



Technische Universität Berlin



# **Gesture Recognition for Human-Robot Interaction: An approach based on skeletal points tracking using depth camera**

## **Masterarbeit**

am Fachgebiet Agententechnologien in betrieblichen Anwendungen und der  
Telekommunikation (AOT)

Prof. Dr.-Ing. habil. Sahin Albayrak  
Fakultät IV Elektrotechnik und Informatik  
Technische Universität Berlin

vorgelegt von

**Sivalingam Panchadcharam Aravinth**

Betreuer: Prof. Dr.-Ing. habil. Sahin Albayrak,  
Dr.-Ing. Yuan Xu

Sivalingam Panchadcharam Aravinth  
Matrikelnummer: 342899  
Sparrstr. 9  
13353 Berlin

# Statement of Authorship

I declare that I have used no other sources and aids other than those indicated. All passages quoted from publications or paraphrased from these sources are indicated as such, i.e. cited and/or attributed. This thesis was not submitted in any form for another degree or diploma at any university or other institution of tertiary education

Place, Date

Signature

# Abstract

Human-robot interaction (HRI) has been a topic of both science fiction and academic speculation even before any robots existed [?]. HRI research is focusing to build an intuitive and easy communication with the robot through speech, gestures, and facial expressions. The use of hand gestures provides an attractive alternative to complex interfaced devices for HRI. In particular, visual interpretation of hand gestures can help in achieving the ease and naturalness desired for HRI. This has motivated a very active research concerned with computer vision-based analysis and interpretation of hand gestures. Important differences in the gesture interpretation approaches arise depending on whether 3D based model or appearance based model of the gesture is used [?].

In this thesis, we attempt to implement the hand gesture recognition for robots with modeling, training, analyzing and recognizing gestures based on computer vision and machine learning techniques. Additionally, 3D based gesture modeling with skeletal points tracking will be used. As a result, on the one side, gestures will be used command the robot to execute certain actions and on the other side, gestures will be translated and spoken out by the robot.

We further hope to provide a platform to integrate Sign Language Translation to assist people with hearing and speech disabilities. However, further implementations and training data are needed to use this platform as a full fledged Sign Language Translator.

## Keywords

Human-Robot Interaction (HRI), Computer Vision, Depth Camera, Hand Gesture, 3D hand based model, Skeleton tracking, Gesture Recognition, Sign Language Translation, Hidden Markov Model, NAO

# Acknowledgements

Der Punkt Acknowledgements erlaubt es, persönliche Worte festzuhalten, wie etwa:

- Für die immer freundliche Unterstützung bei der Anfertigung dieser Arbeit danke ich insbesondere...
- Hiermit danke ich den Verfassern dieser Vorlage, für Ihre unendlichen Bemühungen, mich und meine Arbeit zu fördern.
- Ich widme diese Arbeit

Die Acknowledgements sollte stets mit großer Sorgfalt formuliert werden. Sehr leicht kann hier viel Porzellan zerschlagen werden. Wichtige Punkte sind die vollständige Erwähnung aller wichtigen Helfer sowie das Einhalten der Reihenfolge Ihrer Wichtigkeit. Das Fehlen bzw. die Hintanstellung von Personen drückt einen scharfen Tadel aus (und sollte vermieden werden).

# Contents

<b>Statement of Authorship</b>	<b>II</b>
<b>Abstract</b>	<b>III</b>
<b>Contents</b>	<b>V</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Goal</b>	<b>2</b>
<b>3 Solution</b>	<b>4</b>
3.1 Experimental Designs . . . . .	4
3.1.1 Everything On-Board . . . . .	4
3.1.2 Extending NAO with Single Board Computer . . . . .	5
3.1.3 Everything Off-Board . . . . .	5
3.2 Implementation . . . . .	6
3.3 Gesture Recognition . . . . .	6
3.3.1 Hand Gestures Modeling . . . . .	7
3.3.2 Training . . . . .	9
3.3.3 Prediction . . . . .	10
3.4 Human-Robot Interaction . . . . .	11
3.4.1 Gesture-to-Speech . . . . .	13
3.4.2 Gesture-to-Motion . . . . .	13
3.4.3 Gesture-to-Gesture . . . . .	15
3.5 Toolchain . . . . .	15
3.5.1 Develop . . . . .	15
3.5.2 Build . . . . .	16
3.5.3 Patch . . . . .	17
3.5.4 Deploy . . . . .	17

<i>CONTENTS</i>	VI
3.5.5 Version Control . . . . .	17
3.5.6 Data Analysis . . . . .	18
3.5.7 Documentation . . . . .	18
<b>Bibliography</b>	<b>19</b>
<b>List of Figures</b>	<b>20</b>
<b>List of Tables</b>	<b>21</b>
<b>Abbreviations</b>	<b>22</b>

# Chapter 1

## Introduction

Huge influence of computers in society has made smart devices, an important part of our lives. Availability and affordability of such devices motivated us to use them in our day-to-day living. The list of smart devices includes personal automatic and semi-automatic robots which are also playing a major role in our household. For an instance, Roomba [?] is an autonomous robotic vacuum cleaners that automatically cleans the floor and goes to its charging station without human interaction.

