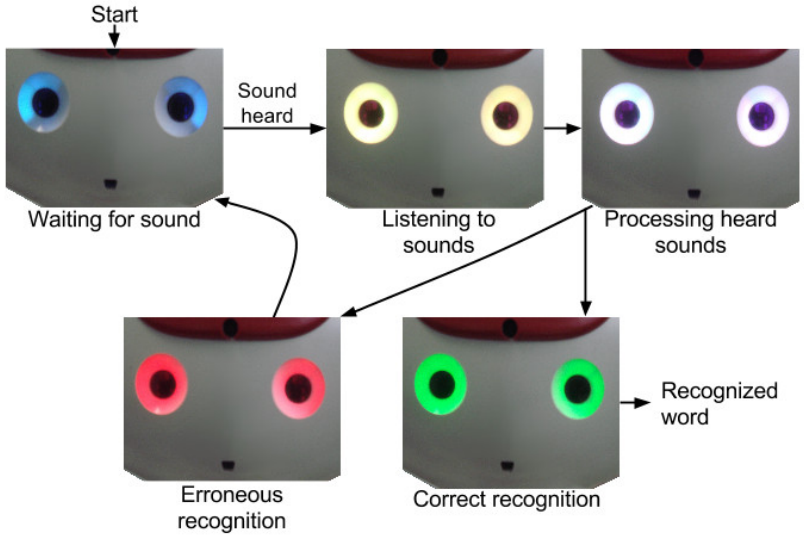


Figure 4.11: NAO's eyes LEDs while performing speech recognition.

4.3.3.1 Object disambiguation

Another special case of interaction is performed by the object disambiguation sub state machine. It is the one which processes the object segmentation result, and outputs the object which the user selected.

First of all, the state tries to discriminate whether the pointed location was close to a single object and far enough of the rest of them. If this is the case, there is no doubt and that object is selected. Else, the robot needs more information in order to accomplish its task, and makes a question to the user. Again, the question depends on different factors in order to make it more varied and easy for the user. Provided that the objects are significantly different in size, the robot asks whether the biggest object was the one the user wanted. If the user answer is negative, it will know that the desired object is the other one, if there are only two objects or will ask again if it is the smallest one, so it can then know which object was. In case the objects can not be differentiated by its size, it asks to the user if the desired object is the one in NAO's left, repeating the same discarding method as for the size question. A flowchart of the process is depicted in Figure 4.12.

The object disambiguation process involves a natural dialogue with the user, enhancing the interaction along with the ways of interacting, and simplifying

the task while improving the chances of success by removing any guessing or random choice when there is a doubt.

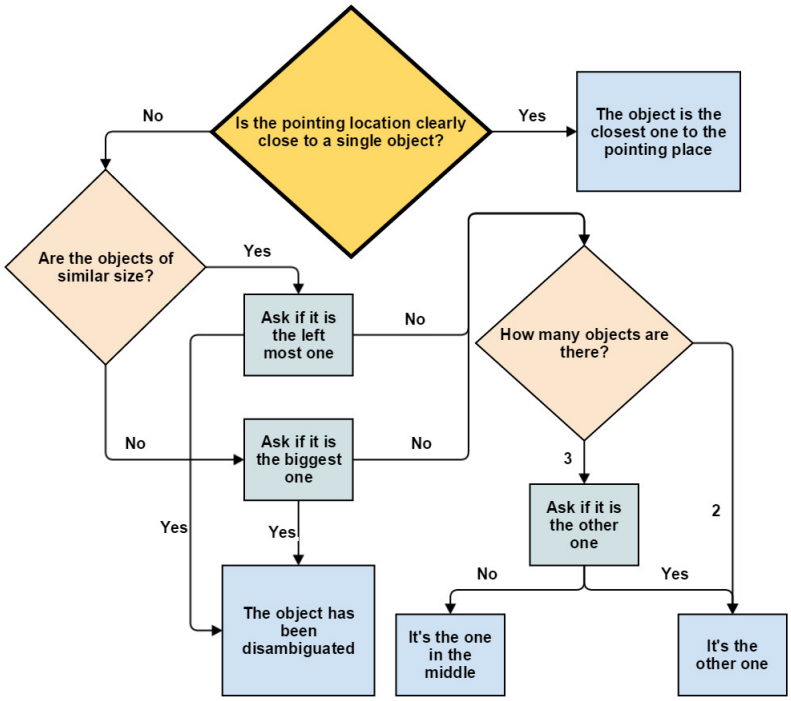


Figure 4.12: Flowchart of the object disambiguation process.

5 Experimental Evaluation

The system implemented and tested, some evaluation is needed in order to obtain some performance measures that give some idea about how the different methods behave. The focus of this evaluation will be on the gesture recognition and interaction parts, first introducing the evaluation methods used, then showing the gesture recognition evaluation and finally user experience results.

5.1 Data, methods, and settings

Several data has been collected in order to perform an exhaustive evaluation of the methods proposed in this work. To begin with, seven skeletal sequences of three different users doing the proposed gestures were recorded in order to use them to obtain the gesture recognition parameters as explained in 4.2.1.3 Dynamic Time Warping (DTW), but also to compute a performance metric on the gesture recognition method. Those sequences were manually labeled to obtain the starting frame and the ending frame of each gesture appearing in them. The sequences have a total of 2082 gesture frames and 61 gestures, 27 of them being static gestures and the other 34 dynamic gestures.

The chosen metric is the overlap measure, also known as Jaccard Index. It is computed as

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{5.1}$$

where A is the ground truth information (set of frames which contains a gesture) and B is the set of frames in which a gesture has been recognized. An index value of one means a perfect recognition to the frame level, while a zero would mean that no frame was correctly recognized. Figure 5.1 depicts an example of its computation.

The other source of data comes from user experience evaluation. Several users were invited to perform different tests with the system and then fill a questionnaire to state their opinions and answer some questions about how they felt the interaction. Those tests were videotaped to better analyse the results. User information gives us an idea of how good the system behaves in terms of user experience rather than getting some numbers which depend on some predefined sequences. Given that the objective is to generate a natural interaction application, user opinion and system performance while

operated by different users gives us a more real world idea about how well does it perform.

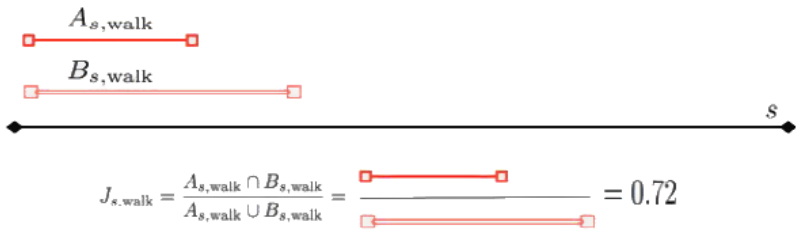


Figure 5.1: Jaccard Index computation example. Extracted from [9].

5.2 Gesture recognition evaluation

To obtain a general measure of the performance of the algorithm on the recorded skeletal sequences, a Leave-One-Out Cross-Validation (LOOCV) strategy has been used. Hence, one sequence is left out from the threshold selection method (explained in 4.2.1.3 Dynamic Time Warping (DTW)) and the other ones are used to compute the better thresholds to recognize the gestures they include. After the parameters were computed, the test sequence was evaluated to obtain its overlap measure in unseen data. This procedure is repeated for each one of the sequences, obtaining the performance measures both for the static gestures and the dynamic ones. Those measures were averaged for the seven sequences, obtaining a general Jaccard Index of each type of gesture, and the average of the mean overlap of both categories gives the final gesture recognition performance value of the system.

