Shujun Liu

A HUMAN-FOLLOWING NAO ROBOT USING PYTHON PROGRAMMING LANGUAGE

Information Technology 2019

FOREWORD

Three years of my Bachelor period have already passed in the Information Technology department at Vaasan Ammattikorkeakoulu, Vaasa University of Applied Sciences, and this thesis is my final project.

First of all, I would like to thank to my supervisor, Dr. Yang Liu. It is he who provided plenty of recommendations for me in time during the whole period of my studies, not only in this thesis but also in former projects. Although he does not work for Vaasan Ammattikorkeakoulu, Vaasa University of Applied Sciences any more, he was still patient and he kept his promises that supported me to complete my thesis. Besides, Dr. Yang Liu possesses highly professional guidance. Without his many suggestions, I would not have improved my professional knowledge so fast.

As a software engineering student, I am very thankful for the principal lecturer Dr. Ghodrat Moghdampour. He was my the teacher in most of courses, and he helped me a lot when I faced difficulties in my specialized field. In addition, he is very warmhearted and kind. He was willing to help students to apply for the Master's degree.

Finally, I want to express my appreciation to my parents and friends who supported and encouraged me to go through with three years in Vaasa. And all the staff in Vaasan Ammattikorkeakoulu, Vaasa University of Applied Sciences, Dr. Chao Gao, Mr. Santiago Chavez Vega, Dr. Seppo Mäkinen and the other teachers who taught me and gave me a lot of help.

Shujun Liu

Vaasa, Finland

28.04.2019

ABSTRACT

Author	Shujun Liu
Title	A Human-Following NAO Robot Using Python Programming Language
Year	2019
Language	English
Pages	56
Name of Supervisor	Yang Liu

Robots are today studied by a growing number of research institutes to research of putting robots, especially humanoid robots, into the market that to replace traditionally human workers. Apart from this, robot is a powerful tool to engage students with developing AI in education. In the future, robots will be much more human-friendly with human beings and it is a trend to enable robots to co-exist with human beings. With more and more humanoid robots being manufactured, robots and humans are getting as close as possible to each other. Humanoid robots are appearing in many areas of life, such as doing housework, taking care of the elders, and accompanying children. To create a healthy co-existence environment of human and robot, robots have to learn to follow mankind as one of the methods to move.

This thesis presented face and object detection to achieve human-following function. The NAO robot could avoid obstacles and move to human beings. This project included Google speech recognition, which converted audio files to text files and enabled the NAO robot to carry out a simplified communication with human beings, using OpenCV trained cascade, which recognized human's face and the corresponding box. Also, the Caffe model, which drew face frames and face's confidence level was used. Finally, the work utilized NAOqi modules to control the whole-body joint of the NAO robot and employ the NAO robot to follow human.

The study was carried out with a pre-trained model to detect human's face, walking to the front of human beings. In addition, the NAO humanoid robot would avoid a specific box and follow the user. The result indicated that OpenCV detected face by using an object or a face trained cascade file is not accurate enough at the moment. However, it is enough for this thesis.

Keywords Face and object detection, NAO humanoid robot, Human-following

CONTENTS

LIST OF FIGURES AND TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CCD	Charge-coupled Device
CMS	advanced content management systems
HTML	HyperText Markup Language
XML	Extensible Markup Language
JSON	JavaScript Object Notation
FTP	File Transfer Protocol
IMAP	Internet Message Access Protocol
SciPy	A collection of packages for mathematics, science, and engineering
IPython	A powerful interactive shell
GUI	Graphical User Interface
macOS	Macintosh operating systems
SSH	Secure Shell
IP	Internet Protocol
FTP	File Transfer Protocol
OS	Operating System
PC	Personal Computer
API	application programming interface

DMC	Device Communication Manager
FSR	Force Sensitive Resistors
LED	Light-emitting diode
HTTP	HyperText Transfer Protocol
BSD	Berkeley Software Distribution license
I/O	Input/Output
GPU	Graphics Processing Unit
FPS	Frames per second
ID	Identity
HD	High Definition
CNN	Convolutional Neural Networks

1 INTRODUCTION

1.1 Background

Since robots appear in human's life frequently, a humanoid robot will be achievable in the coming decade. A humanoid robot is also called Android that people provide the model for them to imitate. The object of a humanoid robot is to look like a real human being, not only in appearance but also in behavior. With the rapid revolution in robots and humanoid robot area, scientists have made it come true and created a human-robot social interaction. Robots will hold an advantageous market position in the years to come, especially for professional service robots and service robots for individuals and families.

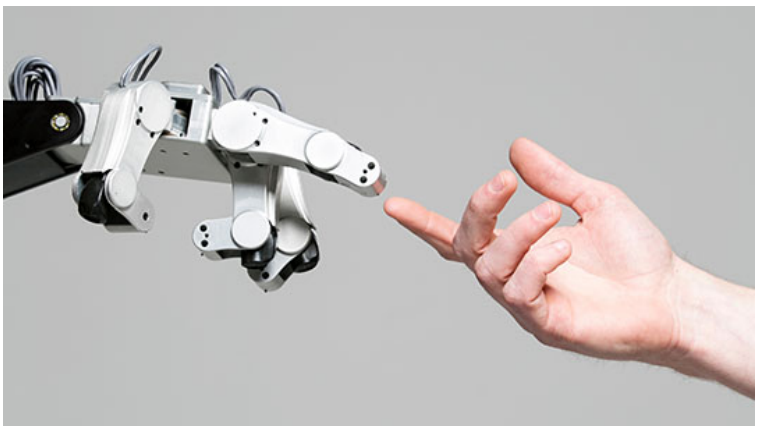


Figure 1. Human-robot interaction /1/

This thesis employed a humanoid robot which is named NAO robot. NAO robot is a member of Softbank Robotics Community in France and it is the first robot member in this research. A NAO robot was used for school and lab. Later, Pepper robot, the first wireless humanoid robots, has been released by Softbank Robotics Community and the following is Romeo robot who is the NAO robot's brother and the target is to be human's friend.

On the software side, NAO robot is equipped with the NAOqi operating system by Softbank Robotics Community. Besides, Softbank Robotics Community worked out an application "Choregraphe" to control the NAO robot easily by using Python programming language.

With more and more robots being manufactured, it is necessary to create a healthy co-existence environment and social between robots and human beings. Therefore, it is not enough to enable robot to look like a human, training the robot is needed. The first thing is to train the robot to follow by human beings. This is the reason why to follow a human was chosen as the topic of study.

A Human-following robot has been researched and developed actively the past decades due to its many applications in daily life and manufacturing. A human-following robot requires several techniques such as human's target detection, a robot control algorithm and obstacles avoidance. Various approaches of following robots have been proposed, for example using ultrasonic sensors, voice recognition sensors, laser range sensors, or charge-coupled device (CCD) camera. /2/

AI is regarded as the main task for many countries. In order to improve its performance, developing speech recognition, control algorithm and machine vision are matters of urgency. These are also the key points for solving difficult technical problems on human-following robot. Human-following is a friendly way to achieve interaction and utilization between human beings and robots. Humanoid robots cannot exist in the future without the deep learning method. It is a compilated logic technique that robot records a chain of conversations with human beings and through a certain algorithm passes judgment on human actions, habits and appearances.

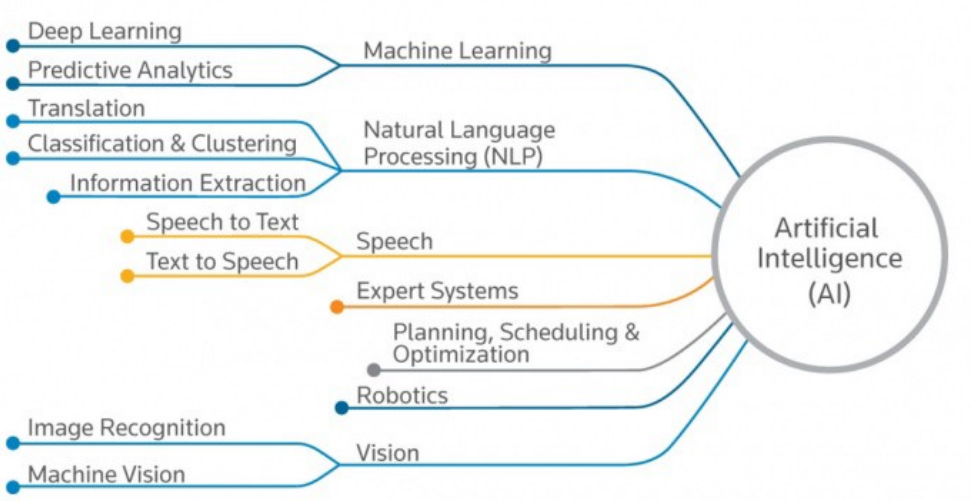


Figure 2. Artificial Intelligence's filed /3/

For the sake of working out a more satisfactory solution for speech recognition, controlling algorithm and machine vision under limited conditions, this paper utilized Google speech recognition, OpenCV and NAOqi modules. Google speech recognition is a module for converting wav files to txt files. It trained some pre-trained models of face and object detection and calculated the distance between robot to human via OpenCV. Finally, there were NAOqi modules. For the purpose of controlling NAO robot easily and simplify, a NAO robot has its operating system and modules. A NAO robot has its own basic modules, for example, a speech module, a video recorder module, a motion module etc. Then, these methods were combined to implement human-following projects based on a NAO robot.

2 TOOLS AND TECHNOLOGIES

2.1 Python

Python is a widely used software programming language and it is relatively easy to learn and understand, which means Python is user-friendly to beginners who want to study software programming languages. For Python's code readability and simplicity, it is easy for a programmer to explain what to show in the lines of codes. In comparison with Ruby and Perl, Python has a simple syntax of coding. Furthermore, Python is equipped with a strong standard library and data structure, which is why Python attracts people. Data analysis, web crawlers, game etcetera have come into usage.

Python is used in many application domains. For web and internet development, Python offers many choices for web development: frameworks such as Django and Pyramid, micro-frameworks such as Flask and Bottle, advanced content management systems such as Plone and Django CMS. Python's standard library supports many Internet protocols: HTML and XML, JSON, e-mail processing, support for FTP, IMAP, and other Internet protocols, easy-to-use socket interface. For purposes, Python is widely used in scientific and numeric computing: SciPy is a collection of packages for mathematics, science, and engineering, Pandas is a data analysis and modeling library, IPython is a powerful interactive shell that features easy editing and recording of a work session, and supports visualizations and parallel computing, the Software Carpentry Course teaches basic skills for scientific computing, running bootcamps and providing open-access teaching materials. For education, Python is a superb language for teaching programming, both at the introductory level and in more advanced courses. The Tk GUI library is included with most binary distributions of Python. Python is often used as a support language for software developers, for build control and management, testing, and in many other ways. /4/

Different from other programming languages, Python language is simplified, which means Python can use 100 lines codes to replace 1000 lines codes of other programming languages. So, this project used Python 3 as the main programming

language. It is worth nothing that because Python 2 will not update after 2020, it is better to use Python 3 as the main programming language from now and then. Basically, the idea of Python 2 and Python 3 are the same, there is only some difference with a small amount of syntax. For this reason, whenas you are familiar with Python to a certain degree, people who are familiar with Python 2 can write Python 3 code in a very short time. The differences between the two version are shown in Table 1.

	python 2.x	**python 3.x**

Print function	`print 'Python', sys.version`
	`print 'Hello, World!'`
	`print ('Hello, World!')`
	`print('Hello, World!')`
		`print('Python', sys.version)`
		`print('Hello, World!')`

Unicode	ASCII str()	ASCII str() Unicode (utf-8)
Division operation	For the division between integers, the result will not be a floating-point number.	For the division between integers, the result will also be a floating-point number.
Inequality	!= <>	!=
xrange() function	Both xrange() function and range() function can be used.	xrange() function may occur errors.
Module name changed	repr configParser raw_input() and input() …	reprlib configparser input() …

Table 1. Differences between Python 2 and Python 3

Below is an example of Python:

```
import datetime
oTime = datetime.datetime.now()
print(oTime.isoformat())
```

Code Snippet 1. Simple example of Python

After downloading Python's library by Terminal or PyCharm CE in Mac OS, Python is simplified to import a library, in comparison with Java which should

import library file to a specifically location manually. After that, it can, for example, be programmed and printed.