Interaction with smart devices has still been mostly through displays, keyboards, mouse and touch interfaces. These devices have grown to be familiar but inherently limit the speed and naturalness with which we can interact with the computer. Usage of robots for domestic and industrial purposes has been continuously increasing. Thus in recent years, there has been a tremendous push in research toward an intuitive and easy communication with the robot through speech, gestures and facial expressions.

Tremendous progress had been made in speech recognition and several commercially successful speech interfaces are available. However, speech recognition systems have certain limitations such as misinterpretation due to various accents and background noise interference. It may not be able to differentiate between your speech, other people talking and other ambient noise, leading to transcription mix-ups and errors.

Furthermore, there has been an increased interest in recent years in trying to introduce other human-to-human communication modalities into HRI. This includes a class of techniques based on the movement of the human arm and hand, or hand gestures. The use of hand gestures provides an attractive alternative for Human-robot interaction than the conventional cumbersome devices.

# Chapter 2

## Goal

As described earlier, HRI research is focusing to build an intuitive and easy communication with the robot through speech, gestures and facial expressions. The use of hand gestures provides the ease and naturalness with which the user can interact with robots.

In this thesis, we attempt to implement the feature for NAO to recognize gestures and execute predefined actions based on the gesture. NAO will be extended with an external depth camera, that will enable NAO to recognize 3D modeled gestures. This 3D camera will be mounted on the head of NAO, so that it can scan for gestures in the horizon. Additionally, skeletal points tracking algorithm with machine learning technique using Hidden Markov Models will be used to recognize the gestures. Due to the computational limitations of NAO, gesture recognition algorithm will be executed on off-board computer. With the hand gesture recognizing feature, NAO will be available to the users in two modes.

- **Command mode:** In this mode, a gesture will be recognized by NAO and related task will be executed. Even though the gesture based interaction is real time, NAO can not be interrupted or stopped by using any gesture while it is executing a task. However, other interfaces such as voice commands can be used in such situation to stop or interrupt the ongoing task execution.
- **Translation mode:** In this mode, NAO will be directly translating the meaning of the gesticulated gestures. To achieve this, text-to-speech library of NAO will be used and recognized gesture can be spoken out using the integrated loudspeaker. In future, it will allow NAO to translate a sign language to assist people with hearing and speech disabilities.



In this thesis, we planned to train NAO with few very simple gestures due to the reason that NAO has computational limitations. Gestures will involve both the hands or single hand to interact with the robot.

# Chapter 3

## Solution

To build an effective and easy to use hand gesture recognition system for NAO, various tools and technologies were studied during this thesis. Figure 3.1 shows the individual components which are essential parts of this thesis in implementing the goal. The main challenge is to find a solution that can integrate all these components into a robust system. However, due to the computational and compatibility limitations of NAO, we have faced problems in implementing few contemplated solutions which are described in the next section. Finally, the successful solution in achieving the goal will be discussed in the following sections.

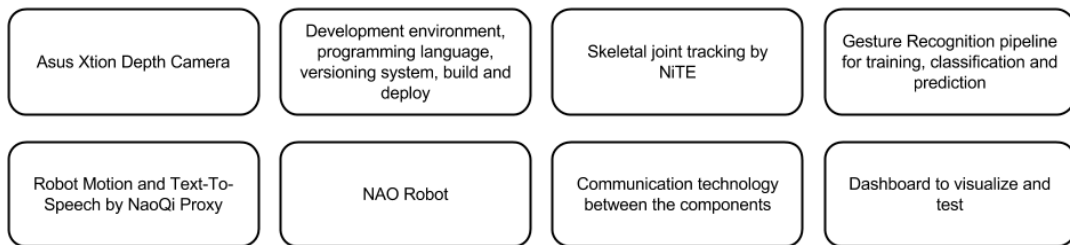


Figure 3.1: Component

## 3.1 Experimental Designs

### 3.1.1 Everything On-Board

First experiment design was conceived in a way that depth camera, skeletal joint tracking, gesture recognition infrastructure and robot motion will be embedded into the on-board computer of NAO. However, gesture recognition infrastructure is composed

of computationally intensive machine learning processes and along with skeletal joint tracking by NiTE had pushed NAO to 100 % CPU load consistently.

### 3.1.2 Extending NAO with Single Board Computer

In order to escape the computational limitation of NAO, another experimental design was contemplated, that the robot will be extended as shown in the figure 3.2 with a powerful Single Board Computer such as pcDuino or RaspberryPi. However, Asus Xtions higher power consumption of 2.5 Watts with weight of 250 grams, pcDuinos power consumption of 2A at 5VDC with weight of 100 grams and additional weight by 3D printed mounts, heat sinks and wires will make NAO to be heavier and ultimately result in poor motion performances and higher power consumption.

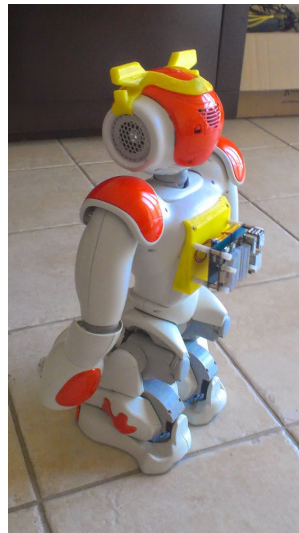


Figure 3.2: NAO Bag

### 3.1.3 Everything Off-Board

This experimental design pushes all the components to an off-board computer that could be a PC connected with depth camera at a fixed location. User will gesticulate in front of the camera and all processing will be done on PC. Finally predicted gesture will be transformed into a motion and voice, and it will be sent to NAO via Aldebaran proxies using WLAN. This design completely decouples the robot from other components and degrades the natural interaction between human and the robot. However, this design will suit for other applications such as indoor navigation and localization of NAO.