The results obtained with this methodology are shown in Table 5.1. Each fold is a test sequence, and the results of the parameter selection with all the sequences but the fold sequence are in the left columns, while the right ones show the performance on the fold sequence with the parameters obtained with the rest of the data. The "Global" row shows the results of the parameter selection when all the sequences are used to compute the parameters, and the test sequence columns show the mean of the above rows.

As it can be seen in Table 5.1, the mean Jaccard Index in unseen sequences is about 0.49. This means that roughly the 50% of the gestures are well recognized from begin to end – although more than the 50% may be correctly

Fold	Parameter selection			Test sequence		
	Static gestures	Dynamic gestures	Mean	Static gestures	Dynamic gestures	Mean
0	0.703642	0.552158	0.627611	0.349593	0.636364	0.49298
1	0.641827	0.658219	0.650023	0.711538	0.0	0.35577
2	0.713837	0.557703	0.63543	0.279476	0.603093	0.44128
3	0.71538	0.625359	0.67037	0.172078	0.186992	0.17954
4	0.640198	0.554127	0.597163	0.721311	0.624549	0.67293
5	0.667304	0.528129	0.59772	0.543605	0.77037	0.65699
6	0.595776	0.54133	0.56855	-[a]	0.620818	0.62082
Global	0.653063	0.564187	0.608625	0.46293	0.491744	0.48862

[a] Sequence 6 does not contain any static gesture.

Table 5.1: Gesture recognition performance evaluation results.

recognized, but not segmented –, which is not a bad result but still could be much better. The parameter selection results are better, reaching the 60% of overlap in the gestures, mainly because all the sequences are used to compute them, as more variability considered and it is slightly over fitted, like a training error. As for the different kinds of gestures, it looks like dynamic gestures are better segmented than the static ones. However, some sequences failed in recognizing the dynamic gestures, such as the sequence 1. This may be caused by the little amount of data and the variability of the gestures between users, therefore if the other sequences are similar this could lead to this result. A solution, leading to a more accurate measure, would be to get more data, but it was not feasible mainly due to the high amount of labeling work but also to the lack of candidates at the time of the skeletal sequences recording, which was at the early stages of this project. Furthermore, the user experience evaluation is another valuable source of performance information which compliments the data exposed in this section.

5.3 User experience evaluation

The system was prepared to be tested publicly in the Facultat de Matemàtiques i Estadística (FME). A total of twenty-four people came there to try the Human Robot Interaction application and give their opinion about it.

5.3.1 Experiment design and setup

The test began with a simple explanation of the objectives of this work and the task to perform along with an introduction to the NAO robot. Then the available gestures were explained and an example of the wave gesture was shown to the users, as it can be done in several different ways – moving only the hand or moving the whole straight arm –. After that, the test was performed and finally the user answered the survey about its experience.

Each user had to perform three tries of the task: one in which only one object was placed in the robot area, another one with two of them and a final one with one of the two objects replaced. Those who desired it could make more tests. The objects which were used were two milk bottles and an empty cookie box. The order of the tries was changed between users to avoid any bias in the results due to user fatigue. The objects used in each of the test was also varied, being some tests performed with the two bottles and others with a bottle and the box. There were no restrictions about the order of the gestures to be performed, but users tended to begin with a wave gesture to then point at an object. Also, the objects were usually placed by the test controller, but those who asked were allowed to place the objects by themselves.

The environment in which the tests took place was a corridor of the FME building, given that a big open space was needed to both receive the people and to perform the tests. The testing setup can be observed in Figure 5.2. It consisted on a table in which the main laptop was placed, the robots and the objects were in the middle of the corridor and a desktop computer in the contiguous room was ready for the users to answer the questionnaire. A camera was settled at the other side the corridor in order to record the tests, prior informed consent of the user. Given that a battery for the Kinect sensor could not be obtained, a power supply extension cord was needed.

This testing environment was a little bit troublesome because of the high reflectivity of the floor, which made the Kinect see the objects mirrored on the ground, as the infrared projections got reflected on the ground. This resulted in the appearance of underground objects, and a fix was needed to be implemented in order to filter such reflections. An example of the issue is shown in Figure 5.3.

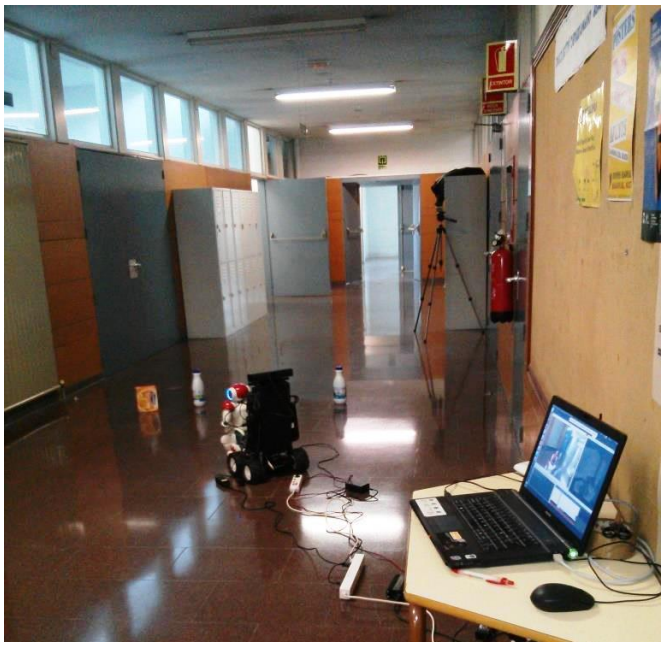


Figure 5.2: User testing environment.

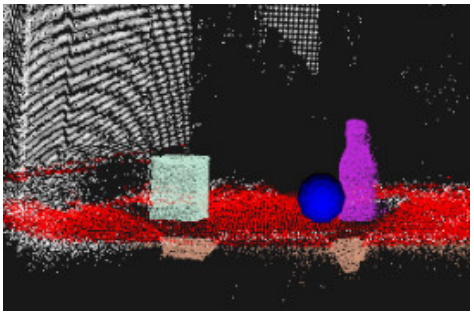


Figure 5.3: Example of object reflections on the ground.

5.3.2 User's survey analysis

This section highlights some interesting results which were obtained from the answers to the questionnaire that the twenty-four volunteers who participated in the experiment had to fill after the test. The questionnaire can be found in the Annex B User tests questionnaire. Tables and Figures in this section refer to the questions as "QX" where X is the question number.

To begin with, demographic data was asked to know which is the profile of the users that tested the system. As it can be seen in Figure 5.4 and Table 5.2, users aged from 21 to 42 years, being the range of 26 to 30 years the most common one, and being them mostly male as it can be seen in Figure 5.5. As for the background, many people were from the robotics field, computer science and artificial intelligence, but a great percentage of them had very different backgrounds. For instance, a medical doctor, an optician, a translator or a school teacher are examples of this variation, being most of them still studying. Consequently, the system was tested with a great variety of people from different education levels and ages.

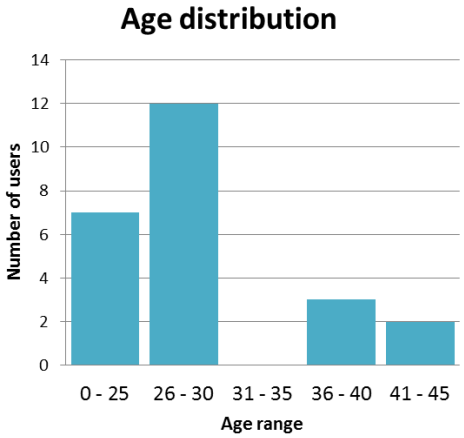


Figure 5.4: User's age distribution (Q1).

