VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Chaoyue Wang

BEHAVIOUR DESIGN OF NAO HUMANOID ROBOT

Playing Tic Tac Toe Game

Technology and Communication

2015

FOREWORD

This is my undergraduate thesis in the Degree Program of Information Technology, at Vaasan Ammattikorkeakoulu, University of Applied Sciences. So far, it has been two years since I study in Finland as a 2+2 exchange student. I would like to appreciate everyone, not only for helping me to complete of my final thesis, but also for these two years.

First of all, I would like to express my deepest appreciation to my supervisor, Dr. Yang Liu, for his guidance and instruction of my thesis as well as in almost all my advanced professional courses. In the past few days when I was working on my thesis, he always asked about what I have done and what challenges I have met, and also gave me much inspiration and constructive advice, so that I could finish my thesis at a high level successfully in a short time. Without his experience, I would not even have had a chance to explore in the field of NAO robot.

Moreover, I would like to sincerely thank all the teachers and staff who is working at Vaasan Ammattikorkeakoulu, including Dr. Chao Gao, who taught me telecommunication. Dr. Ghodrat Moghadampour, who taught me java and other programming languages. Mr. Jani Ahvonen, who taught me embedded system design. Mr. Santiago Chavez Vega, who always take care of me. Mr. Antti Virtanen. Mr. Jukka Matila, Dr. Smail Menani and all the other staff who have taught me with their patient and great knowledge in their professional field. Thanks to all the help and guidance, that has made a deep impression on my life.

Last but not least, I would like to take this opportunity to thanks my parents who gave me their concern and encouragement as much as possible, so that I could grow up like a happy little bird. Even during the time of completing the thesis, it was my parents who gave me power to conquer everything. Thanks also all my friends in the robot lab, including Xia Minxue, Sun Xiao, Zhou Ziye, Liao Ke, Li Xiuyang, for the great study atmosphere and working environment. Hope they can obtain great achievement in the future.

Vaasa 17 September

Chaoyue Wang

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Chaoyue Wang
Title	Behaviour Design of NAO Humanoid Robot: Playing Tic Tac Toe Game
Year	2015
Language	English
Pages	55
Name of Supervisor	Yang Liu

This thesis was written about my final project on the behaviour design of NAO humanoid robot playing Tic Tac Toe game. There are two parts in this thesis, the analysis of vision system and game strategy of Tic Tac Toe with NAO robot. In this thesis Tic Tac Toe game is used as an example, another game can be used in its stead. This thesis mainly focuses on the communication between the human being and the NAO robot, so they can play together, in this case the NAO robot could be improved to take care of child which seems like a promising research.

In the vision module OpenCV was used as a significant tool, Hough line transform and Hough circle transform were used in vision module to detect lines and circles during the project running. Priorities were set to the NAO's movement applied in the game strategy module. Key frames in timeline were used to make the NAO's animation, therefore the robot can move and draw.

This project was completed with the Python language under Windows 7 operating system and NAOqi 2.1 operating system. Also for the 5th generation NAO robot was used for developing this project. The implementation methods were planning, developing, debugging and testing.

In a word, NAO is becoming more and more popular for home use, this project is a good start for users get familiar with NAO.

Key words: NAO robot, vision system, game strategy, animation design

CONTENTS

LIST OF ABBREVIATIONS

PC	Personal Computer
RoboCup	Robot Soccer World Cup
V5	Version 5
V	Volt
Ah	Ampere Hour
DCM	Device Communication Manager
SDK	Software Development Kit
IP	Internet Protocol
3D	Three Dimension
DHCP	Dynamic Host Configuration Protocol
LPC	Local Procedure Call
RPC	Remote Procedure Call
OpenCV	Open Source Computer Vision Library
Fps	Frame per second
BSD-licensed	Berkeley Software Distribution license
PNG	Portable Network Graphics
VGA	Video Graphics Array
FABRIK	Forward and Backward Reaching Inverse Kinematics

LIST OF FIGURES AND TABLES

1 INTRODUCTION

1.1 Purpose

This thesis introduces the behaviour design and implementation details of the NAO humanoid robot about playing the Tic Tac Toe game with a human player on a paper board. The thesis mainly concerned about three aspects, the first part is the vision processing of finding the 3x3 board and also circles, lines on this board. The second part is about game strategy. In the third part animation design with timeline will be included.

1.2 Overview Structure

This thesis consists of eight chapters. The first chapter gives the introduction of the thesis and background including the NAO robot, and Tic Tac Toe game, and motivation. The second chapter shows the overall structure of this application, it explains the whole project by a flowchart and also gives a brief introduction of each module. The third chapter introduces the communication module in details. The fourth chapter describes the vision module by explaining the vision hardware and how to use OpenCV to complete this vision part. The fifth chapter presents the strategy module including the algorithm of the Tic Tac Toe game and the NAO robot behaviour design. The sixth chapter introduces the implementation details. The seventh chapter is the overview of future research. The ninth chapter is the conclusion of this project and thesis.