## 3.2 Implementation

After analyzing the disadvantages of other experimental designs, the final design was chosen to build an efficient real-time hand gesture recognition for human-robot interaction using skeletal points. Figure 3.3 shows the architecture of the solution that was implemented during this thesis by grouping many components into 4 different modules which serve various purposes. Each module is implemented in different environment as shown in the figure and they communicate with one another to complete the data flow. All these modules use a common configuration file that contains information such as port number, host name and log path.

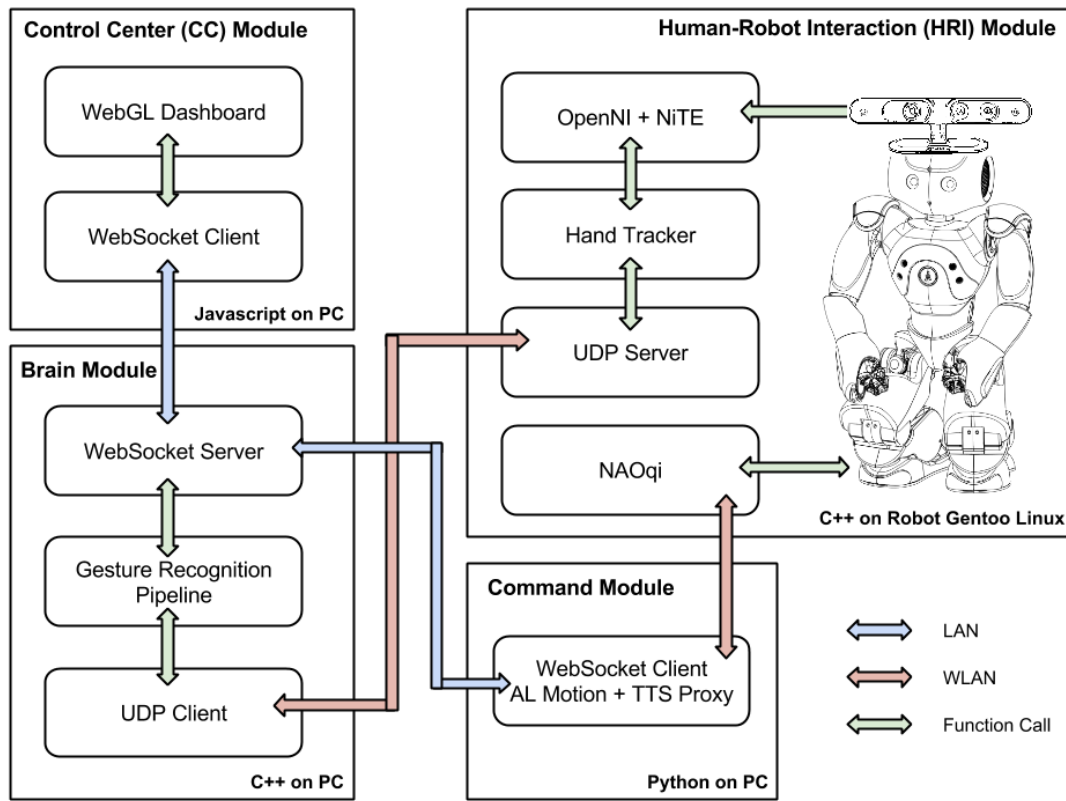


Figure 3.3: HRI Architecture

## 3.3 Gesture Recognition

Above sections described the necessary tools that are implemented to execute a real time hand gesture recognition system. In this thesis we have decided to train the system with static gestures. However, the system can be easily extended to recognize temporal

gestures with the flexibility of Gesture Recognition Toolkit (GRT). Initially a set of simple gestures are chosen and the training data is collected for all those gestures.

### 3.3.1 Hand Gestures Modeling

In this thesis we have modeled five static hand gestures involving both the hands of the user. These are communicative hand gestures and they symbolize few referential action. Apart from Sign Language used by people with speech disability, various hand gestures are being used by humans in their day to day living. Figure 3.4 shows the hand signals used by different personnels in wide variety of application.





Traffic	Military	Sports	Construction
			
<b>Cyclist : Left Turn</b>	<b>Army : Understood</b>	<b>Football : Timeout</b>	<b>Mobile Crane : Hoist</b>

Figure 3.4: Hand Signals

This thesis focuses on hand gesture recognition to felicitate Human-Robot interactions. One greater application using hand gestures for robots is commanding the robot to move to another position. Additionally it could translate the gestures to spoken words to help people with speech disability.

Therefore, we have chosen five simple static gestures as shown in the figure 3.3.1 which are conceptualized by the traffic police hand signals. All the gestures are modeled to the direction of the user and they will be understood as mirrored gestures. For example, left side of the user will be right side to the robot that is facing the user. Additionally our system makes use of two dynamic gestures of NiTE which are used as focus gestures to gain control or start hand tracking.