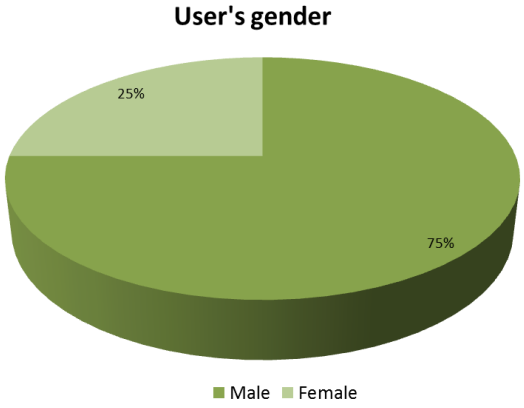


Figure 5.5: User's gender distribution (Q2).

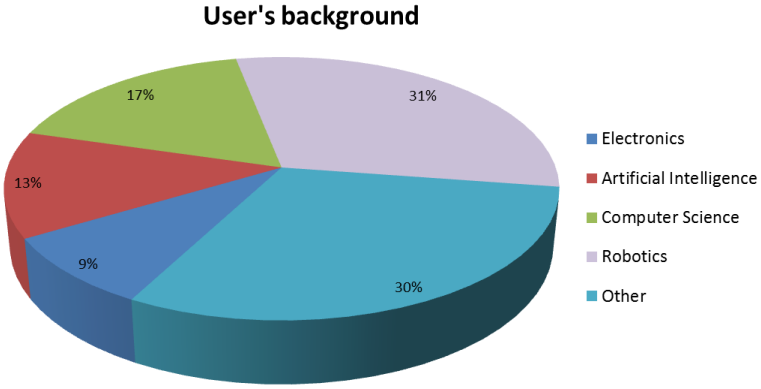


Figure 5.6: User's background distribution (Q4).

The answers to the interaction questions were mostly in the positive sense. For the "Wave gesture", a 96% of the users expected the robot's behaviour, and they though that this response was quite fast, being the mean a 4.21 out of 5, and the lowest score was a 3. Also, as it can be seen in Figure 5.7, they though the gesture was natural to perform, even though some users felt it was hard to perform or not natural. Some suggestions on the wave gesture were that it should be able to understand different type of waves (for instance with only the hand), or maybe a verbal salute. They also thought it would be good to wave with both arms (it was restricted to the right hand for the tests), that NAO could try to make visual contact with the user, or that he should speak the "hello" faster.

The "Point At" gesture was also agreed to be natural – as it is shown in Figure 5.7 – and also fast, even though not as fast as the "Wave one". Most of the users thought that they did not have to point at the object for too much time, that it usually got the correct object and that the robot showed which was the object it understood. The disambiguation part did not appear in all the tests, as the robot saw clearly which object was the user pointing at. In the cases the robot needed to ask, most of the volunteers understood what it said, even tough a high percentage of them (25%) did not and needed external help, and another 25% of them did not know if NAO understood them or not. But almost all of them agreed that NAO's behaviour was the one they expected after their gesture. Some comments about the "Point At" included the difficulty for the user to know when they could point at the object, and some failures at the time of detecting more than one object, as well as it could be faster or more natural with the arm more relaxed.

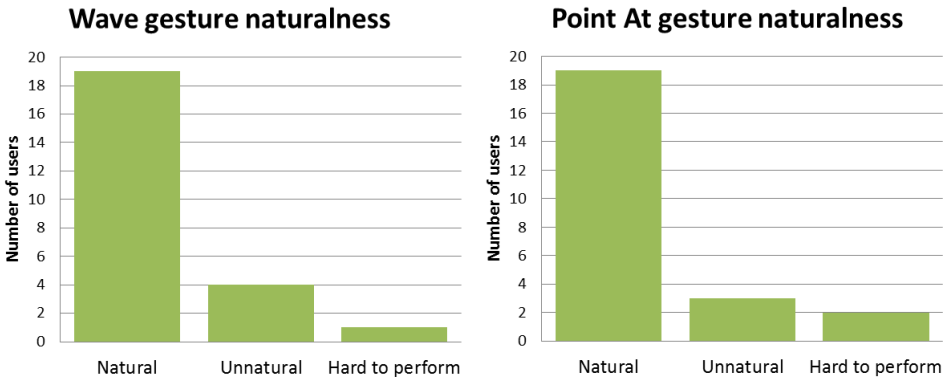


Figure 5.7: User opinion about gesture naturalness (Q8 and Q13).

Question	Range	Min	Max	Mean ± SD
Age (Q1)	-	21	42	28.46 ± 6.14
Familiarity with robots (Q3)	1-5	1	5	3.75 ± 1.33
NAO's response speed to a Wave (Q9)	1-5	3	5	4.21 ± 0.88
NAO's response speed to a Point At (Q14)	1-5	2	5	3.83 ± 0.82
NAO got the correct object (Q16)	1-4	2	4	3.00 ± 0.78
NAO clearly showed its object guess (Q17)	1-4	2	4	3.17 ± 0.76
Naturalness of the whole interaction (Q24)	1-5	2	5	3.71 ± 0.91

Table 5.2: Numerical user's answers to the survey.

Looking at the whole test interaction, the users graded it with a mean value of 3.71 ± 0.91 out of 5, which indicates that the interaction was sufficiently natural. Almost all of them were able to understand the robot indications (although some needed help), and only a 29% felt frustration, which is a feeling that usually arises when dealing with robots. This frustration appeared when the pointed point was erroneously estimated and the robot went away or when the robot did not get the gesture. Some of them were not aware about what was the robot doing at some moments such as when it was waiting for a gesture after waving, and some of them were able to understand what was going on while the robot was performing its tasks. Another important result is that most of them thought it was easy and satisfactory to interact with the robot, and all of them enjoyed the test.

Some questions about future extensions were also asked. For instance, they asked about the possible integration of head movements to answer "yes" and

Question	Yes	No	I don't know	Need to repeat	Robot didn't ask
I am still Studying (Q6)	83%	17%	-	-	-
The wave response was expected (Q10)	96%	4%	-	-	-
I had to point for too much time (Q15)	33%	67%	-	-	-
I understood robot's question (Q18)	58%	17%	-	-	25%
The robot understood my answer (Q19)	50%	0%	25%	-	25%
It was easy to anwer the question (Q20)	46%	8%	-	8%	38%
The Point At response was expected (Q21)	92%	8%	-	-	-
I understood robot indications (Q25)	88%	13%	-	-	-
I felt frustrated (Q26)	29%	71%	-	-	-
I had unawareness of what was happening (Q28)	25%	75%	-	-	-
It was easy to make NAO do a task (Q30)	96%	4%	-	-	-
It was a satisfactory interaction (Q32)	88%	13%	-	-	-
I enjoyed the test (Q33)	100%	0%	-	-	-
It'd be better to answer with a head movement (Q34)	58%	42%	-	-	-
It'd be easier to answer with the head (Q35)	54%	46%	-	-	-
It'd be better to have both answering modes (Q36)	100%	0%	-	-	-
Application could be useful in homes (Q37)	96%	4%	-	-	-

Columns with a - were not an option in the question.

Table 5.3: Rest of the answers to the questionnaire.