2.2 Terminal (macOS)

Terminal is the terminal application of macOS X operating system by Apple Inc. Terminal is an xterm-compliant shell with full international support, a Safari style tabbed interface, and streamlined preferences for easy access to the UNIX command line. Terminal application includes features such as:

- Colored output
- Enhanced drag-and-drop support
- Full-screen support
- New status controls
- Session support
- Built-in systemwide services
- Richer background images /5/

This thesis utilizes Terminal application to work on the remote NAOqi operating system. As a common tool to connect to a remote Unix/Linux system with the SSH protocol, Terminal basically need the IP address and use the command to make this connection, reducing the procedure of login phrase. A terminal emulator enables users to communicate with a program though graphical user interface, which is called the Terminal window in Figure 3.

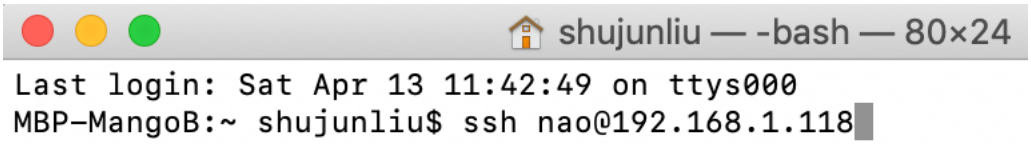


Figure 3. Terminal Window

When openning Terminal application, the command prompt will be shown. The basic usage of remote connection is to enter SSH syntax. “nao” is the remote user’s name. “192.168.1.118” is the IP address of the remote computer, which is the IP address of NAO robot. Then enter the correct password is entered based on SSH

protocol of NAO robot into Terminal, and the remote connection will be done through SSH.

2.3 ForkLift

As a file manager and FTP tool for macOS, ForkLift can be a tool substitute for Total Commander, which is a file manager for Windows. BinaryNights, LLC developed a free version of ForkLift in App Store, which is named ForkLift 2. Users can also install another version, the most advanced dual pane file manager ForkLift 3 in binarynights.com website. This thesis uses ForkLift 2 as the main tool in the whole project. ForkLift 2 has the following features:

- Multiple styles of panes
- Edit and delete files remotely
- Transfer date directly between FXP-enabled servers
- Open Terminal at the current path
- Search files though name, type, tag or content etc.

In order to edit and transfer some Python files to NAOqi OS, this paper uses ForkLift 2 to connect to NAOqi OS. As Figure 4 shows below, ForkLift 2 will request user to enter NAO robot IP address and password for the server. The primary thing to note is that the PC and NAO robot should be under the same Wi-Fi connection.

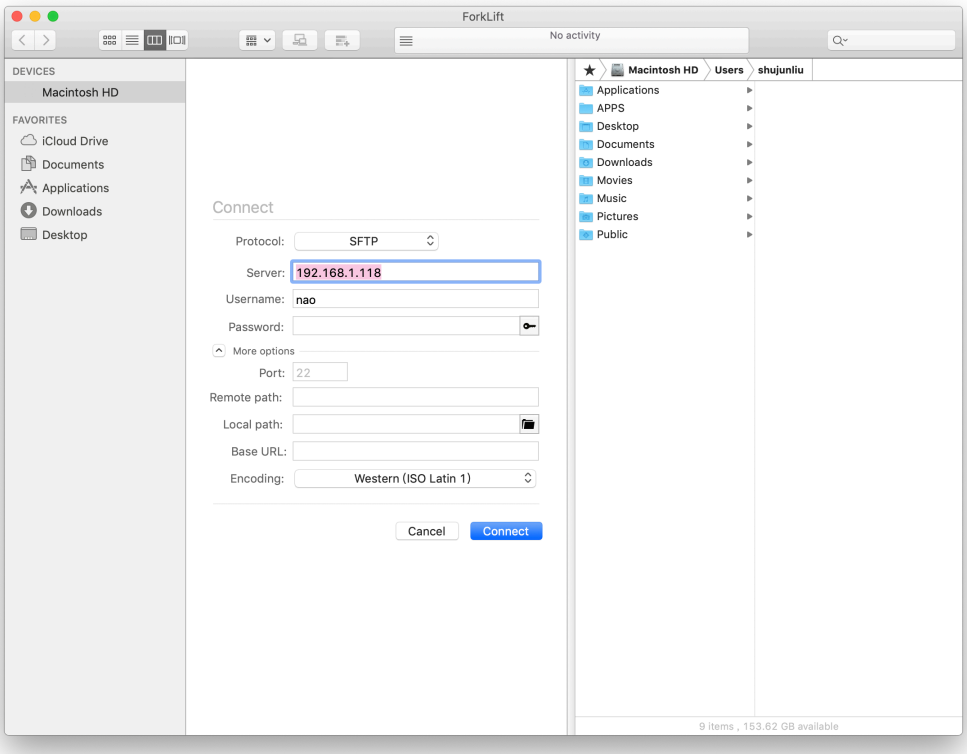


Figure 4. ForkLift 2 window

2.4 NAO Robot

NAO is the first robot created by SoftBank Robotics. NAO is a tremendous programming tool and it has especially become a standard in education and research. NAO is also used as an assistant by companies and healthcare centers to welcome, inform and entertain visitors. /6/

Sufficient language packages and flexible joints enable NAO to help human beings either in education or in research. The development of NAO humanoid robot is based on the human development of programming languages. Generally speaking, NAO, as a programmable humanoid robot, has a small number of components, including:

- Twenty-five degrees of freedom which enable him to move and adapt to his environment.

- Seven touch sensors located on head, hand and feet, sonars and an inertial unit to perceive his environment and locate himself in space.
- Four directional microphones and speakers to interact with humans.
- Speech recognition and dialogue available in 20 languages, English, French, Spanish, German, Italian, Arabic, Dutch, Portuguese, Czech, Finnish, Russian, Swedish, Turkish etc.
- Two 2D cameras to recognize shapes, objects and even people.
- Open and fully programmable platform. /7/

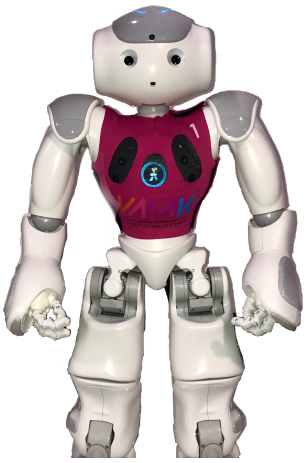


Figure 5. NAO Robot at VAMK

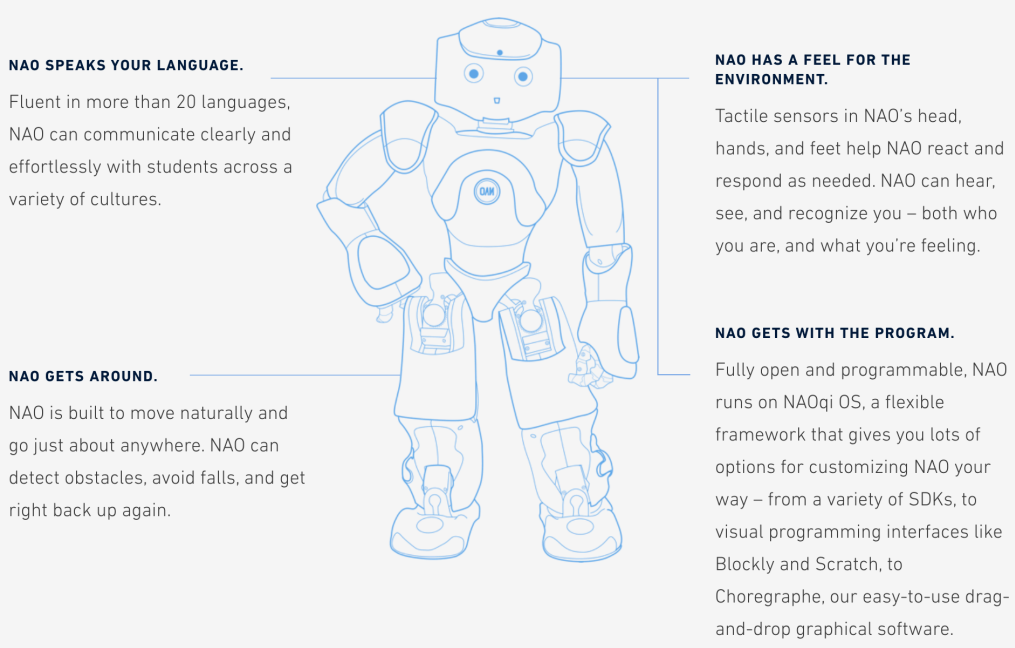


Figure 6. Futures of NAO Robot /8/

Three Touch sensors were equipped on the NAO robot, which are on its head, in the front of the head, in the middle of the head and in the rear of the head. To enable robot and human interaction easily, this paper utilizes the front of the head and the middle of the head to manage NAO robot's start and stop of moving to action.

2.5 NAOqi

The operating system of the NAO robot is named NAOqi OS. It is utilized for meeting the demands of robots from SoftBank Robotics. NAO robot, Pepper robot and Romeo robot all are members of the NAOqi's family.

Figure 7. NAOqi's Family/9/

The NAOqi OS virtual machine provides a virtualized NAO environment. Thus, there is no graphical frontend, just a console. It mainly targets developers and aims at fitting their needs. Table 10 shows the key inclusion NAOqi OS: /10/

Developer tools	gcc compiler suite
	CMake
	qiBuild
	Autotools
	emerge and portage package tree
	Make easier building third-party libraries for Aldebaran robots.

Aldebaran material	libqi/libnaoqi
Miscellaneous	man pages

Table 2. Key Inclusion of NAOqi's OS

Moreover, according to the supported programming languages on NAOqi, Python is the best match programming language. Python can run directly on both a computer and a robot. In comparison, Java cannot run directly on the robot. Furthermore, in Choregraphe (the easiest platform for controlling NAO robot and building your projects without writing lines of codes), Python has the advantage over other programming languages of supporting to building apps and editing codes. Thus, Python is the better choice for fitting and creating projects with a NAO robot.