**Turn Left** It is gesticulated as shown in the figure 3.5 by holding the right hand up and left hand wide open. It refers to an action that turn to left and stay in position.



Figure 3.5: Turn Left Gesture



Figure 3.6: Turn Right Gesture

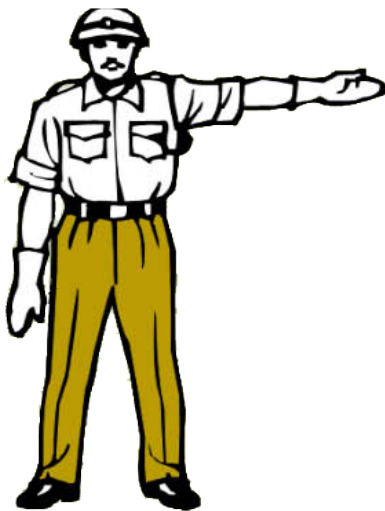


Figure 3.7: Move Left Gesture

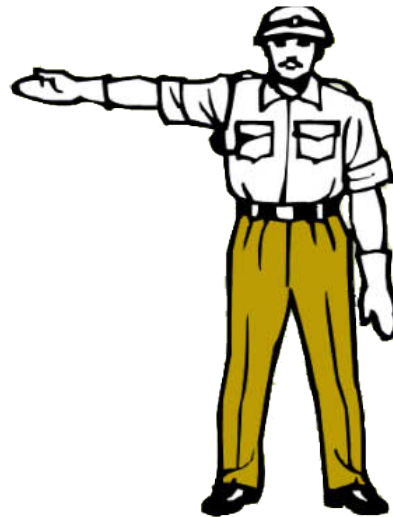


Figure 3.8: Move Right Gesture



Figure 3.9: Walk Gesture

**Turn Right** It is gesticulated as shown in the figure 3.6 by holding the left hand up and right hand wide open. It refers to an action that turn to right and stay in position.

**Move Left** It is gesticulated as shown in the figure 3.7 by holding the right hand down and left hand wide open. It refers to an action that turn to left and keep moving in the forward direction.

**Move Right** It is gesticulated as shown in the figure 3.7 by holding the left hand down and right hand wide open. It refers to an action that turn to right and keep moving in the forward direction.

**Walk** It is gesticulated as shown in the figure 3.7 by holding the left and right hand up. It refers to an action that keep moving in the forward direction.

**Wave** It is gesticulated by holding a hand up and move it several times from left to right and back. This gesture is executed to initiate hand tracking, if no hands are tracked or tracked hand is been lost.

**Click** It is gesticulated by holding a hand up, push the hand towards the sensor then immediately pull the hand backwards.

### 3.3.2 Training

Our gesture recognition pipeline is configured to have 15 seconds preparation time and 20 seconds recording time with 6 dimensional input of both left and right hands at positions x, y and z in the Cartesian coordinates. Depth camera is at the origin of the coordinate system as shown in the figure 3.10.

Brain is set to training mode and CC is started to visualize the hand positions in order to align the trainer during the preparation time. Each gesture is isolated in time and gesticulated for 20 seconds. Samples are added to the training dataset and when the timer stopped the recording, Brain asked the trainer to train the same class again or another. Every gesture was assigned a class label from 1 to 5 and the mapping of class label to hand gesture is stored in a configuration file named signs.json.

**Minimum and Maximum Distance Training** If the gestures are gesticulated with only one person at a static position in space in front of the camera, then the recognition algorithm would not recognize the same gesture gesticulated by another person or the

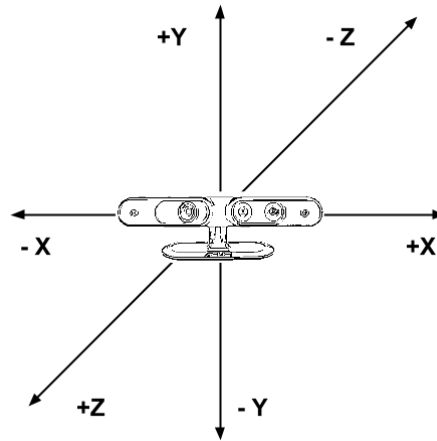


Figure 3.10: Depth Camera origin

same person in different position. In order to scale the range of recognition, every gesture was gesticulated in 4 different positions as shown in the plot 3.11 and in all possible combinations that hands are kept wider or narrower as shown in the plots 3.3.2. Therefore, each gesture in the training data is recorded in 4 positions with each for 20 seconds at 30 samples per second created 2400 samples per gesture.

ANBC is an iterative learning algorithm that improves the classification accuracy with increase in positive training data. Plot 3.3.2 shows that the trained data makes our gesture recognition system to detect gestures at the minimum distance from 1700 mm to the maximum distance 2500 mm away from the sensor and 800 mm left or right to the sensor. If the user leaves this field of view, the hand tracking algorithm will lose the hand or gesture will fall in the Null Rejection region of the classifier.

**Training Data** Once all the gestures are recorded, they replayed using CC to find out, if there is any false samples are added to the training data. Such false data leads to an incorrect model that will ultimately affect the prediction performance. Such samples are removed from the training data and a final dataset with all 5 classes are stored as hri-training-dataset.txt. Additionally, some test data for each gesture is recorded in order to evaluation the accuracy of the recognition system. Furthermore, a set of non-gesture dataset was recorded in order to test the Null Rejection accuracy of the classifier.