"no" questions. There is not a clear tendency about whether this would improve the interaction or it would be easier, even though the majority thought so. However, they all think it would be good to have the ability of answering both via head movements and speech. Also, most of them said that applications like this one could be useful in household environments to help the humans.

Finally, they proposed new gestures to be added to the system. Some of the most interesting ones are a "stop" gesture with the open hand in front (which would be a static one) was repeated a lot of times. "Come here" and "go there" gestures were also recurrent, and silent gesture to mute the robot or put it in sleep mode was also proposed, along with a "help" gesture. Some other suggestions they provided were to give more feedback about what was the robot doing, that an obstacle avoidance system or a Kinect based correction for NAO's walking could be implemented. Some users also suggested that it would be good to make the NAO grasp the objects to, for example, help people with reduced mobility.

Tables 5.2 and 5.3 show the detailed results which have been described above.

5.3.3 External test and user's behaviour analysis

The external observations of the tests, taken by the test controller, are also a valuable source of information about the performance and proper functioning of the system.

The users showed an appropriate learning curve, and minimal external intervention was needed. They usually began by pointing too high and the estimated point was too far away of the objects, hence NAO could not see them, but they learned that they should point more precisely to the objects. Besides, some left handed user tried naturally to perform the gestures with the left hand even though they were told to do it with the right one. Also, most of them began with a wave, and then waited for the controller to allow them to point at, when the robot was already waiting. This could be the sign of a lack of feedback from robot's side, and they usually asked before performing the gestures instead of trying things freely. Nevertheless, some users tried to see what would happen when pointing at an object when the NAO was already on the ground, and got a pleasant surprise when NAO answered that he could not go there. Some other people tried to harden the tests by placing the objects by themselves, some times resulting in a too hard placement (with obstacles in the middle).

Moreover, some issues arouse while testing the system with the volunteers. In the beginning of the experiments, the robot was not walking properly and showed a deviation to the left, which needed to be fixed. This may have been due to the slipping floor.

The height and clothing of the person also had influence in the tests. For instance, some users were too tall to have the skeleton estimated from where they were placed, their hands went out of the field of view during a wave or they had to walk back after the Wifibot had moved in order to be in the Kinect's field of view to perform a new gesture. What's more, some clothes – usually black, as some black dyes absorb infrared light – made that the skeleton could not be tracked. This fact implied that one user had to take off its jacked, and another had to cover his trousers. Other problems in the skeleton estimation are shown in Figure 5.8.

The final evaluation after all the tests were performed is that the tests were performed correctly, and the users enjoyed them. it was clearer in those users who had not had much contact with robots, and even they asked to repeat the test more times. The robot speech was hard to understand for some users due

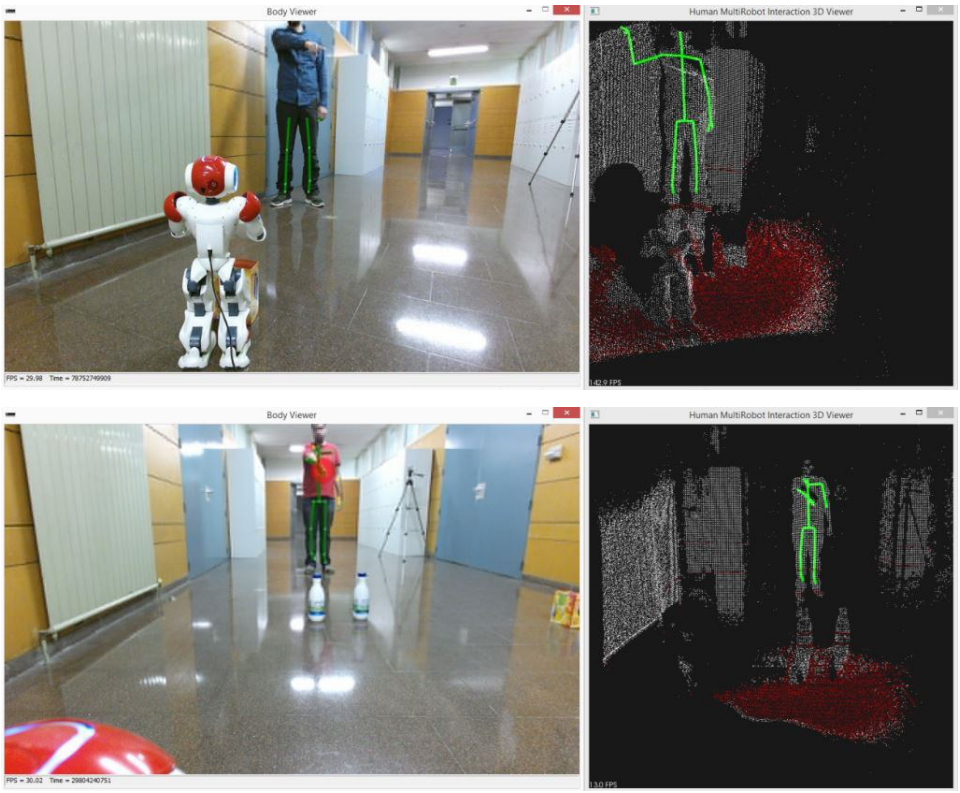


Figure 5.8: Skeletal tracking errors during the tests.

to NAO's spelling, but they got used to it at the end. The batteries supposed some problems and NAO was needed to be plugged, even though it did not affect the test performance. Some good points are that, even though the pointed location was deviated, NAO was able to approach the location and refine the search to find the correct object at the end. But, this fact implied that not many disambiguation speeches were performed, which is not bad, but the speech part could not be studied in the same depth as the others. External intervention from the collector was only needed when the user did not know what to do or when the robot went to a wall, in which case was stopped for safety reasons. Furthermore, NAO's performance was correct in almost all the tests, failing only twice: when the whole robot computer got frozen and once that he got trapped with the cable. Moreover, the robot did not fall to the ground any time, and was able to successfully go down the Wifibot all the times he needed to.

6 Conclusions and future work

A real-time gesture based Human Robot Interaction system has been proposed and implemented in this Master Thesis with the novel Microsoft's Kinect v2 sensor. Two types of gestures have been described, the static gestures and the dynamic ones, and a gesture of each type has been included in the system. Features obtained from skeletal information have been used in the algorithm, being them gesture independent and allowing different number of features per gesture. Moreover, the system allows easy addition of new gesture with its specified features. An implementation of a feature weighted Dynamic Time Warping algorithm has been applied to the dynamic gesture recognition, and a parameter selection method has been implemented to tune the different values of the algorithm. Additionally, some utilities from the PCL library have been used in order to detect the floor plane in order to find the user's pointing location, and an Euclidean cluster extraction algorithm has been employed to segment the object candidates around the referred point.

A medium size humanoid robot, Aldebaran's NAO, has been used to interact with the user via both speech and gestures, and a wheeled platform, Nexter Robotics' Wifibot robot, is employed to ease NAO's navigation and sensor movement. NAO is able to ride Wifibot and to go down of it once they have approached a given goal. Both robots work in an independent way, and they are able to collaborate each other in order to fulfill a task which right now includes, but it is not limited to, finding an object which has been referred by the user with a pointing gesture, with a speech based disambiguation using spatial or dimensional characteristics. Some extensions to this task could be adding more types of interactions or using the Wifibot to see the elements from other points of view.

Furthermore, an extensive series of tests with a varied set of users have been carried out, resulting in a good experience for them. Most of the users thought the gestures were a natural way of interacting with the robot, and the response the robot had was the expected one and fast enough. This means that the system is easy to be used for human beings with minor indications on how to perform the gestures and, thus, the initial objectives of this project are considered as successfully accomplished.