1.3 Introduction to Nao Robot

1.3.1 History

The NAO humanoid robot was developed by Aldebaran Robotics which is a France company in 2006 is a programmable, open architecture robot. NAO is a 58-cm tall humanoid robot. This robot is able to move, recognise, hear and even talk. Before home use, NAO became a famous humanoid robot in the world of education. In more than 70 countries, it has been used in computer and science classes, from primary school through to university. /2/

Here is a picture of the humanoid robot—NAO.

Figure 1. NAO robot /1/

1.3.2 Key Components of NAO Robot

NAO is a small character with a unique combination of hardware and software: it consists of sensors, motors and software driven by NAOqi which is a dedicated operating system. NAO has following key components/3/:

- NAO has the body with 25 degrees of freedom (DOF) whose key elements are electric motors and actuators, so that it can move freely.

- NAO has a sensor network with two cameras, four directional microphones, sonar rangefinder, two IR emitters and receivers, one inertial board, nine tactile sensors and eight pressure sensors.

- NAO has various communication devices, including voice synthesizer, LED lights, and 2 high-fidelity speakers, so that it can hear, speak, and blink the light.

- The main CPU is Intel ATOM 1,6ghz CPU which is located in the head that runs a Linux kernel and supports Aldebaran’s proprietary middleware (NAOqi).

- NAO has two CPU, the second CPU is located its the torso.

- NAO has a 48.6-watt-hour battery that provides NAO with 1.5 or more hours of autonomy, depending on usage.

1.3.3 Feature of NAO Robot Vision 5

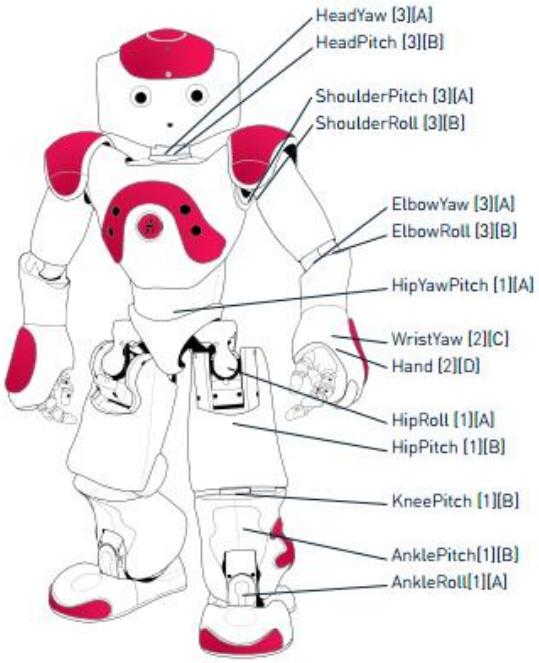

Figure 2. Joints for NAO Robot /4/

NAO Robot V5 was used to build this thesis. The basic constructions of NAO V5 is shown in Table 1. What should be noticed is that a special plastic material was made for its body and it has a 48.6-watt-hour battery which can be used up to 1.5 or more hours of autonomy, for example 90 minutes for normal use and 60 minutes for active use, and more than 90 minutes for rest.

As shown in Figure 2, this robot has 25 joints in total and also many kinds of sensors, such as nine tactile sensors and eight pressure sensors.

Here are the basic constructions of NAO Robot V5 Evolution./4/

Table 1. Construction of NAO Robot V5 Evolution/4/

NAO V5 Constructions	
Height (mm)	574
Depth (mm)	311
Width (mm)	275
Weight (kg)	5.4
Autonomy	1.5 hours for active use
Degrees of freedom	25
CPU	Intel Atom @ 1.6 GHz
Material	ABS-PC/PA-66/XCF-30
Built-in OS	NAOqi 2.0 (Linux-based)
Compatible OS	Windows, Mac OS, Linux
Programming Language	Python, C, C++, Java, MATLAB, .NET, Urbi
Battery	Build in Rechargeable battery
Connectivity	Ethernet, Wi-Fi

In the figure below the dimension of V5 on NAO robot can be seen, from both front view and top view.