### 3.3.3 Prediction

After successfully collecting the training data for all the gestures, Brain is set to prediction mode where the pipeline is trained. HRI module starts to track the user's hand,



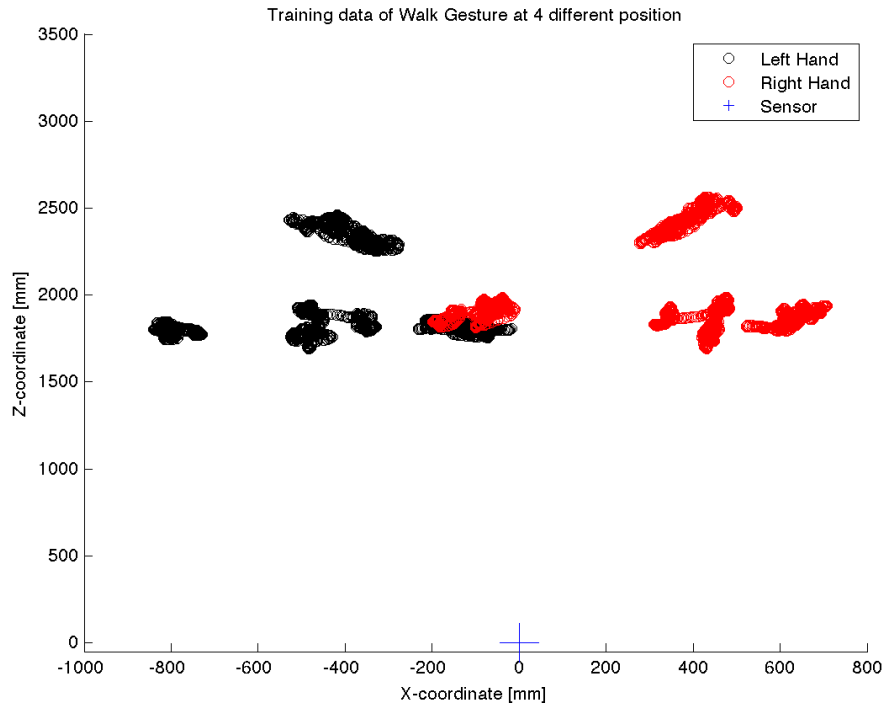
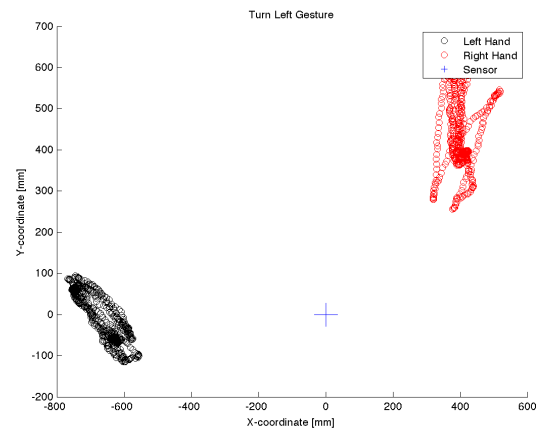
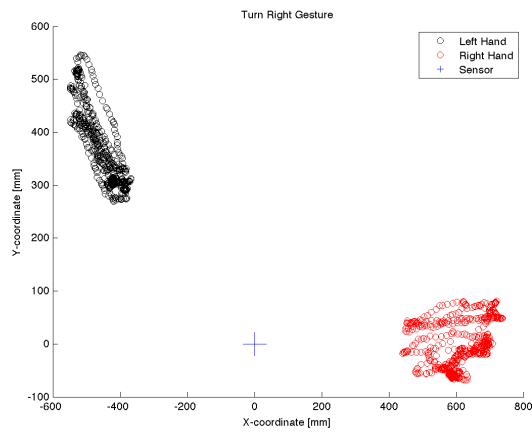


Figure 3.11: Walk Gesture in 4 different positions

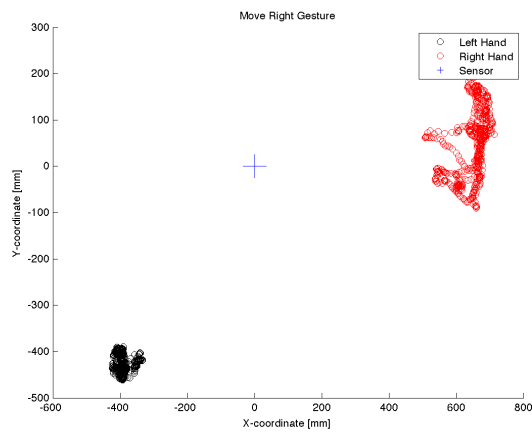
Brain predicts a gesture when both the hands are present in the input sample. Figure 3.12 shows Control Center where prediction output for every sample with maximum likelihood is displayed all the time. The predicted gesture is triggered only after it was gesticulated for more than one second.