Programming Languages	Bindings running on		Choregraphe support	
	Computer	Robot	Build Apps	Edit code
Python	✔	✔	✔	✔
C++	✔	✔	⊘	⊘
Java	✔	⊘	⊘	⊘
JavaScript	✔	✔	✔	⊘

✔	OK
⊘	Not available

Table 3. Supported Programming Languages /11/

NAOqi OS is equipped a few of NAO robot modules. For example, NAOqi core API, NAOqi motion API, NAOqi audio API, NAOqi vision API, NAOqi sensor API, NAOqi tracker API and DCM API. NAOqi core API is one of the key points of NAOqi modules. It is used to control behaviors and connections of network. Moreover, it can be regarded as an API that you can create your own modules and then edit to use or obtain another modules' function to use. NAOqi motion API provides some modules to control NAO robot's 25 degrees of freedom, all over NAO robot's neck, arms, shoulder, fingers, legs, hip and feet. NAOqi audio API is to manage sound and speech modules. Sound module also can detect the location of object that near with NAO robot. Speech module enables NAO robot to record what human beings said and say something based on a text file. On the basis of NAOqi vision API, NAO robot can do more things, such as record a video, capture an image from video, detect dark or landmark and so on. NAOqi sensor API includes battery, infrared, Force Sensitive Resistors (FSR), laser sensors, sonar

sensors and LED sensors. NAOqi tracker API allows NAO robot to track targets, like a red ball or a face so that NAO robot will always look at the object and enable target in the middle of NAO's camera. This thesis utilized all APIs above.

2.6 Speech Recognition

Speech recognition is a popular and usual technique currently, especially in the machine learning area. It can convert human voice to text file. Robots are being used by a growing number of companies and factories, even in families and schools, to complete some work and research which is traditionally done by human workers. For this reason, speech recognition becomes a necessary technique. To enable a robot and a human being to interact on the basis of the same language, it is a trend to use speech recognition in artificial intelligence and machine learning filed.

Today, many large companies provide APIs for performing different machine learning tasks. Speech recognition is not an exception. You do not have to be expert in natural language processing to use these APIs. Usually, they provide a convenient interface. All you need to do is to send an HTTP request with the required content to the API's server. Then you will receive the response with completed tasks. This approach is useful whenas you don't need anything special, if your problem is standard and well-known. One more advantage of this way is that you can save such valuable resources such as time and money.

Nevertheless, there are many situations where it is not possible to use API and need to develop a speech recognition system from scratch. This way is rather complex, it requires much effort and resources, but as a result, it is possible to create a system that will be ideally compatible with your needs. Also, it is possible to improve the quality of the results if algorithms is built. However, it is good to know about APIs. You can understand what each API can do and what pros and cons they have. As a result, you will be able to detect when you should use API (and what API) and when you should think about your own system. In this article, we want to compare the most popular APIs which can work with human speech. /12/

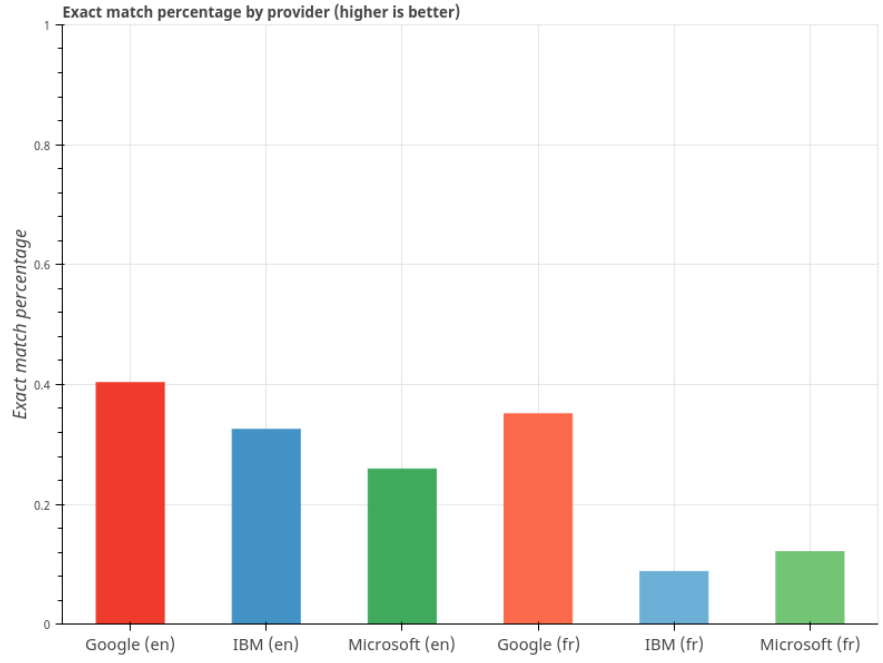
Speech recognition is a library of Python and Google speech recognition was accepted by it. The full name of Google speech recognition function is Cloud Speech-to-Text. As it indicates literally, Google speech recognition is utilized to convert audio files to text files in this thesis. Afterwards the NAO robot will recognize the text files and then have a conversation. In addition, SpeechRecognition 3.8.1 will be installed and download which is an ordinary characteristic of Python projects.

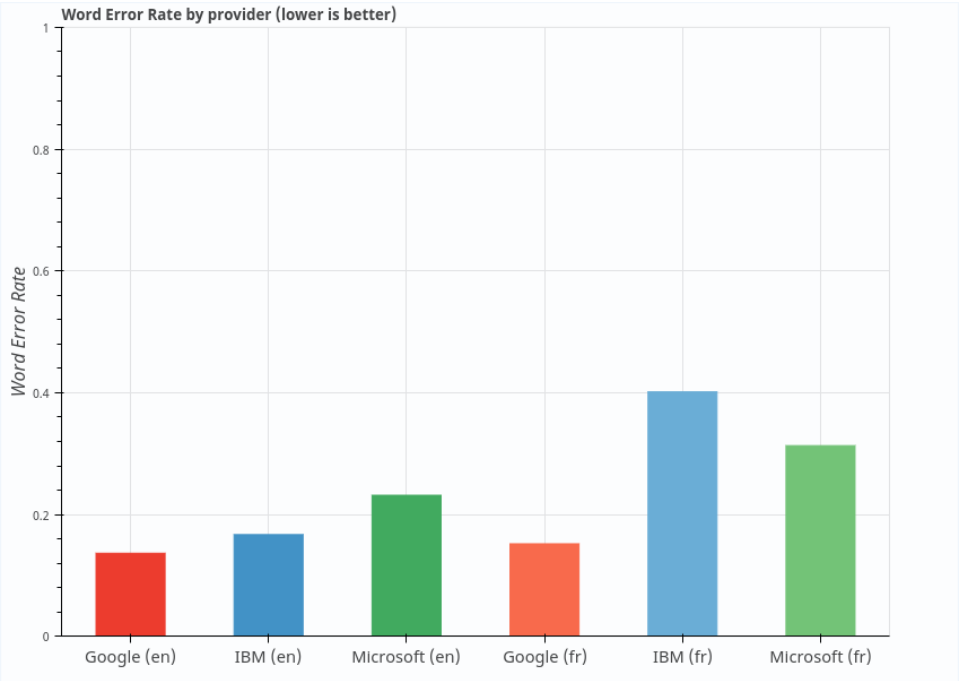
Figure 8. Latest version of SpeechRecogntion

Python has the library of speech recognition, moreover, this Python library supports most of engines and APIs. SpeechRecognition 3.8.1 is a common Python project which is used to effect compatibility between different engines. The main engines and APIs are:

- CMU Sphinx (works offline)
- Google Speech Recognition
- Google Cloud Speech API
- Wit.ai
- Microsoft Bing Voice Recognition
- Houndify API
- IBM Speech to Text
- Snowboy Hotword Detection (works offline),

As one of the top speech recognitions in the world, Google speech recognition has its own advantages. In addition, there are other speech recognitions, such as IBM Watson Speech to Text, Microsoft Azure Bing Speech API, Amazon Transcribe, and Amazon Polly. In comparison with other speech recognitions, Google speech recognition has a better performance. Figure 9 shows the differences between speech recognitions applications.


Exact match percentage by provider (higher is better)
Exact match percentage
1
0.8
0.6
0.4
0.2
0
Google (en)
IBM (en)
Microsoft (en)
Google (fr)
IBM (fr)
Microsoft (fr)


Word Error Rate by provider (lower is better)
Word Error Rate
1
0.8
0.6
0.4
0.2
0
Google (en)
IBM (en)
Microsoft (en)
Google (fr)
IBM (fr)
Microsoft (fr)

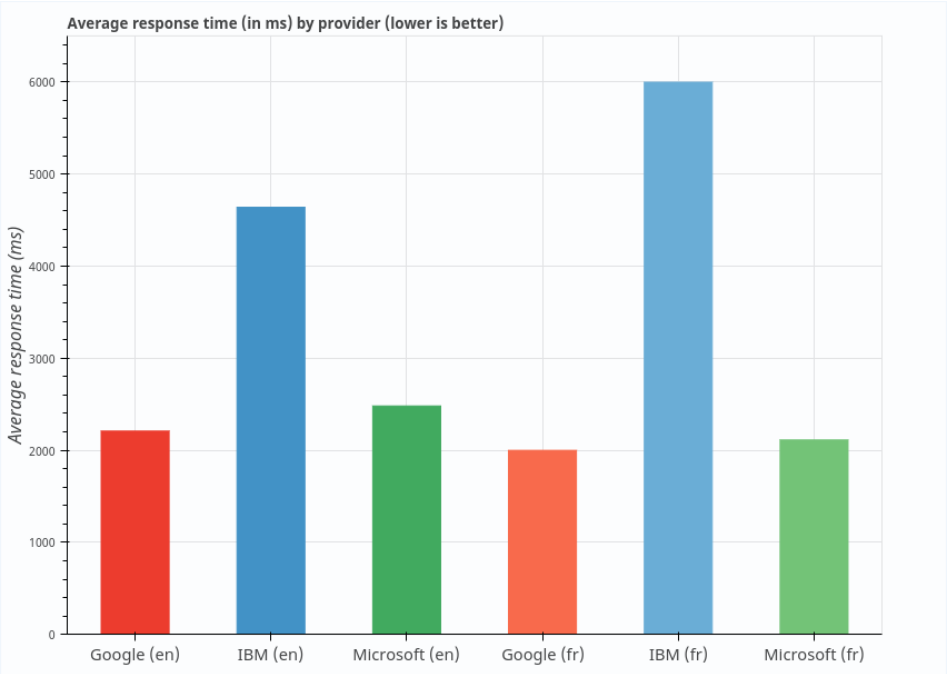


Figure 9. Differences between speech recognitions /13/

The charts in Figure 9 clearly shows that Google speech recognition has greater combination properties and advantages than other speech recognitions. Google speech recognition has higher exact match percentage, which means each word and letters should be the same in one sample. Also, the word error rate is lower. When Google speech recognition converts an audio file to a text file, there is a lower percentage of word recognition errors than in other speech recognitions tools. It is the supreme performance in the machine learning field. Google speech recognition's average response time is lower, so Google can process more speech audio files to text files at a certain time.