It has been shown that applications like this could be really useful in household environments, as the users suggested. Make robots bring something they pointed could be useful for elderly people or those which mobility dif-

ficulties, but waving at them and having a response give them a living and understanding feeling, helping to avoid loneliness. Many other gestures could be added in order to improve this interaction that rather than teleoperating the robot, intends to be a source of information to ease robotic task fulfillment, everything made in a natural, non forced way.

However, many extensions and improvements could be included in the system, as the amount of possibilities is as huge as human imagination, and some limitations needed to be added to the project. One of the most important ones may be the enhancement of the pointing at location estimation, as the elbow-hand direction tends to point to further places, and also humans tend to point above the object. This is not a problem for us to distinguish the object, but it is a handicap for a robotic system. Some solutions to this issue may be the use of a learning method in order to adapt to the user, be it a general user – to get the "general" human pointing deviation – or user specific, or a fixed factor could be applied to solve the major deviations. Also, other cues could be used to improve the estimation of the pointing direction, such as the use of the gaze trajectory in other to adapt the arm one, as humans tend to look to the place they are pointing at.

In addition, more gestures could be added, as the ones suggested by the users, as well as more variability to the current ones, as the wave gesture can be performed in different ways and, even though the proposed way has demonstrated to be natural, some users would have performed it in a different way. Other interesting gestures to add would be head gestures, such as nodding to confirm NAO's suppositions. This would be performed by means of Kinect's face tracking utilities and extracting 3D pose information of, for instance, the eyes and mouth to build gesture features which can be fed to the algorithm. This possibility could enhance the gesture interaction of the system, and a great number of users thought it would be good to have, provided that the speech capability is still present. Also, as some users suggested, better feedback should be included in order to avoid that the user does not know what is happening with the system at any moment.

These extensions could be also used to improve the skeletal tracking, which showed some problems as phantom skeletons or tracking errors as the ones already shown. Using a face detector on the RGB image could be helpful in order to filter some of these errors, as if no face is found it would mean that no person is in front of the camera. Given that the skeletal tracking is the main basis of the system and depends on it, failures on that method imply the inability of the robot to understand the gestures and hence it should be

the most robust part of the application.

Another issue some users complained about is that the height of the person can imply if the user is recognized or not, and the distance needed to the sensor as they needed to move or their arms went out of the sensor's field of view. This could be tackled in the previous Kinect version by using the included pan and tilt motor, which was discarded in this renewed sensor. Nevertheless, some external motor could be included in order to solve this point.

Besides, different improvements could be applied to the navigation component. Firstly by taking obstacles into account in Wifibot's move to goal behaviour, in order to have a more robust approach. Then, NAO tracked could be used in order to improve NAO's walking behaviour with a closed loop controller, seeing when it deviates and correcting such deviation to assure that it reaches its goal successfully. In addition, a SLAM algorithm could be used to create a map of the environment, which would enormously ease both robot's navigation capabilities, and would give information about the exact location of each robot on a map, which would the need of relying on the odometry of the robots.

Finally, a cognitive architecture could be used in the robotics part of the system. Following the approach of [29], in which the SOAR[1] reasoner is applied to a general purpose service robot, the state machines could be replaced by it. This would simplify the programming of the robots, easing the addition of new gestures as there would not be the need of taking all the possible transitions into account, as SOAR would create the appropriate plan obtaining any needed information. A first coding of our system skills into SOAR operators was performed, but this method could not be tested deeply enough to extract sufficient conclusions.

[1] soar.eecs.umich.edu

References

[1] V. Alvarez-Santos, Iglesias. R., X. M. Pardo, C. V. Regueiro, and A. Canedo-Rodriguez. Gesture-based interaction with voice feedback for a tour-guide robot. *Journal of Visual Communication and Image Representation*, 25(2):499 – 509, 2014.

[2] T. Arici, S. Celebi, A. S. Aydin, and T. T. Temiz. Robust gesture recognition using feature pre-processing and weighted dynamic time warping. *Multimedia Tools and Applications*, 72(3):3045–3062, 2014.

[3] P. Barros, G. I. Parisi, D. Jirak, and S. Wermter. Real-time gesture recognition using a humanoid robot with a deepneural architecture. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids '14)*, pages 83–88. IEEE, 2014.

[4] E. Bernier, R. Chellali, and I. M. Thouvenin. Human gesture segmentation based on change point model for efficient gesture interface. In *Proceedings of the 2013 IEEE RO-MAN*, pages 258–263, Aug 2013.

[5] F. Bettens and T. Todoroff. Real-time DTW-based gesture recognition external object for Max/MSP and Puredata. In *Proceedings of the Sound and Music Computing conference (SMC '09)*, pages 30–35, 2009.

[6] C. Breazeal, C.D. Kidd, A.L. Thomaz, G. Hoffman, and M. Berlin. Effects of nonverbal communication on efficiency and robustness in human-robot teamwork. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005)*, pages 708–713, Aug 2005.

[7] J.A. DeVito and M.L. Hecht. *The Nonverbal Communication Reader*. Waveland Press, 1990.

[8] D. Droeschel, J. Stuckler, and S. Behnke. Learning to interpret pointing gestures with a time-of-flight camera. In *Proceedings of the 2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 481–488, March 2011.

[9] S. Escalera, X. Baró, J. González, M. A. Bautista, M. Madadi, M. Reyes, V. Ponce, H. J. Escalante, J. Shotton, and I. Guyon. Chalearn looking at people challenge 2014: Dataset and results. In *Proceedings of European Conference on Computer Vision (ECCV) 2014, ChaLearn Looking at People Workshop*, 2014.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commununications of the ACM*, 24(6):381–395, June 1981.

[11] T. Fujii, J. Hoon Lee, and S. Okamoto. Gesture recognition system for human-robot interaction and its application to robotic service task. In *Proceedings of The International MultiConference of Engineers and Computer Scientists (IMECS 2014)*, volume I, pages 63–68. International Association of Engineers, Newswood Limited, 2014.

[12] Tzafestas S. G. Mobile robot control I: The lyapunov-based method. In Spyros G. Tzafestas, editor, *Introduction to Mobile Robot Control*, pages 137 – 183. Elsevier, Oxford, 2014.

[13] F. Garcia, R. Frizera, and E. O. Teatini. Human–robot interaction and cooperation through people detection and gesture recognition. *Journal of Control, Automation and Electrical Systems*, 24(3):187–198, 2013.

[14] Bill Gates. A Robot in Every Home. *Scientific American Magazine*, January 2007.

[15] Md. Hasanuzzaman, T. Zhang, V. Ampornaramveth, H. Gotoda, Y. Shirai, and H. Ueno. Adaptive visual gesture recognition for human–robot interaction using a knowledge-based software platform. *Robotics and Autonomous Systems*, 55(8):643 – 657, 2007.

[16] A. Hernández-Vela, M. Á. Bautista, X. Perez-Sala, V. Ponce-López, S. Escalera, X. Baró, O. Pujol, and C. Angulo. Probability-based Dynamic Time Warping and Bag-of-Visual-and-Depth-Words for Human Gesture Recognition in RGB-D. *Pattern Recognition Letters*, 50(0):112–121, 2014. Depth Image Analysis.

[17] S. Iengo, S. Rossi, M. Staffa, and A. Finzi. Continuous gesture recognition for flexible human-robot interaction. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation, ICRA 2014*, pages 4863–4868, 2014.