### 3.4 Human-Robot Interaction

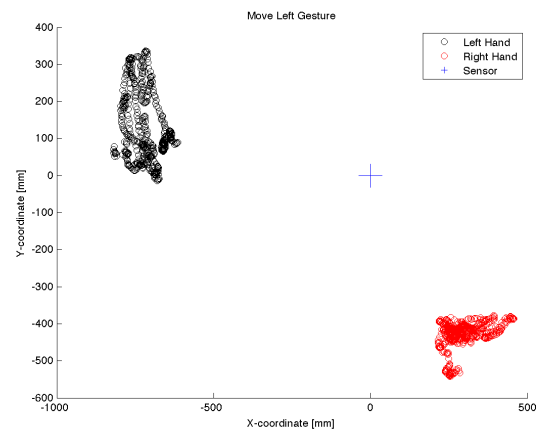
Fundamental goal of this thesis to build a systematic hand gesture recognition system to interact with machines such as robot or a computer. Interaction with them are mostly through displays, keyboards, mouse and touch interfaces. These devices have grown to be familiar but inherently limit the speed and naturalness. Previous sections have explained how we build a system to facilitate a natural interaction with the humanoid robot called NAO. Following sections illustrate how robot reacts to the hand gestures in real time with the help of Command module.



Turn Left Gesture

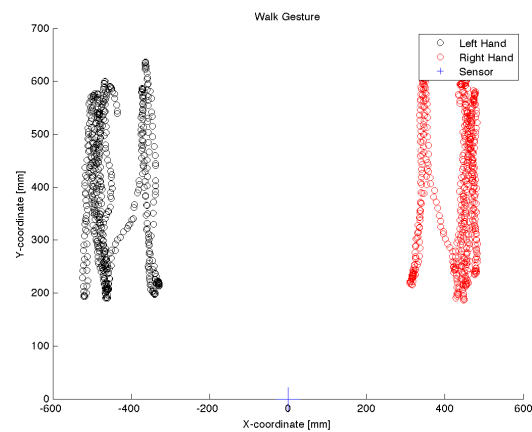


Turn Right Gesture



Move Left Gesture

Move Right Gesture



Walk Gesture

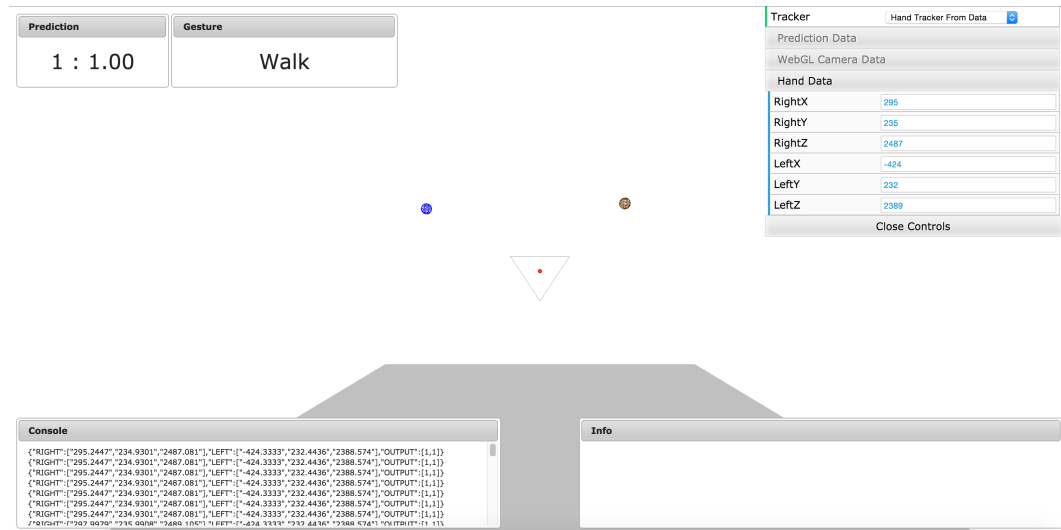


Figure 3.12: Walk Gesture

### 3.4.1 Gesture-to-Speech

Easiest translation from the recognized hand gesture is to speak it out loud. We have used Text-To-Speech (TTS) engine that was built internally inside Aldebaran modules. When the user gesticulate the focus gesture, NAO says "WAVE" and denoting that hand tracking is started. Furthermore, the robot says words such as "Walk", "Turn Left", "Turn Right", "Move Left" and "Move Right", whenever those gestures are recognized. Additionally, it says info messages such as "Left Hand is lost", "Right Hand is lost" and "Both hands are lost" to inform the user about the internal status of hand tracking.

### 3.4.2 Gesture-to-Motion

This thesis was initially conceived as a hand gesture translator just to say the recognized gestures loud. To make this system more useful, Gesture-to-Motion feature was added to the Command module. This functionality helps us to move the robot from one position to another in 2 dimensional space. Therefore each gesture was assigned a locomotion task as follows:

**Walk** This gesture commands the robot to walk in forward direction at a normalized maximum velocity (1.0) with the step frequency of 0.5. Robot walks approximately for 5 seconds and waits for the next command.