2.7 OpenCV

OpenCV is open source computer vision library, based on BSD protocol and designed for computational efficiency and with a strong focus on real-time applications. It is utilized for doing some research and business. OpenCV has the programming languages interfaces, which allows users to connect with C++, C, Python and Java, and OpenCV supports Linux, Windows, Mac OS, IOS and Android systems. OpenCV is dedicated to furthering computational efficiency and

with a strong focus on real-time applications. Because optimization of OpenCV was written by C++, it takes full advantages of multi-processor and OpenSL interface is being developed in active. There are over 500 algorithms and about ten times as many functions that compose or support those algorithms. It is usually implemented on machine learning and network maps.

The library of OpenCV has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people in its user community and the estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and governmental bodies. /14/

The following are the common and static libraries of OpenCV:

- Core functionality
- Image processing
- Video
- Calib3d
- Features2d
- Objectect
- Highgui
- Video I/O
- GPU
- Some supported modules

To implement OpenCV with Python, the two main libraries are needed: Numpy, and Matplotlib. While OpenCV's installation is successful, codes, snippets shown in Code Snippet 2, are used import these three main libraries,

```
import numpy as np
import matplotlib
import cv2
```

Code Snippet 2. Import the main libraries

2.8 Caffe

Caffe is a Deep Learning Framework created by Facebook which allows users to create Deep Learning models. Deep Learning is utilized in many fields, especially in machine learning.

The model that is used in this work is a Convolutional Neural Network. A Convolutional Neural Network is a Deep Learning Architecture that is used for image recognition. This architecture has several variants, and some are better than others in recognizing objects. A certain architecture is chosen depending on the needs, but basically, all have the same architecture. In a high level, a Convolutional Neural Networks is trained using supervised learning, which basically means that the network needs to be shown several pictures which have a certain object, for example, a car and then the network is told that object represents a car then the network is trained by fixing the weights. In CNN the weights belong to a matrix, for example, 3×3 matrix (Although it could be a 4×4 or 5×5 matrix) then a convolutional operation is completed with that matrix to the image, then the Relu is applied (Relu is an activation function which reduces the likelihood for the vanishing problem). After that, the result is passed for a pooling layer, which is just a square-matrix (for example a 2×2 matrix) that will reduce the convolutional image. What the pooling layer does is choose the max number inside the matrix and created a smaller image. Those steps are repeated several times. Too many layers or to few layer will result in poor object recognition. /15/

Convolutional Neural Networks are always used for image classification and object detection. In this project, a pre-trained model is utilized for face detection and for calculating a confidence level for human face.

3 ANALYSIS AND DESIGN

3.1 The Structure of Project

The overall structure of this project consists of 13 modules: ALProxy module, ALBroker module, ALModule module, ALMemory module, ALVideoRecorder module, ALAudioRecorder module, Google Speech Recognition module, ALTextToSpeech module, ALVideoDevice module, OpenCV module, Caffe model, ALMotion module and ALTracker module. The main purpose is to enable a NAO robot to follow human by face detection. On this basis, the NAO robot will avoid specifical obstacles. The language is set as English.

The use case diagram and flowchart are shown in Figure 11 and 12. At first, it should be made sure that Terminal and ForkLift have been connected to the NAO robot correctly. After the user touches the front head sensor of the NAO robot, the robot will open its own camera and microphone. The NAO robot will ask the user if she/he wants to play with the robot. And then, the NAO robot will record what the user answers to a .wav file, which will be recognized by Google Speech Recognition API. If user says yes, the NAO robot will wake up as an initial stand pose. In the next session, the NAO robot will capture images by its camera. Next, NAO will check if human face in front of its camera by using OpenCV. If OpenCV detects a face, next it calculates the distance between the human to the robot. After checking if the distance is more than 30 inches, the NAO robot will make a decision to move to the human or stop. In addition, if the NAO robot detects that the object is in front of its, it will move right or left three steps and then walk to the human. However, its obstacle avoidance is inaccurate each time. Finally, after going through a certain number of times, the NAO robot will ask for a break. If the answer is yes, the whole project will end. Otherwise, the NAO robot will repeat the movement progress.

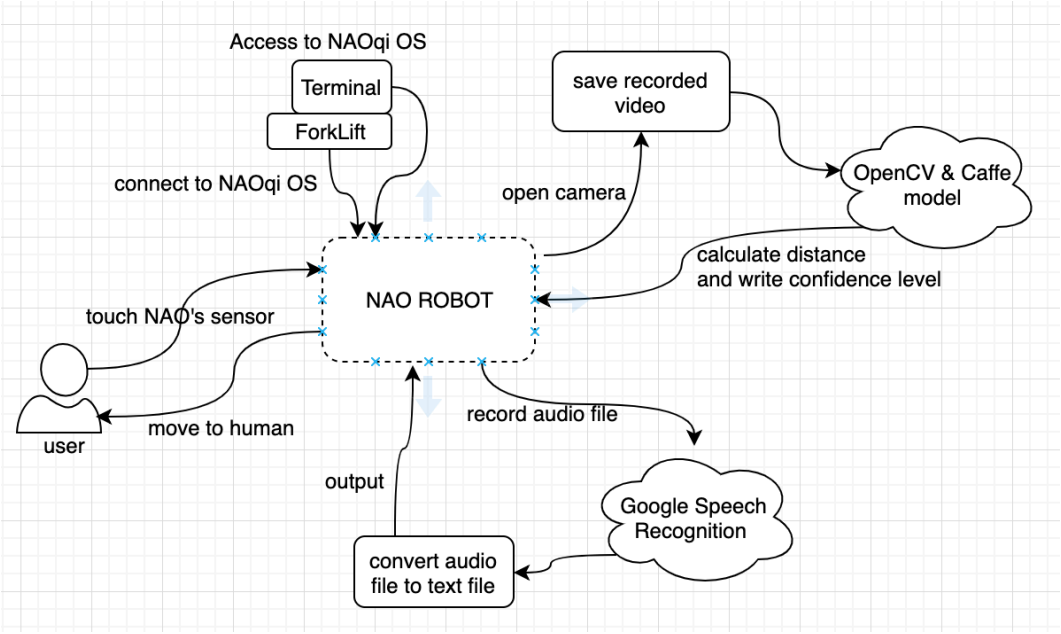


Figure 10. Use case diagram of project

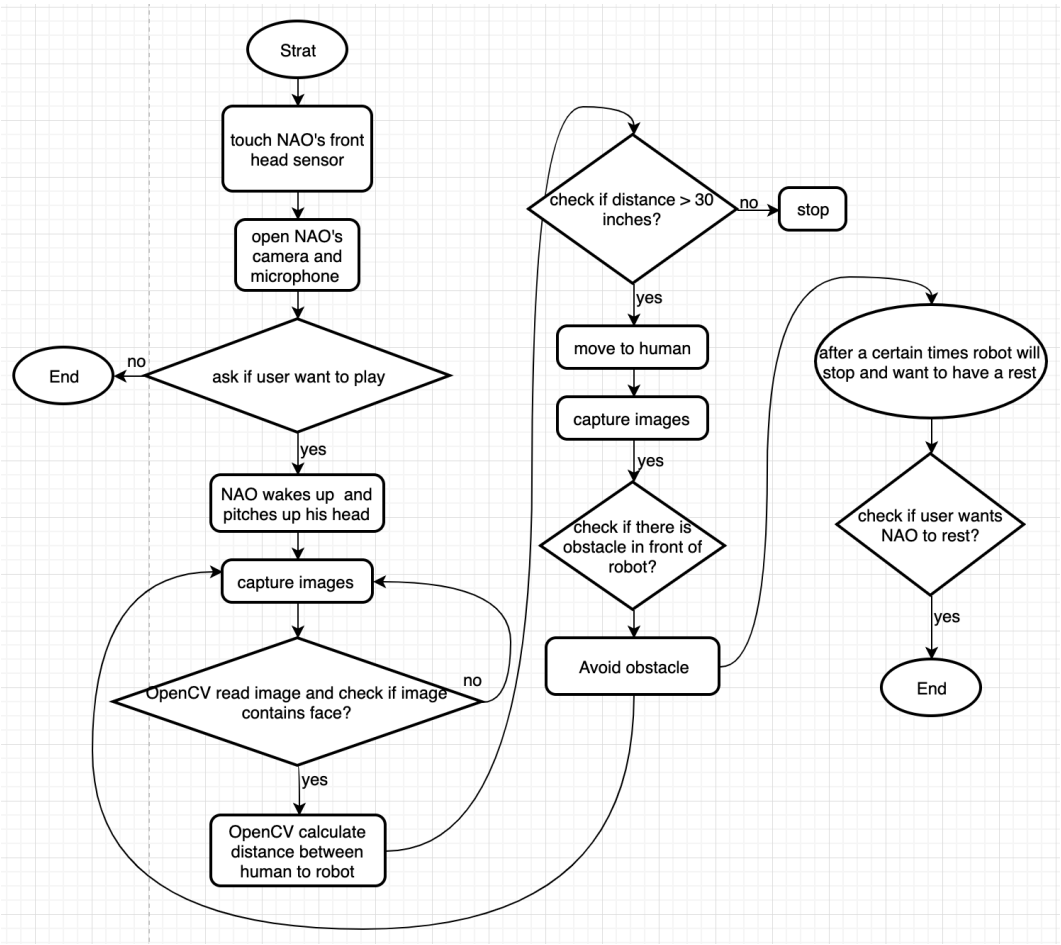


Figure 11. Flowchart of project

3.2 Description of Modules

NAOqi has several different modules to control parts of the NAO robot including motion, audio, vision, sensors and trackers. The human-following the NAO robot project uses mainly NAOqi API, ALMemory module, ALVideoRecorder module, ALAudioRecorder module, Google Speech Recognition module, ALTextToSpeech module, ALVideoDevice module, OpenCV module, Caffe model, ALMotion module and ALTracker module.

3.2.1 Mainly NAOqi Modules

As each module, AL is the namespace of all modules and all modules inherits from ALModule.

The mainly Python modules are ALProxy, ALBroker and ALModule in NAOqi OS. ALProxy as the key NAOqi Python module is used to create a proxy to a module, and it is an essential module in all NAO robot's projects. There are two types of constructors used in Python project, as shown in Table 4. The differences of these two constructors is whether or not the project has a Broker instance available.

ALProxy(name, ip, port)	● name - name of the module. ● ip - The IP of the broker in which the module is running. ● port – The port of the broker.
ALProxy(name)	● name - name of the module.

Table 4. Supported Constructors of ALProxy Object .

ALBroker means creating a Python broker and ALModule means writing NAOqi modules in Python. They are only used on a specific occasion when the aim is to write NAOqi modules in Python, for instance because there is a need to subscribe to an event with a callback.

ALBroker (name, ip, port, parent_ip, parent_port)	● name - name of the broker. ● ip - The IP the broker will be listening to.