[18] H. Kim, S. Hong, and H. Myung. Gesture recognition algorithm for moving kinect sensor. In *Proceedings of the 2013 IEEE RO-MAN*, pages 320–321, August 2013.

[19] Y. Kondo, K. Takemura, J. Takamatsu, and T. Ogasawara. Body gesture classification based on bag-of-features in frequency domain of motion. In *Proceedings of the IEEE RO-MAN*, pages 386–391, Sept 2012.

[20] R.C. Luo, S. Chang, and Y Yang. Tracking with pointing gesture recognition for human-robot interaction. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 1220–1225, December 2011.

[21] C. Matuszek, L. Bo, L. Zettlemoyer, and D. Fox. Learning from unscripted deictic gesture and language for human-robot interactions. In *Proceedings of the 28th National Conference on Artificial Intelligence (AAAI)*, Québec City, Quebec, Canada, March 2014.

[22] J. Nagi, H. Ngo, A. Giusti, L. M. Gambardella, J. Schmidhuber, and G. A. Di Caro. Incremental learning using partial feedback for gesture-based human-swarm interaction. In *Proceedings of the 2012 IEEE RO-MAN*, pages 898–905, Sept 2012.

[23] K. Nickel and R. Stiefelhagen. Visual recognition of pointing gestures for human–robot interaction. *Image and Vision Computing*, 25(12):1875 – 1884, 2007. The age of human computer interaction.

[24] K. O'Brien, J. Sutherland, C. Rich, and C. L. Sidner. Collaboration with an autonomous humanoid robot: A little gesture goes a long way. In *6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 215–216, March 2011.

[25] K. Ogata. *Modern Control Engineering*. Instrumentation and controls series. Prentice Hall, 2010.

[26] G. I. Parisi, D. Jirak, and S. Wermter. Handsom - neural clustering of hand motion for gesture recognition in real time. In *Proceedings of the 2014 RO-MAN: The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 981–986, August 2014.

[27] C. Park and S. Lee. Real-time 3d pointing gesture recognition for mobile robots with cascade {HMM} and particle filter. *Image and Vision Computing*, 29(1):51 – 63, 2011.

[28] M. Pateraki, H. Baltzakis, and P. Trahanias. Visual estimation of pointed targets for robot guidance via fusion of face pose and hand orientation. *Computer Vision and Image Understanding*, 120(0):1 – 13, 2014.

[29] J. Puigbo, A. Pumarola, C. Angulo, and R. Tellez. Using a cognitive architecture for general purpose service robots control. *Connection Science*, 2015.

[30] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[31] J. L. Raheja, A. Chaudhary, and S. Maheshwari. Hand gesture pointing location detection. *Optik - International Journal for Light and Electron Optics*, 125(3):993 – 996, 2014.

[32] A. Ramey, V. González-Pacheco, and M. A. Salichs. Integration of a low-cost rgb-d sensor in a social robot for gesture recognition. In *Proceedings of the 6th International Conference on Human-robot Interaction*, HRI '11, pages 229–230, New York, NY, USA, 2011. ACM.

[33] M. Reyes, G. Dominguez, and S. Escalera. Feature weighting in Dynamic Time Warping for gesture recognition in depth data. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1182–1188, November 2011.

[34] L. D. Riek, T. Rabinowitch, P. Bremner, A. G. Pipe, M. Fraser, and P. Robinson. Cooperative gestures: Effective signaling for humanoid robots. In *Proceedings of the 2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 61–68, March 2010.

[35] R. B. Rusu. Clustering and segmentation. In *Semantic 3D Object Maps for Everyday Robot Manipulation*, volume 85 of *Springer Tracts in Advanced Robotics*, chapter 6, pages 75–85. Springer Berlin Heidelberg, 2013.

[36] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, February 1978.

[37] A. Sauppé and B. Mutlu. Robot deictics: How gesture and context shape referential communication. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 342–349, New York, NY, USA, 2014. ACM.

[38] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, and S. Fitzgibbon, A. amd Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the ACM CHI '15*, April 2015. To appear.

[39] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1297–1304, Washington, DC, USA, 2011. IEEE Computer Society.

[40] M. Sigalas, H. Baltzakis, and Trahanias. P. Temporal gesture recognition for human-robot interaction. In *Proceedings of Mutimodal Human-Robot Interfaces Workshop. IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, USA, May 2010.

[41] A. St. Clair, R. Mead, and M. J. Mataric. Investigating the effects of visual saliency on deictic gesture production by a humanoid robot. In *Proceedings of the 2011 IEEE RO-MAN*, pages 210–216, July 2011.

[42] E. A. Topp. *Human-Robot Interaction and Mapping with a Service Robot: Human Augmented Mapping*. PhD thesis, KTHKTH, Computer Vision and Active Perception, CVAP, Centre for Autonomous Systems, CAS, 2008. QC 20100914.

[43] H. Tran and T. Thanh. How can human communicate with robot by hand gesture? In *Proceedings of the 2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, pages 235–240, January 2013.

[44] M. Van den Bergh, D. Carton, R. de Nijs, N. Mitsou, C. Landsiedel, K. Kühnlenz, D. Wollherr, L. J. Van Gool, and M. Buss. Real-time 3d hand gesture interaction with a robot for understanding directions from humans. In Henrik I. Christensen, editor, *Proceedings of the 2011 IEEE RO-MAN*, pages 357–362. IEEE, 2011.

[45] S. Waldherr, R. Romero, and S. Thrun. A gesture based interface for human-robot interaction. *Autonomous Robots*, 9(2):151–173, 2000.

[46] D. Xu, X. Wu, Y. Chen, and Y. Xu. Online dynamic gesture recognition for human robot interaction. *Journal of Intelligent & Robotic Systems*, pages 1–14, 2014.

[47] X. Zhao, A. M. Naguib, and S. Lee. Kinect based calling gesture recognition for taking order service of elderly care robot. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication, 2014 RO-MAN*, pages 525–530, Aug 2014.

Glossary

API Application Program Interface. 12, 22

Deictic gesture A gesture that specifies identity or spatial location depending on the context of one or more of the participants in the communication act, which can be accompanied by a deictic utterance such as here, there, this or that. 6, 7, 21, 23

Differential drive robot A robot which has two wheels mounted on the left and right side of it, and driven independently. A passive wheel is usually added for stability purposes. 36

DTW Dynamic Time Warping. 4, 25, 26, 28–30, 55

Dynamic gesture A gesture which is defined by a movement the person performs with the whole body or a part of it. 3, 5, 22, 23, 26, 30, 44–46, 55

ESAII department Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial de la UPC (Department of Systems Engineering, Automatics and Industrial Informatics from UPC). 11

FME Facultat de Matemàtiques i Estadística (Mathematics and Statistics Faculty) from UPC. 46, 47

FSM Finite State Machine. 5, 6, 15, 16, 39, 65–67

GUI Graphical User Interface. 20, 21, 33

HMM Hidden Markov Model. 5, 6

HRI Human Robot Interaction. 1, 4, 5, 18, 46, 55

Humanoid robot A robot whose body shape resembles that of the human body. 1, 3, 11, 40, 55

LOOCV Leave-One-Out Cross-Validation. 45

MLP Multi-Layer Perceptron. 4

Odometry Method which provides an estimation of the position information of a robot or its change over time. 36–38, 57