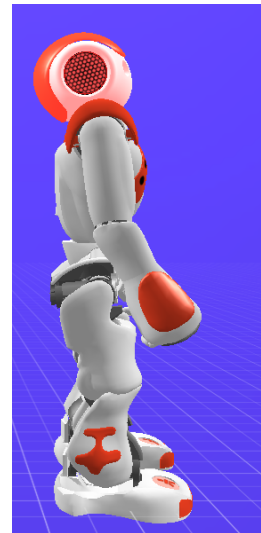
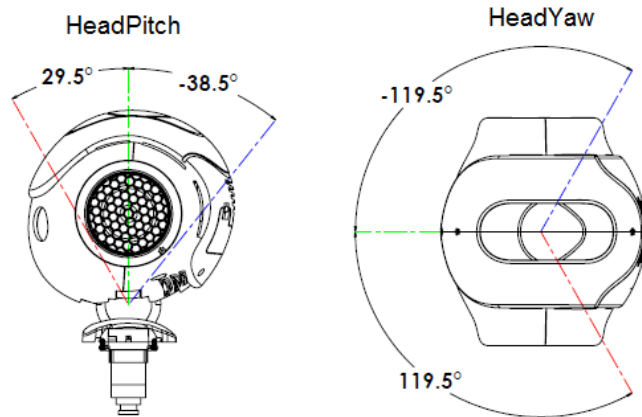


Figure 3.13: NAOs head Pitch and Yaw angle range      Figure 3.14: NAO with tilted head to look at the user

**Turn Left** This gesture commands the robot to rotate itself around z-axis in the left direction at a normalized maximum velocity (1.0) with the step frequency of 0.5. Robot walks approximately for 3 seconds and waits for the next command.

**Turn Right** This gesture commands the robot to rotate itself around z-axis in the right direction at a normalized maximum velocity (1.0) with the step frequency of 0.5. Robot walks approximately for 3 seconds and waits for the next command.

**Move Left** This gesture combines Walk and Turn Left by commanding the robot to rotate itself around z-axis in the left direction for 3 seconds and walk forward for 5 seconds and waits for the next command.

**Move Right** This gesture combines Walk and Turn Right by commanding the robot to rotate itself around z-axis in the right direction for 3 seconds and walk forward for 5 seconds and waits for the next command.

**Click** This gesture is used to gain the control of the robot, when the robot lost its balance and fell down. When this gesture is executed, robot wakes up from the sleeping mode and sets itself to the standing position.

### Head Position

As described in the section 3.3.2, collection of training data for each gesture was carried out in 4 different positions in front of the robot. During this phase robot was set to standing position where the height of the robot is 58 cm. Figure 3.13 shows that NAOs head can be tilted by adjusting the pitch and yaw of the head joint. In order to avoid confusing camera perspective during the training, NAOs head pitch was set to -18.0 degrees and yaw was set to 0.0 degree as shown in the figure 3.14. At this angle, field of view of the depth camera was enough to cover upper body of the user.

However, keeping the head tilted with mounted camera will cause the robot to lose balance. Therefore, NAOs head position is reset to initial stand position before it executes the received Gesture-to-Motion command. Once the locomotion phase is completed it looks back at the user. This functionality greatly improves in locating the user at any position in the Minimum-Maximum range as show in the plot 3.11.

### 3.4.3 Gesture-to-Gesture

Apart from offering the essential functionalities, Command module also provides Gesture-to-Gesture translation where NAO will be imitating hand gestures of the user. Shoulder Roll and Pitch, Shoulder Roll and Yaw angles were measured by manually by positioning NAO for every gesture. When a gesture is detected, the Command modules sets the predefined angles to the shoulder and elbow joints of both the hands of NAO, therefore, translating the human hand gesture to a robotic hand gesture.

## 3.5 Toolchain

During the implementation of this thesis, many tools are used for various purposes. Every module in this thesis uses different programming language, therefore, different toolchains are used. Following sections talk about the tools that are used to develop, build, deploy and document this thesis work.

### 3.5.1 Develop

**Xcode** Core functionalities of this thesis are developed in C++ on Mac OSX. Therefore, Xcode is used to develop HRI and Brain module. Xcode is an IDE containing a suite of software development tools developed by Apple for developing software for OS X and iOS.

**WebStorm** Control Center module is developed in Javascript with the help of a popular IDE for Web development called as WebStorm. It is a commercial IDE for JavaScript, CSS and HTML built on JetBrains IntelliJ IDEA platform.

**PyCharm** Command module is developed in Python using an IDE name as PyCharm. It is implemented by a company called JetBrains and it provides code analysis, a graphical debugger, an integrated unit tester and supports web development with Django.

### 3.5.2 Build

**Javascript and Python** They are traditionally implemented as interpreted languages and therefore they do not need any special compilers to build them. Control Center module needs just a latest browser with WebSocket and WebGL support to run the Javascript code. Python binary is available in most modern operating systems and we used Python version 2.7.6 to run Command module.

**C++** The code that was implemented in C++ used 2 different compilers to build it, because the development is done on 64-bit Mac operating system and target system is a 32-bit Gentoo Linux operating system.

**Clang and Xcode** Development code was built using Clang with LLVM libc++ Standard library. Clang is a compiler developed by Apple for C, C++, Objective-C and Objective-C++ programming languages. Build settings such as header, library search paths, macros, environment variables and linking are configured using Xcode.

**GCC and Cmake** Production code was built using GNU Compiler Collection (GCC) with libstdc++ GNU++11 Standard library. It is a compiler system produced by the GNU Project supporting various programming languages such as C, C++, Objective-C, Objective-C++, Fortran, Java, Ada, and Go. Build settings such as header, library search paths, macros, environment variables and linking are configured using Cmake. CMake is cross-platform free and open-source software for managing the build process of software using a compiler-independent method. The HRI module code was copied to OpenNAO and built using GCC and then copied to real NAO.