	● port - The port the broker will be listening to. If the it equals 0, we will find a free port and use this. ● parent_ip - The IP of an other broker to connect to. ● parent_port - The port of an other broker to connect to.
ALModule(name)	● name - name of the broker.

Table 5. Supported Constructors of ALBroker and ALModule Object

All these mainly modules are utilized in this project. As a result, the import mainly modules should be indicated as Code Snippet 3 shown:

```
from naoqi import ALProxy
from naoqi import ALBroker
from naoqi import ALModule
```

Code Snippet 3. import mainly modules

3.2.2 ALMemory module

ALMemory is utilized to store and retrieve events which are named in NAOqi OS. ALModule contains many specific methods, for example, declareEvent(), getData(), getEventHistory(), subscribeToEvent(), subscribeToMicroEvent(), etc. This thesis uses subscribeToEvent() method to wake up NAO robot and shut down NAO robot.

```
global memory
memory = ALProxy("ALMemory")

memory.subscribeToEvent("FrontTactilTouched",
                        "walkTo",
                        "onWake")

memory.subscribeToEvent("MiddleTactilTouched",
                        "walkTo",
                        "onRest")
```

Code Snippet 4. subscribeToEvent() method

SubscribeToEvent is used to subscribe to an event, FrontTactilTouched and MiddleTactilTouched events, and lunch the event automatically. The event name can also be given via creating an event method. In this thesis, FrontTactilTouched and MiddleTactilTouched events are subscribed. FrontTactilTouched means that

when touching the NAO robot' front head sensor, something will happen. It is same to MiddleTactilTouched event, which means that while touching the NAO robot's middle head sensor. WalkTo function will be called. OnWake and onRest are names of the module's methods. In short, when the front head sensor is touched, the NAO robot will wake up and run this walkTo project. When the middle head sensor is touched, the NAO robot will shut down and stop this walkTo project.

3.2.3 ALVideoRecorder module

The ALVideoRecorder module allows you to record a video via NAO robot's cameras and save them on NAOqi OS. ALVideoRecorder has startRecording() and stopRecording()functions, which are used to record AVI format videos. Similarly, call stopRecording() function stops recording video and call startRecording() starts recording video. After stopping the stopRecording() function, the recorded video will be saved to the specific path with a specific name. After creating a proxy to ALVideoRecorder, its functions are quite straightforward to use. ALVideoRecorder has its own default parameters, but all of parameters can be manually changed. the parameters of NAO's video includes color space, frame rate, resolution and video format. The default parameters are:

- Color space: BGR
- Frame rate: 15 frames per second (FPS)
- Resolution: QVGA 320*240
- Video format: MJPG

However, the default parameters do not support legible image to make a face detection. As the result, this thesis uses below parameters:

- Color space: BGR
- Frame rate: 20 frames per second (FPS)
- Resolution: VGA 640*480
- Video format: MJPG

Color space parameter contains 17 values. The value name of BGR is kBGRColorSpace which includes triplet on the format 0xRRGGBB. BGR has three

channels and ID value is 13. MJPG cannot be recorded with a framerate lower than 3 fps. The value of resolution can be QQVGA, QVGA, VGA and 4VGA. The ID name of them are 0, 1, 2 and 3. 0 means image of 160*120px. 1 means image of 320*240px. 2 means image of 640*480px. 3 means image of 1280*960px. 4VGA should be better for face detection, but it is a large data that data analysis would be slower than VGA. Other two values are too indistinct.

```
def videorecording(self):
    self.videoRecorderProxy.stopRecording()
    # This records a 640*480 MJPG video at 20 fps.
    # Note MJPG can't be recorded with a framerate lower than 3 fps.
    self.videoRecorderProxy.setResolution(2)
    self.videoRecorderProxy.setFrameRate(20)
    self.videoRecorderProxy.setVideoFormat("MJPG")
    self.videoRecorderProxy.startRecording("/home/nao/WORK-
SPACE/src/cameras", "record")
```

Code Snippet 5. Set video's parameters and start recoding

It is essential to stop the video recording function at first, or the project will show the error that setResolution 1 resolution are occupied. Then the parameter of the recorded video is set. Set video's resolution is 640*480, frame rate is 20 frames per second and video format is MJPG format. The video's path is /home/nao/WORKSPACE/src/cameras and the video's name is "record". In addition, the video will record the whole process, so it does not need to set video record time. And all function will stop and shut down with stop function.

3.2.4 ALAudioRecorder module

ALAudioRecorder is used to record sounds from NAO's microphones and save it to a wav or a ogg file. The recording capabilities are for now limited to the following formats:

- four channels 48000Hz in OGG.
- four channels 48000Hz in WAV.
- one channels (front, rear, left or right), 16000Hz, in OGG.
- one channels (front, rear, left or right), 16000Hz, in WAV.

```
# ---------> Recording <---------
self.record.stopMicrophonesRecording()
print('Start recording...')
# tts.say("start recording...")
record_path = speech_record_file_path

self.record.startMicrophonesRecording(record_path, 'wav', 16000, (0,
0, 1, 0))
time.sleep(3)
self.record.stopMicrophonesRecording()
print('stop recording')
# tts.say("stop recording")
```

Code Snippet 6. Start and stop microphones recording methods

ALAudioRecorder has two functions named startMicrophonesRecording() and stopMicrophonesRecording(). Just same as the ALVideoRecorder module, the recording of the microphones should be stopped first in each time, or the port will be occupied, and error messages will occur. StartMicrophonesRecording() function should indicate parameters, like filename, signal type, sample rate and channels. Record_path is the absolute path of the file and it is indicated before. 'wav' is one of the signal's type. Sample rate and channels are requested. The recording sounds will be saved as a specified wav file to record_path, and the recording of the NAO's front microphone at 16000Hz. In the next parameters, the first 0 means the left channel, and then the right channel, the last one is the rear channel. This project only configures the front channel to be recorded. Time.sleep(3) is to stop the microphone recording and close the file after 3 seconds.

3.2.5 Google Speech Recognition module

Firstly, Google Speech API has to be installed in NAOqi OS which is a Linux system. But Speech Recognition can also be transferred by ForkLift, if it is already download in Windows system and Mac OS as shown in Code Snippet 7.

```
git clone http://people.csail.mit.edu/hubert/git/pyaudio.git
cd pyaudio
sudo python setup.py install
sudo apt-get installl libportaudio-dev
sudo apt-get install python-dev
sudo apt-get install libportaudio0 libportaudio2 libportaudiocpp0 portaudio19-dev
sudo pip3 install SpeechRecognition
```

Code Snippet 7. Command of Speech API Installation

After installing, the speech recognition module can be used to record audio and the recorded speech can be sent to the Google speech recognition API, r.recognize_google(audio) returns a string, which will then return the output as shown in Code Snippet 8.

```
from naoqi import ALProxy
from naoqi import ALBroker
from naoqi import ALModule
import speech_recognition as sr

# audio file and text file's path
speech_record_file_path = "/home/nao/WORKSPACE/src/speechRecord.wav"
sample_text_file_path = "/home/nao/WORKSPACE/src/sample_text.wav"

def communication(self):
    global audio, r
    audio = r = None
    r = sr.Recognizer()
    ……
    # ---------> Recording <---------
    self.record.stopMicrophonesRecording()
    print('Start recording...')
    # tts.say("start recording...")
    record_path = speech_record_file_path

    self.record.startMicrophonesRecording(record_path, 'wav', 16000,
(0, 0, 1, 0))
    time.sleep(3)
    self.record.stopMicrophonesRecording()
    print('stop recording')
    # tts.say("stop recording")

# ---------> Speech recognition <---------
    with sr.WavFile("./src/speechRecord.wav") as source:
        audio = r.record(source)
    try:
        text = r.recognize_google(audio)
        print(text)
        if text == "yes":
            ……
    except sr.UnknownValueError:
        print("Google speech Recognition could not understand")
        self.tts.say("Speech recognition failed \\pau=500\\ you can
touch my front head and try again")

    except sr.RequestError as e:
        print("Could not request results from Google Speech Recogni-
tion service; {0}".format(e))
        self.tts.say("Speech recognition failed \\pau=500\\ you can
touch my front head and try again")
```

Code Snippet 8. example of Google speech recognition

3.2.6 ALTextToSpeech module

ALTextToSpeech is one of the basic modules in NAOqi API. It can get a notification, get the available languages, set volume and say something. This project uses the ALTextToSpeech module to say the specified string of characters function and have a simplify communication with human being.

As Code Snippet 9 shows, the tts.say function have a parameter which is a string. This string is the text that the NAO robot will say to human beings and it should be encoded in UTF-8. In code snippet 9, the ALTextToSpeech module is used to broadcast a wav file.

```
self.tts.setVolume(1.0)

self.tts.say("Hi \\pau=1000\\I am Nao robot \\pau=1000\\ do you want
to play with me ")
```

Code Snippet 9. set volume and say function

3.2.7 ALVideoDevice module

ALVideoDevice module is a portion of NAOqi vision API. The responsibility of ALVideoDevice is used to capture images from a video and supply images to be managed by other modules. However, it will obtain empty frames in NAO robot's camera, when processing some HD images from video. For example, when requesting 640*480px images, it can only get 12fps under 100Mb Ethernet. It could happen that a face is detected and the data processor will be slower. The parameters of ALVideoDevice module are shown in Table 6:

Parameters	Values
name(id)	It is a specific name of one module. while there is more than one module, It should be subscribe and unsubscribe.
resolution	4VGA (1280* 960) VGA (640 * 480) QVGA (320 * 240) QQVGA (160 * 120)
color space	YUV422 (native format of the camera) YUV (24 bits) Y (8 bits)

	RGB (24 bits) BGR (24 bits) HSY (24 bits)
frames per second (fps)	

Table 6. Parameters of ALVideoDevice

The same as ALAudioRecorder module and ALVideoRecorder, ALVideoDevice also should notice that video device unsubscribes before subscribing. But when testing the project, it becames inevitable that process errors occur after subscribing. Therefore, the project does not unsubscribe and it cannot run again. In this situation, the way to solve the problem is to change the name id to make sure it is different from the former one.

```
self.video_service = ALProxy("ALVideoDevice")

name_id = self.video_service.subscribe("getlength", 2, 11, 20)
nao_images = self.video_service.getImageRemote(name_id)
cv_images = self.to_cv_img(nao_images)
self.video_service.unsubscribe(name_id)
```

Code Snippet 10. Subscribe and unsubscribe video device

After capturing images, OpenCV should get image arrays. This paper indicated the values of the image's width, height and image data. The following is image information's array:

- [0]: width
- [1]: height
- [2]: number of layers
- [3]: ColorSpace
- [4]: timestamp (seconds)
- [5]: timestamp (micro-seconds)
- [6]: array of size height * width * nblayers containing image data
- [7]: camera ID (kTop=0, kBottom=1)
- [8]: left angle (radian)
- [9]: topAngle (radian)
- [10]: rightAngle (radian)
- [11]: bottomAngle (radian).