OpenMP Open Multi-Processing. 22

OSRF Open Source Robotics Foundation. 14

PCL Point Cloud Library. 16, 20, 31, 33, 34, 55

PID Proportional-Integral-Derivative controller. 36

RANSAC RANdom SAmple Consensus. 32, 33

RGB Red-Green-Blue data channel information from a color image. 4, 5, 20, 56

ROS Robot Operating System. 6, 14–16, 18, 36, 37

RPC Remote Procedure Call. 15

SD Standard Deviation. 51

SDGRA Static and Dynamic Gesture Recognition Algorithm. 30, 32

SDK Software Development Kit. 10, 23, 24

SLAM Simultaneous Localization And Mapping. 57

SMACH State MACHine. 15, 16, 40

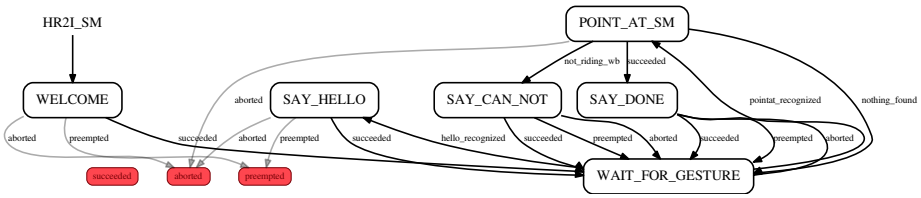
SOM Self Organizing Maps. 4

Static gesture A gesture in which the person does not move neither the whole body nor some limbs but rather stays still in a specific position for some short period of time. 3, 22, 23, 30, 44–46, 52, 55

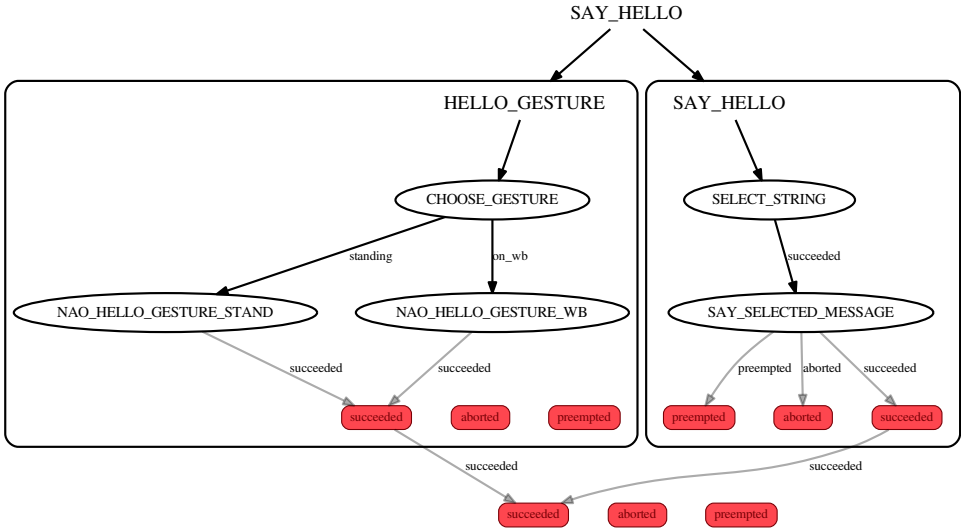
UPC Universitat Politècnica de Catalunya (Polytechnic University of Catalonia). 11, 63