### 3.5.3 Patch

**OpenNAO / NAO OS** Aldebaran provides an image of the NAOs operating system named as OpenNAO to use the robotic system virtually and it can run in a virtual machine in any host. OpenNAO is 32-bit Gentoo Linux modified by Aldebaran for Intel Atom Processor with i686 Architecture.

**Emerge** Emerge is the package manager for Gentoo Linux and Portage is the package tree. Aldebaran forces users not to update Emerge to avoid conflict with Aldebaran modules. Portage tree used with NAO OS was last updated on 11 Jan 2012.

**GCC 4.5.3** Due to outdated packages on OpenNAO, GCC version is 4.5.3 and C++ library version is libstdc++.so.6.0.14. NiTE middleware library (libNiTE2.so) was built by PrimeSense using higher version of GCC (higher than GLIBCXX\_3.4.14 or libstdc++.so.6.0.14). Therefore, building HRI module on OpenNAO will throw an error while linking libNiTE2.so, 'usr/lib/libstdc++.so.6: version GLIBCXX\_3.4.15 not found'

This issue was solved by finding higher version of libstdc++ from debian repository for 32-bit architecture and copying that to /usr/lib folder on OpenNAO/NAO OS and linking current libstdc++ to the copied version. Repository of our thesis contains the required version of libstdc++. Following shell commands show how it must be patched on OpenNAO or NAO OS to run HRI module without any errors.

---

```
# On OpenNAO/NAO OS, execute the following to apply the patch
cd /source/human-robot-interaction
sudo cp lib/libstdc++.so.6.0.16 /usr/lib
sudo rm libstdc++.so
sudo ln -s libstdc++.so.6.0.16 libstdc++.so

# This command should show versions upto GLIBCXX_3.4.16
strings /usr/lib/libstdc++.so.6 | grep GLIBC
```

---

### 3.5.4 Deploy

### 3.5.5 Version Control

Proposal to final results of a project goes through many iterations or modification of the source files. Such changes are intentional and sometimes they are acciden-

tal. Therefore, source files of this thesis are stored and tracked using a version control system called Git. Git is a version control system that records changes to a file or set of files over time. To avoid accidental loss of source files, free git online service such as GitHub allows us to store them in a remote repository that can be cloned from anywhere via Internet. <https://github.com/AravindhPanch/gesture-recognition-for-human-robot-interaction> is the link to the online repository that contains all the informations regarding this thesis.

### **3.5.6 Data Analysis**

During this thesis work, significant amount of data are collected. These data consist of training and test data of full body skeletal points / hand joints which are recorded to train and test the hand gesture recognition system. This information must be analyzed and studied for the purpose of statistically understand this system. Additionally, these data must be plotted as shown in the graph 3.3.2 and hence, MATLAB is used. MATLAB is a multi-paradigm numerical computing environment developed by MathWorks to do matrix manipulations, plotting of functions and data and implementation of algorithms.

### **3.5.7 Documentation**

Proper documentation is a crucial part of any research work. Therefore, we have used LaTeX to document this thesis. LaTeX is a high-quality typesetting system that includes features designed for the production of technical and scientific documentation. TeXstudio on Mac OSX is the editor that was used to compose and manage the LaTeX documents of this. LaTeX source files can be found inside the document folder of the git repository of this thesis.



# **Bibliography**

# List of Figures

3.1	Component . . . . .	4
3.2	NAO Bag . . . . .	5
3.3	HRI Architecture . . . . .	6
3.4	Hand Signals . . . . .	7
3.5	Turn Left Gesture . . . . .	8
3.6	Turn Right Gesture . . . . .	8
3.7	Move Left Gesture . . . . .	8
3.8	Move Right Gesture . . . . .	8
3.9	Walk Gesture . . . . .	8
3.10	Depth Camera origin . . . . .	10
3.11	Walk Gesture in 4 different positions . . . . .	11
3.12	Walk Gesture . . . . .	13
3.13	NAOs head Pitch and Yaw angle range . . . . .	14
3.14	NAO with tilted head to look at the user . . . . .	14

# List of Tables

# Abbreviations

<b>HRI</b>	Human-Robot Interaction
<b>OpenNI</b>	Open Natural Interaction
<b>NiTE</b>	Natural Interaction Technology for End-user
<b>GRT</b>	Gesture Recognition Toolkit
<b>CC</b>	Control Center
<b>UDP</b>	User Datagram Protocol
<b>WLAN</b>	Wireless Local Area Network
<b>FOV</b>	Field Of View
<b>JSON</b>	JavaScript Object Notation
<b>DOF</b>	Degrees Of Freedom
<b>TTS</b>	Text-To-Speech
<b>API</b>	Application Program Interface
<b>DP</b>	Dynamic Programming
<b>MAP</b>	Maximum A Posterior Probability
<b>CSV</b>	Comma Separated Values
<b>DTW</b>	Dynamic Time Warping
<b>HMM</b>	Hidden Markov Models
<b>KNN</b>	K-Nearest Neighbor
<b>SVM</b>	Support Vector Machines
<b>PCA</b>	Principal Component Analysis
<b>GUI</b>	Graphical User Interface
<b>IDE</b>	Integrated Development Environment