3.2.8 OpenCV module

OpenCV module is used to develop and process the vision module. This thesis uses OpenCV module to the process captured images and recognize human faces and objects. The main modules of OpenCV are core, highgui, imgproc, video, calib3d, features2d, ml, flann, objectdetection, for example. The details of each are:

- **core** - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- **imgproc** - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- **video** - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- **calib3d** - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- **features2d** - salient feature detectors, descriptors, and descriptor matchers.
- **objdetect** - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- **highgui** - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.
- **gpu** - GPU-accelerated algorithms from different OpenCV modules.
- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others. /16/

It should be attentive when OpenCV reads the image from NAO robot's camera which is colorful version, but it cannot be written by OpenCV. Therefore, OpenCV will convert the colorful version images to gray via cv2.imread method.

```
def to_cv_img(self, nao_img):
    image_width = nao_img[0]
    image_height = nao_img[1]
    array = nao_img[6]
    image_string = str(bytearray(array))

    result = Image.fromstring("RGB", (image_width, image_height), im-
age_string)
    result.save("/home/nao/WORKSPACE/src/cameras/capture_0.png",
"PNG")
    img_gray = cv2.imread("/home/nao/WORKSPACE/src/cameras/cap-
ture_0.png", cv2.IMREAD_GRAYSCALE)
    return img_gray
```

Code Snippet 11. OpenCV read and write images

Before handling images, a clear definition of how to get distance between a human to the robot each time is needed. The camera's image-forming principle is shown in Figure 12.

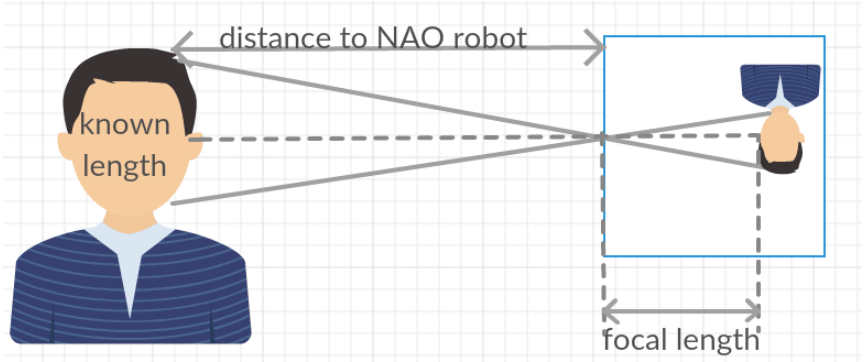


Figure 12. Find distance between human to robot

At the beginning of a human-following the NAO robot project, the first distance between the human to the NAO robot is initialized 30 inches, when the known face length is 8 inches. OpenCV will get NAO robot camera's apparent width in pixels. According to the triangle similarity formula for object to camera distance is:

$$F = \frac{(P \times D)}{W} \quad (1)$$

Camera's apparent width in pixels is multiplied by Distance from the camera to the human and then divided by object with known width. It will get the focal length of the NAO robot camera. As the result, this triangle similarity formula already has three fixed values: focal length, known width and camera's pixels. In the following, each images distance can be calculated:

$$D' = \frac{(W \times F)}{P} \qquad (2)$$

It is worth noticing that while using function cv2.imread() read an image. The image should be on the working directory or a full path of image should be given, because the NAOqi OS will not throw any error when the image path is wrong or missing. To be on the safe side, this program indicated the full path of the image during the complete lines code.

```
KNOWN_DISTANCE = 30.0
KNOWN_PERSON_HEIGHT = 8

img = cv2.imread('/home/nao/WORKSPACE/src/cameras/image_0.png')
mask = self.find_face(img, face_cascade)
for (x1, y1, x2, y2) in mask:
    # cv2.rectangle(img, (x1, y1), (x2, y2), (255, 255, 0), 2)
    y1_max = y1
    y2_max = y2


pix_height = y2_max - y1_max
real_focal_length = (pix_height * KNOWN_DISTANCE) / KNOWN_PER-
SON_HEIGHT
return real_focal_length
```

Code Snippet 12. Get focal length method

```
def distance_to_camera(self, known_width, focal_length, per_width):
    return (known_width * focal_length) / per_width
```

Code Snippet 13. Get distance each time

For the purpose of enabling the user to conveniently check if OpenCV recognized a face or an object correctly, a human-following the NAO robot program wrote the face frame via OpenCV as an image_frame.png file.

```
cv2.imwrite('/home/nao/WORKSPACE/src/cameras/image_0.png', cv_images)

face_frame = self.find_face(cv_images, face_cascade)
for (x1, y1, x2, y2) in face_frame:
    # rectangle frame: img,pt1,pt2,color(green),thickness=None,line-
Type=None,shift=None
    cv2.rectangle(cv_images, (x1, y1), (x2, y2), (255, 255, 255), 3)
cv2.imwrite('/home/nao/WORKSPACE/src/cameras/image_frame.png', cv_im-
ages)
```

Code Snippet 14. Get face frame method

Code snipped 14 enables the NAO robot to recognize a human face or make an obstacle avoidance. detectMultiScale method contains frame, scaleFator, minNeighbors and minSize in this program. The Frame parameter is a vector value of detecting an object or a face. The scaleFator parameter is the scale parameter in each image and the default value is 1.1. The scale parameter value is larger, the project works faster. However, the face and object's haar cascades are not accurate. This part modified 1.9 for obstacle avoidance and 1.3 for face detected. The minNeighbors parameter means that at least five times of overlapping detected, the face or the object exists. At the same time, the response time becomes slower. The minSize parameter indicates s minimal area of face detection.

Code Snippet 15. Face and obstacle detection function

Machine learning will not be developed well without training deep learning data.

```
def find_object(self, frame, cascade):
    object = cascade.detectMultiScale(frame, scaleFactor=1.9, min-
Neighbors=5, minSize=(35, 35))
    object = np.array([[x, y, x + w, y + h] for (x, y, w, h) in ob-
ject])
    pick = non_max_suppression(object, probs=None, overlapThresh=0.65)
    return pick
```

OpenCV train cascade is one of deep learning methods. This paper pre-trained the haar cascade for face and a specific obstacle. The way to mention face and object haar cascade is indicated as follows shown is Code Snippet 16.

```
face_cascade = cv2.CascadeClassifier('/home/nao/WORKSPACE/src/haarcas-
cade_frontalface_default.xml')
obj_cascade = cv2.CascadeClassifier('/home/nao/WORKSPACE/src/box_cas-
cade.xml')
```

Code Snippet 16. Indicate face and object haar cascade

3.2.9 Caffe model

Caffe model is used for deep learning via Python or other programming languages. Caffe offers model definitions, optimization settings and pre-trained weights. This project uses the pre-trained files:

- A .prototxt file – defines the model's structure

- A .caffemodel file – contains the weight of the actual layer

After saving the recorded video in NAOqi OS, loading serialized model from NAOqi OS in the next. After that, it should grab the frame from the threaded video stream and resize it to maximum width of 640 pixels, and the height of 480 pixels. This size should be smaller or equal to the original video. The important part is to grab the frame dimensions and convert it into a blob. Next, passing the blob through the network and obtaining the detections and predictions is done. In the following part, it is to extract the confidence level associated with the prediction. Filtering out weak detections is done by ensuring the `confidence` is greater than the minimum confidence. The last task is to draw the bounding box of the face along with the associated probability

```
prototxt = "/home/nao/WORKSPACE/src/deploy.prototxt"
model = "/home/nao/WORKSPACE/src/face_detector.caffemodel"

net = cv2.dnn.readNetFromCaffe(prototxt, model)

print("[INFO] starting video stream...")

cap = cv2.VideoCapture('r/home/nao/WORKSPACE/src/record.avi')
fourcc = cv2.VideoWriter_fourcc(*'MJPG')
out = cv2.VideoWriter('/home/nao/WORKSPACE/src/out.avi', fourcc, 20.0,
(640, 480))
while cap.isOpened():
    ret, frame = cap.read()

    if ret is True:
        frame = imutils.resize(frame, width=640, height=480)

        (height, weight) = frame.shape[:2]
        blob = cv2.dnn.blobFromImage(cv2.resize(frame, (640, 480)),
1.0,
                                     (640, 480), (104.0, 177.0,
123.0))

        net.setInput(blob)
        detections = net.forward()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        for i in range(0, detections.shape[2]):
            confidence = detections[0, 0, i, 2]
            if confidence < 0.5:
                continue
            box = detections[0, 0, i, 3:7] * np.array([weight, height,
weight, height])
            (startX, startY, endX, endY) = box.astype("int")
            text = "{:.2f}%".format(confidence * 100)
            y = startY - 10 if startY - 10 > 10 else endY + 10
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                          (255, 0, 0), 2)
            cv2.putText(frame, text, (startX, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.45, (11, 255,
255), 2)

        out.write(frame)
        key = cv2.waitKey(1) & 0xFF

    else:
        break
out.release()
cap.release()
```

Code Snippet 17. Caffe model example

3.2.10 ALMotion module

ALMotion module included the Stiffness control method, the Joint control method, the Walk control method, the Cartesian control method, the Whole Body control method, the Self-collision avoidance method, the Fall manager method, the Smart Stiffness method, the General tools method and the ALMotionRecorder method.

The ALMotion module consists of managing walk and head joint part in a human-following the NAO robot project. In summary, the ALMotion module is a NAOqi module for controlling the whole body of the NAO robot.

The NAO robot has limitations of height (58cm). This project utilized the HeadPitch method to raise up to 30 degrees of his head. It is slightly useful for detecting face. Nevertheless, the NAO robot still cannot recognize a human's face who stands in front of him. Therefore, I chose to sit on a chair to record the whole process.

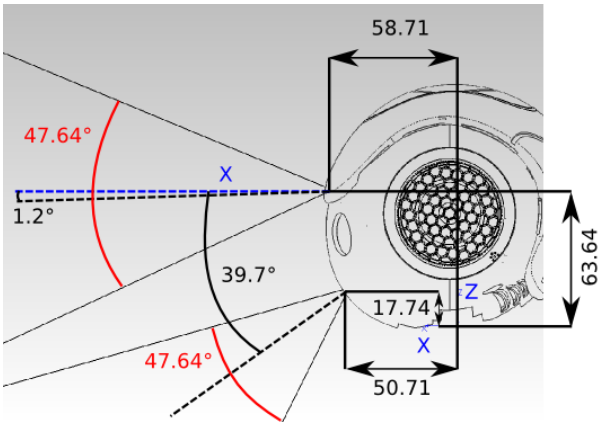


Figure 13. Limitation of NAO's sight /17/

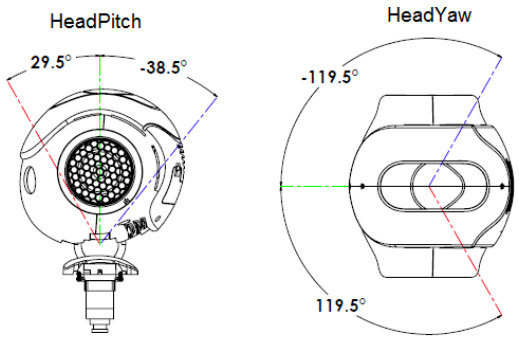


Figure 14. NAO robot's HeadPitch and HeadYaw angle /18/

As Code Snippet 18 shows below, "HeadPitch" is the head joint's name. There are two joints on the NAO robot's head which are named HeadYaw and HeadPitch. HeadYaw controls head shaking in Z-axial derection and its degree range is from -119.5 to 119.5 degrees. HeadPitch method is used to turn the NAO robot's head up and down and the degree range of the HeadPitch is from -38.5 to 29.5 degrees. The

fractionMaxSpeed means that the time the NAO robot requires to complete the entire process.

```
names = "HeadPitch"
angles = -30.0 * m.TO_RAD
fractionMaxSpeed = 0.1
self.motion.setAngles(names, angles, fractionMaxSpeed)
```