A State Machine diagrams

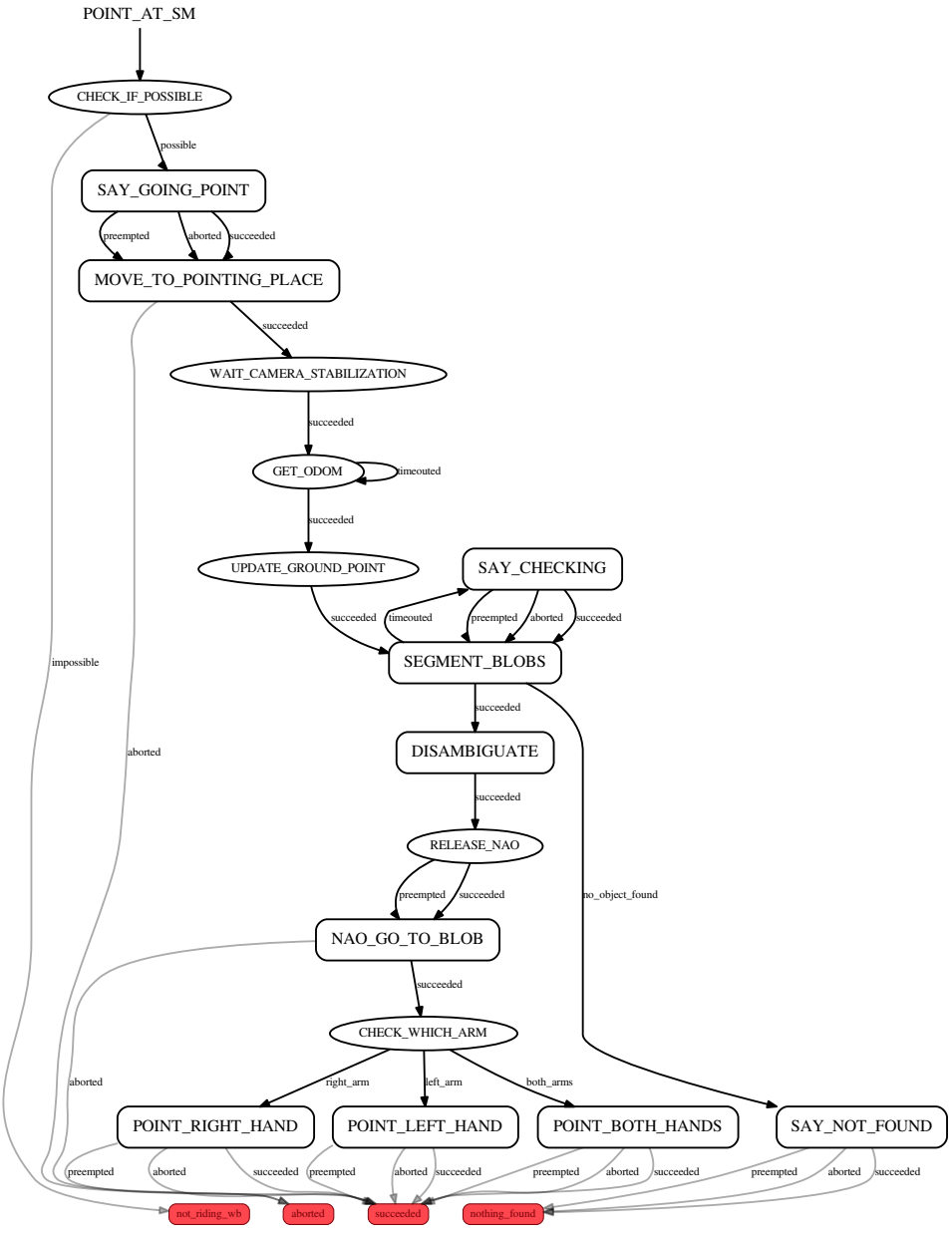
A.1 Main Finite State Machine



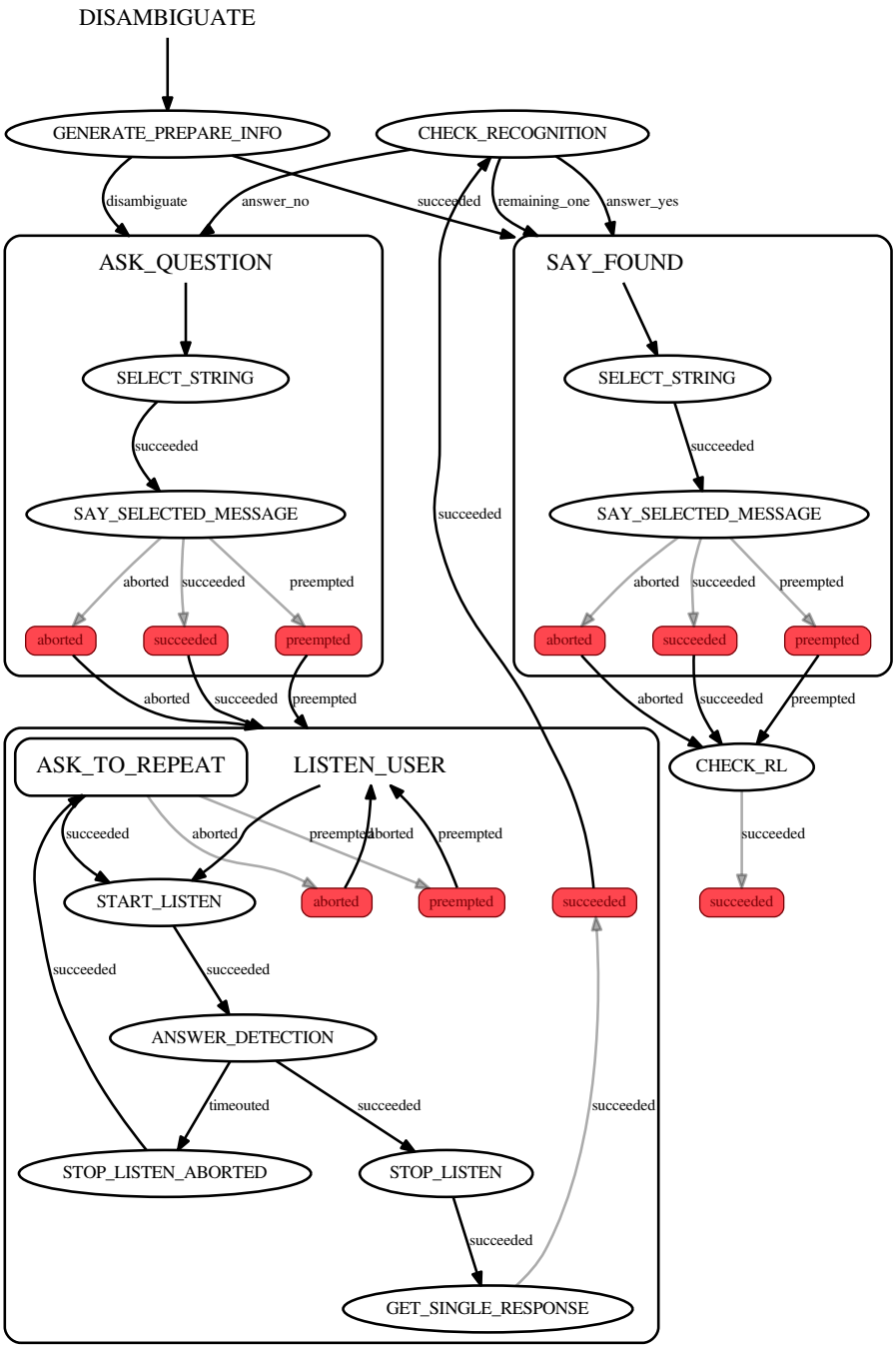
A.2 Wave response sub-FSM



A.3 Point At response sub-FSM



A.4 Disambiguate object sub-FSM



B User tests questionnaire

Please, answer the questions below about the interaction test you have just performed. This is an anonymous form, so feel free to be sincere.

*Questions marked with * are compulsory.*

Demographic data

1. **How old are you?***
2. **And you are...?***
   - ⋆ A man
   - ⋆ A woman
3. **Robot familiarity***
   From 1 to 5, how used to robots are you? (1 - It's the first time I see one, 5 - I live with them around)

	1	2	3	4	5	
Never	○	○	○	○	○	Usually

4. **Studies background***
   - ⋆ Secondary School
   - ⋆ High School
   - ⋆ University
   - ⋆ Professional training
5. **If you chose University or Professional training, which is its major?**
6. **Are you currently studying?***
   - ⋆ Yes
   - ⋆ No
7. **If your answer was yes, what are you studying?**

About the Wave gesture

8. **The Wave gesture was...***
   - ⋆ Natural
   - ⋆ Unnatural
   - ⋆ Hard to perform
9. **NAO's response was...***
   1 - NAO didn't answer the wave, 5 - NAO's response was fast

	1	2	3	4	5	
Non existant	○	○	○	○	○	Fast

10. **Was the response the one you expected?***
    - ⋆ Yes
    - ⋆ No
11. **If you answered no, why?**
12. **Any suggestion for the wave gesture?**

The Point At gesture

13. **The Point At gesture was...***
    - ⋆ Natural
    - ⋆ Unnatural
    - ⋆ Hard to perform
14. **NAO's response was...***
    1 - NAO didn't move, 5 - NAO's response was fast

	1	2	3	4	5	
Non existant	○	○	○	○	○	Fast

15. Do you think you had to point to the object for too much time?*

- ⋆ Yes
- ⋆ No

16. In general, did NAO figure out which object you pointed at?*

	1	2	3	4	
Never	○	○	○	○	Always

17. Did the robot clearly show which object did HE THINK you pointed at?*

Even though he may have mistaken, did he clearly show what he thought?

	1	2	3	4	
Never	○	○	○	○	Always

18. If the robot asked you about which object was, did you understand his question?

- ⋆ Yes
- ⋆ No

19. If he asked you a question, did he understand your answer?

- ⋆ Yes
- ⋆ No
- ⋆ I don't know

20. And if the robot asked you a question, was it easy to answer?

- ⋆ Yes
- ⋆ Yes, but I had to repeat the answer a lot of times
- ⋆ No

21. Was NAO's response the one you expected?*

- ⋆ Yes
- ⋆ No

22. In case of answering NO, why?

23. Any suggestion for the Point At gesture?

About the test in general...

24. How natural has the whole interaction been?*
Tell me your opinion

	1	2	3	4	5	
Very unnatural	○	○	○	○	○	Quite natural

25. Did you clearly understand robot's indications?*

- ⋆ Yes
- ⋆ No

26. Did you feel frustrated at any moment?*

- ⋆ Yes
- ⋆ No

27. In case of answering yes, why?

28. Did you feel like you didn't know what was happening at any moment?*

- ⋆ Yes
- ⋆ No

29. In case of answering yes, at which moment?

30. Was it easy to interact with the NAO and make him do a task?*

- ⋆ Yes
- ⋆ No

31. In case of NO, why?

32. Was the interaction result satisfactory?*

- ⋆ Yes
- ⋆ No

33. Did you enjoy the test?*

- ⋆ Yes
- ⋆ No

About the future...

34. Would it have been better to answer the Yes/No with a head movement?*

- ⋆ Yes
- ⋆ No

35. Would it have been easier?*

- ⋆ Yes
- ⋆ No

36. Do you think it would be better to be able to answer both with voice and head movements?*

- ⋆ Yes
- ⋆ No

To end with...

37. Do you think an application like this would be useful for a household environment?*

- ⋆ Yes
- ⋆ No

38. Any gesture you can imagine which may be useful to interact with the NAO?

39. Any last comment, suggestion or constructive critic?