Code Snippet 18. HeadPitch method

Apart from this, this thesis designed a part that the NAO robot wants to have a break. It also used HeadPitch and HeadYaw methods to emphasize that the NAO robot is becomeing impatient. Code Snippet 19 mentions how to use the HeadPitch and the HeadYaw methods. Diffierent from controlling one method, while using HeadPitch and HeadYaw methods together, names, angleLists and times become array functions.

```
def headPitchYaw(self):
    names = ["HeadYaw", "HeadPitch"]
    angleLists = [[1.0, -1.0, 1.0, -1.0], [-1.0]]
    times = [[1.0, 2.0, 3.0, 4.0], [5.0]]
    isAbsolute = True
    self.motion.angleInterpolation(names, angleLists, times, isAbso-
lute)
```

Code Snippet 19. HeadPitch and HeadYaw methods

The major function of the ALMotion module is to control the NAO robot's motion. The NAO robot can move on carpet, tiles and wooden floors. The NAO robot cannot recognize if the floor has large frictional resistance or not. When frictional resistance is larger, he may not be able to walk and then tumble frontwards. Otherwise, frictional resistance is not enough to support him moving.

The basic foot step planner enables the NAO robot to move on x-axis or y-axis and the theta value which is rolling direction around the z-axis. The processes of all move actions are based on ALMath::Pose2D multiplication.

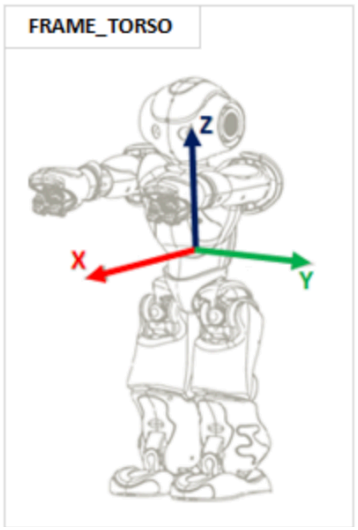


Figure 15. NAO robot axis definition /19/

The Code Snippet 20 shows custom parameters of move to action, x, y, theta. Theta means that the NAO robot degrees of turning, for example, 20 degrees, 30 degrees or 60 degrees. Y means direction of turning and the steps. While y is a negative number, the NAO robot will turn right. Otherwise, the robot turns left.

```
if (pixel/2) - 80 <= mid_point_value < (pixel/2) + 80:
        x = 0.2
        y = 0
        theta = 0
        print("towards")

    elif (pixel/2) - 180 <= mid_point_value < (pixel/2) - 80:
        x = 0.3
        y = 0.2
        theta = math.pi / 9
        print("L", 1)

    elif 20 <= mid_point_value < (pixel/2) - 180:
        x = 0.3
        y = 0.2
        theta = math.pi / 6
        print("L", 2)

    elif 0 < mid_point_value < 20:
        x = 0.3
        y = 0.2
        theta = math.pi / 3
        print("L", 3)

    elif (pixel/2) + 80 <= mid_point_value < (pixel/2) + 180:
        x = 0.3
        y = -0.2
        theta = -math.pi / 9
        print("R", 1)

    elif (pixel/2) + 180 <= mid_point_value <= pixel - 20:
        x = 0.3
        y = -0.2
        theta = -math.pi / 6
        print("R", 2)

    elif pixel - 20 < mid_point_value <= pixel:
        x = 0.3
        y = -0.2
        theta = -math.pi / 3
        print("R", 3)

    else:
        x = 0
        y = 0
        theta = 0
return x, y, theta
```

Code Snippet 20. Parameters of movement

Using moveTo(x, y, theta, frequency) method, the NAO robot can walk to a specific location. It can get a simplified robot position before walking via Pose2D multiplication and use this to know the robot's displacement. In the same way, the NAO robot gets its position after walking. Finally, the robot's' displacement is computed.

```
self.motion.moveInit()
robotPositionBeforeCommand = m.Pose2D(self.motion.getRobotPosi-
tion(False))

self.motion.post.moveTo(x, y, theta, frequency)
self.motion.waitUntilMoveIsFinished()
robotPositionAfterCommand = m.Pose2D(self.motion.getRobotPosi-
tion(False))

robotMoveCommand = m.pose2DInverse(robotPositionBeforeCommand) * ro-
botPositionAfterCommand
print("The Robot Move Command: ", robotMoveCommand)
print("-------")
```

Code Snippet 21. Get NAO robot’s displacement

4 IMPLEMENTATION

At the beginning of a human-following the NAO robot project, object haar cascade should be pre-trained. It is worth noting that the whole process of OpenCV training cascade should be in Linux OS. First, a workspace directory needs to be made to save images, OpenCV and haar cascade file. Next, OpenCV from GitHub is cloned and get some essentials.

```
git clone https://github.com/Itseez/opencv.git
sudo apt-get install build-essential
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev
libavformat-dev libswscale-dev
sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev
libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
sudo apt-get install libopencv-dev
```

Code Snippet 22. Preparation work for training cascade

While installing libjasper-dev, there occurred an error “Unable to locate package libjasper-dev” in Debian gnu/Linux 9 (stretch) system. It is because Python binding had deleted in Debian gnu/Linux 9 (stretch) system. So I did not install libjasper-dev Python bindings.

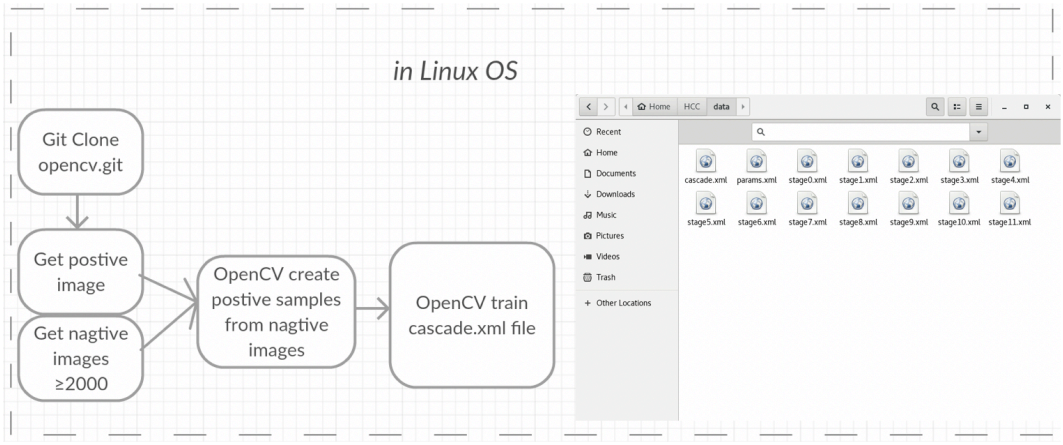


Figure 16. Trained cascade

From now, on the OpenCV trains haar cascade command should be used. Taking pictures of the box that is used in this project, a negative folder and positive folder to save negative images and positive images must be made. There should be more than 2000 negative images. This project has downloaded 3200 negative images.

Next a negative text file is created. The following step is to make some positive samples via opencv_createsamples command as shown in Figure 17.

```
boligao@Boli:~/HCC$ opencv_createsamples -img box.jpg -bg neg.txt -info pos/pos.txt -pngoutput pos -maxxangle 0.5 -maxyangle 0.5 -maxzangle 0.5 -num 3000
Info file name: pos/pos.txt
Img file name: box.jpg
Vec file name: (NULL)
BG  file name: neg.txt
Num: 3000
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 0.5
Max y angle: 0.5
Max z angle: 0.5
Show samples: FALSE
Width: 24
Height: 24
Create test samples from single image applying distortions...
Done
boligao@Boli:~/HCC$ opencv_createsamples -info pos/pos.txt -num 3000 -w 50 -h 30 -vec pos.vec
Info file name: pos/pos.txt
Img file name: (NULL)
Vec file name: pos.vec
BG  file name: (NULL)
Num: 3000
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Width: 50
Height: 30
Create training samples from images collection...
Done. Created 3000 samples
```

Figure 17. Create positive samples

In the end I trained my own haar cascade of box. The process of training is more than five hours in most cases. But the time is according to images' number and custom values. After 11 stages of training, the final cascade.xml file was finished.

```
boligao@Boli:~/HCC$ opencv_traincascade -data data -vec pos.vec -bg neg.txt -numPos 2500 -numNeg 1200 -numStages 12 -w 50 -h 30
PARAMETERS:
cascadeDirName: data
vecFileName: pos.vec
bgFileName: neg.txt
numPos: 2500
numNeg: 1200
numStages: 12
precalcValBufSize[Mb] : 256
precalcIdxBufSize[Mb] : 256
stageType: BOOST
featureType: HAAR
sampleWidth: 50
sampleHeight: 30
boostType: GAB
minHitRate: 0.995
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: BASIC

===== TRAINING 0-stage =====
<BEGIN
POS count : consumed   2500 : 2500
NEG count : acceptanceRatio    1200 : 1
Precalculation time: 5
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|   0.9972| 0.668333|
+----+---------+---------+
|   3|   0.9972| 0.668333|
+----+---------+---------+
|   4|   0.9956|    0.515|
+----+---------+---------+
|   5|    0.996| 0.430833|
+----+---------+---------+
```

Figure 18. Train haar cascade

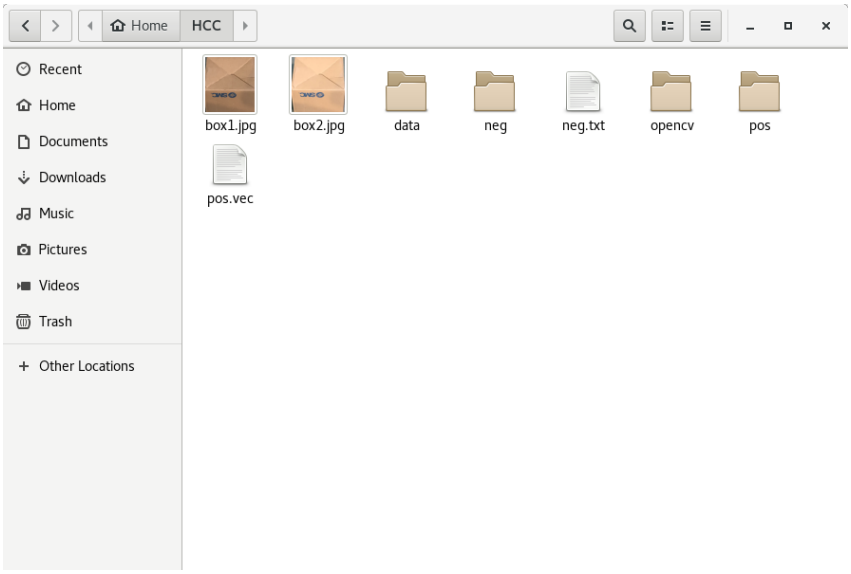


Figure 19. Whole haar cascade folder

Before building a human-following the NAO robot on NAOqi OS, it should be made sure that the NAO robot is equipped with speech recognition API. Terminal and ForkLift should also be installed in MacOS to connect to NAOqi OS. Because NAOqi OS in the NAO robot is not complete, it cannot use 'pip' commend in Terminal. A method to install a speech recognition API to the NAO robot is to transfer the whole SpeechRecognition file, which is installed in MacOS before, to the NAO robot by using ForkLift. It is easy and convenient to install a speech recognition API in Mac/Windows. Using Terminal and input 'pip3 install SpeechRecognition' command. Then the speech recognition is installed automatically and successfully. In the next, there are three steps should be followed, opening ForkLift, inputting NAO robot's IP address or host name, username and passport. For example, '192.168.1.118', 'nao' and 'root and run: $ ssh nao@192.168.1.118 in Terminal application.

Apart from this, the face and the object haar cascades should be transferred to NAOqi OS, the same as in Caffe model. And then, the first step will be done when transferring speech_recognition folder to the NAO robot. Next, whole files of the project are transferred to the NAO robot. The project should be executed under the folder of project by using 'python filename.py'.

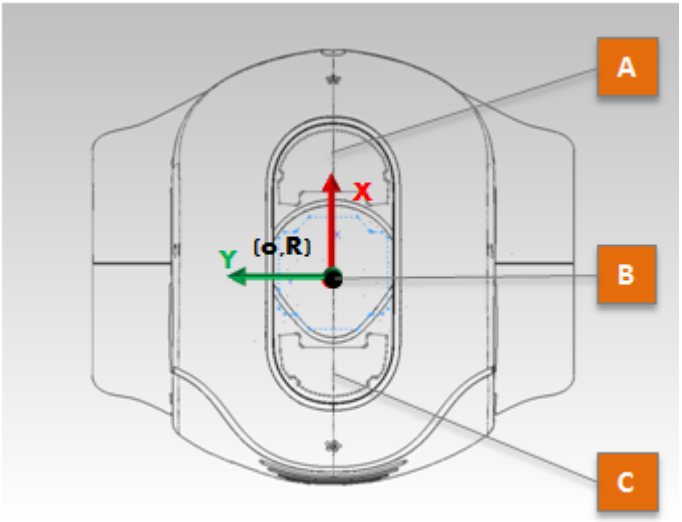


Figure 20. Head tactile sensor

At first, the user should touch the front head sensor which can trigger the ALMemory event to wake up the NAO robot. After this, the NAO robot will open the camera to record the video and use the ALAudioRecorder module to ask if you want to play. Google speech recognition will convert the audio file to a text file, which enables the NAO robot to recognize what the user said. If the answer is positive, the NAO robot will wake up and start capturing images to detect user's face via OpenCV. If answer is no, NAO robot will say good bye to the user. This project only accepts English as a communication language at the moment now. In the following step, the NAO robot will collect the image of the user's face and check if he recognizes the user' face.

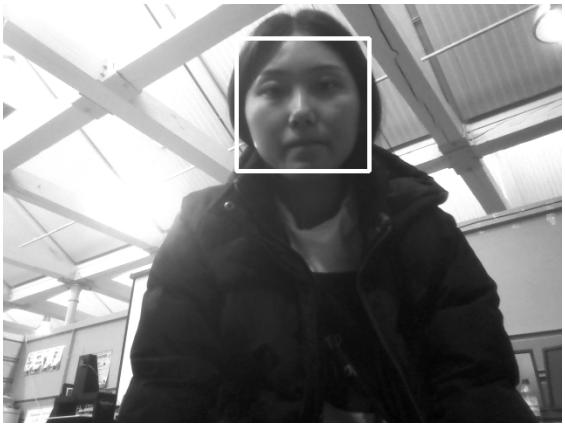


Figure 21. Face frame by OpenCV

After detecting the user's face and writing a face frame, OpenCV will calculate the focal length and distance to the user. Next, according to the distance and direction, the NAO robot will make a decision that uses the ALMotion module to move straight, turn left or turn right. When the distance is more than 30 inches, the NAO robot will continue walking towards the human. Otherwise, the NAO robot will stop.

```
let's start! by the way, touch my middle head if you want to stop!
ALTracker successfully started, now show your face to robot!
Use Ctrl+c to stop this script.
[[267  39 419 191]]
-----
Unsubscribe the video!
face detected
-------
[[413 364 466 417]
 [210 113 335 238]]
face detected 36.48 inches
face detected 272
towards
The Robot Move Command:  Pose2D(x=1.19209e-07, y=0, theta=0)
-------
-------
[[237 141 341 245]]
face detected 43.85 inches
face detected 289
towards
The Robot Move Command:  Pose2D(x=0.399999, y=4.76837e-07, theta=0)
-------
```

Figure 22. Move to user

After running the whole program, all functions will stop and shut down. The NAO robot will have a rest and then the recoded video will be writing the face frame and confidence level on video using Caffe model.

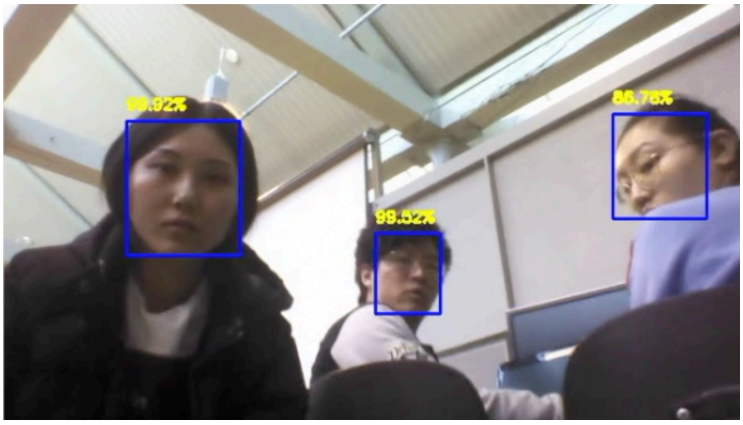


Figure 23. Confidence level

In addition, when the specific box is in front of the NAO robot, he will make judgment on box and avoid obstacles. While the box's distance to the NAO robot is less than 30 inches, the NAO robot will avoid it. However, this obstacle avoidance did not always work. If the box is too close so that the NAO robot did not have a full view of the specific box, then the NAO robot cannot recognize the specific box and he will continue to move towards the human being. In another situation, the box's haar cascade is not so accurate that the NAO robot only can

recognize a special side, based on the prepared positive images, or the NAO robot distinguished and avoided a wrong object.

In the last part of a human-following the NAO robot, the NAO robot will ask for a break after certain times of recognition. The user can decide to enable the NAO robot to continue or not.

5 CONCLUSION

The main goal of a human-following the NAO robot is to create a robot-human interaction environment. The widespread use of humanoid robots may increase productivity of numerous workloads, but the humanoid robots should imitate and follow human beings before using humanoid robots in a variety of human occasions. Human beings are still worried about the human-robot co-existence in social situations. However, when a robot becomes more and more like people, not only in appearance but also in actions, human beings would not worry about robot any more.

This project can be a reference tool for creating a healthy co-existing environment between humans and robots. The human-following NAO robot system considered the utilization of different technologies and tools. It utilized the NAOqi's API, Google speech recognition API which enables users to carry out simple communication with a NAO robot and OpenCV, Caffe model which makes face detection, obstacle avoidance and confidence level in the end. With the development of science and technology, all the modules can be replaced by more adaptive ones, especially in detecting face and object techniques.

The major challenge in this project was to train my own haar cascade. It was explained previously that much time should be used to train one haar cascade. The valued data adjusts higher, and more time is consumed. Therefore, the project's progress was too slow for a time. The haar cascade does not have a higher recognition rate beyond this. However, the final version is almost the best one.

6 FURTHER IMPROVEMENTS

For the future, it is necessary to update the accuracy rate of obstacle avoidance. OpenCV training haar cascade have been utilized for many years. Although OpenCV is updating now, this technique may not be advanced enough. In addition, training one object haar cascade would occupy much time. In regards to the error of obstacle recognition, a human-following the NAO robot project's object detection does not have a higher accuracy rate of recognition. So this part must improve in the future.

Adding voice recognition is to be considered. The subsequent face detections are all based on the first images captured by the NAO robot's camera. It is not a precise decision in this project. If a robot is put into a public area, the robot cannot make decisions or actions based on the first images. That is the reason why voice recognition should be taken into account. In pubic, the robot can be wake up by a clap or human's directive sound. A voice recognition sensor will recognize the direction of human and then the robot will move to the origin of sound.

In addition, setting as many different languages as possible so that the robot may be utilized by customers in different countries in the future is a task to be carried out.

REFERENCES

/1/ Human-Robot Interaction https://asegrad.tufts.edu/academics/explore-graduate-programs/human-robot-interaction

/2/ Dang Q K , Suh Y S . Human-following robot using infrared camera[J]. 2011.

/3/ What is Artificial Intelligence (AI) https://becominghuman.ai/what-is-artificial-intelligence-ai-4bde325e5462

/4/ Python introduction https://www.python.org/about/apps/

/5/ OS X for UNIX Users http://images.apple.com/media/us/osx/2012/docs/OSX_for_UNIX_Users_TB_July2011.pdf

/6/ The future of NAO https://www.softbankrobotics.com/us/NAO

/7/ The components of NAO https://www.softbankrobotics.com/emea/en/nao

/8/ The future of education is NAO https://www.softbankrobotics.com/us/NAO

/9/ NAOqi – Developer guide http://doc.aldebaran.com/2-5/index_dev_guide.html

/10/ NAOqi OS on a virtual machine http://doc.aldebaran.com/2-1/dev/tools/vm-intro.html

/11/ NAOqi Developer guide – Programming http://doc.aldebaran.com/2-1/dev/programming_index.html

/12/ Comparison of Top 10 Speech Processing APIs https://medium.com/activewizards-machine-learning-company/comparison-of-top-10-speech-processing-apis-2293de1d337f

/13/ You Shall Not Speak: Benchmarking Famous Speech Recognition APIs for Chatbots https://cai.tools.sap/blog/benchmarking-speech-recognition-api/

/14/ About OpenCV https://opencv.org/about/

/15/ Real Time Object Recognition with OpenCV | Python | Deep Learning – Caffe Model https://davidmatablog.wordpress.com/2017/12/05/real-time-object-recognition-with-opencv-python-deep-learning-caffe-model/

/16/ OpenCV API Reference https://docs.opencv.org/2.4/modules/core/doc/intro.html - api-concepts

/17/ NAO – video camera http://doc.aldebaran.com/2-1/family/robots/video_robot.html

/18/ NAO – Head joints http://doc.aldebaran.com/2-1/family/robots/joints_robot.html

/19/ Cartesian control http://doc.aldebaran.com/2-1/naoqi/motion/control-cartesian.html