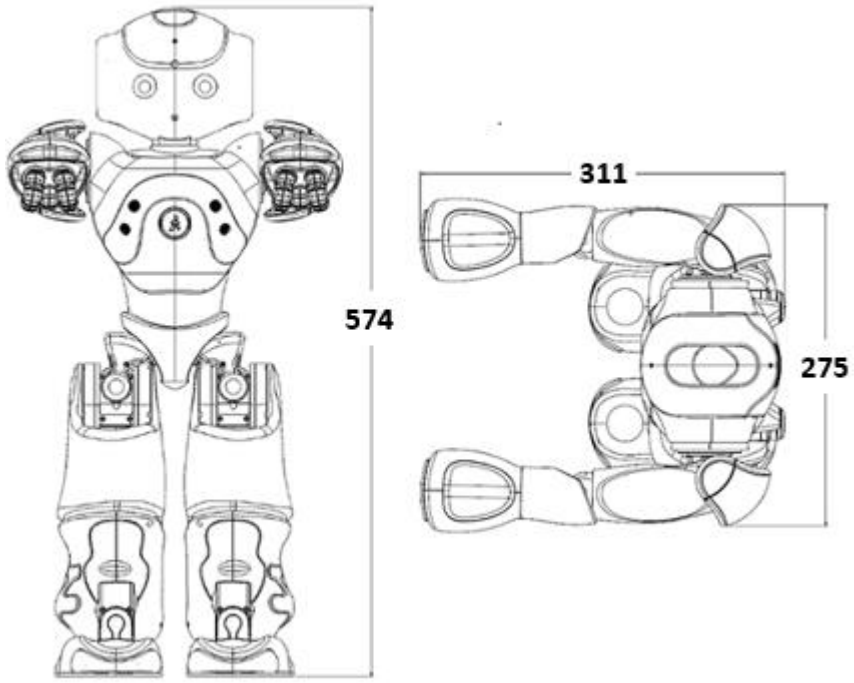

Figure 3. NAO Robot V5 dimension/5/

1.4 Background of Bothnia Humanoid Robot Team

Bothnia Humanoid Robot Team was organized by the original Bothnia SSL Robot Team. Bothnia SSL Robot Team has participated in RoboCup world competition for several times and they also win a prize in the competition. Nowadays, the Bothnia Humanoid Robot Team is set up and led by Dr. Yang Liu. In 2014, Vaasan ammattikorkeakoulu, University of Applied Sciences bought two NAO Robot V5 for educational and research use. So far some applications have been already made by the previous team players, such as NAO robot play Candy Crush Saga game and picking up a ball and throwing it to a box.

1.5 Software of NAO Robot

In order to complete this project, the following software was in this project, Choregraphe, Webots, Monitor and also an embedded software NAOqi.

1.5.1 Choregraphe

Figure 4, shows the interface of this software. Choregraphe is a multi-platform desktop application, with which animations, behaviours and dialogs can be created. These can be tested on a simulated robot or directly on a real one. In this software the robot can be monitored and controlled. Choregraphe behaviours can be enriched with the Python code./6/

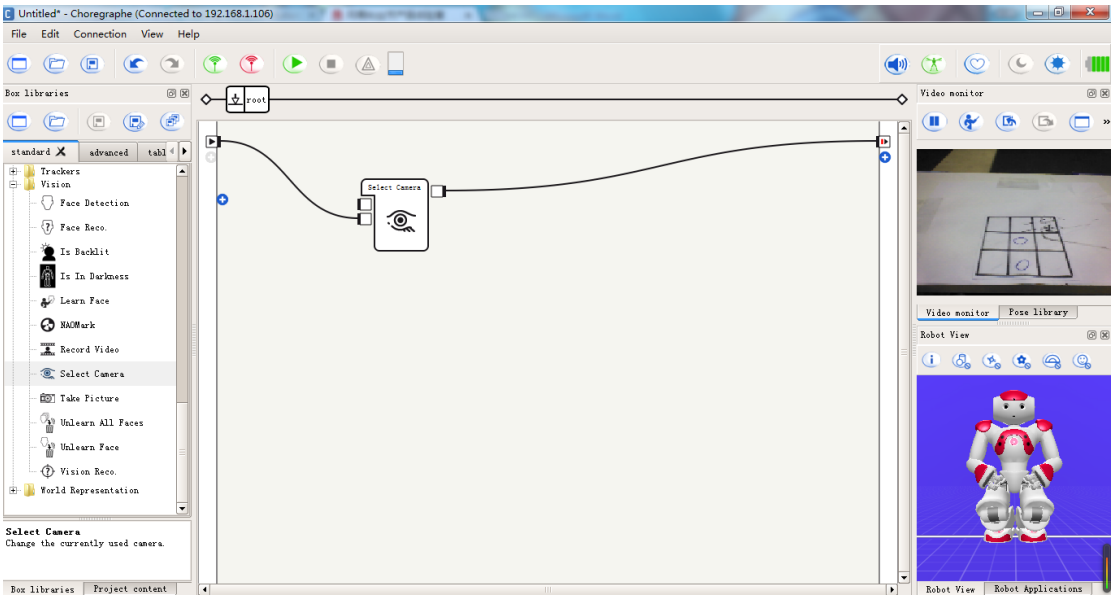

Figure 4. Choregraphe Software 2.1.3 desktop interface

1.5.2 Webots

Webots is a simulator for programming, simulating and modelling many kinds of robots. Webots for NAO is a simulation software. It offers a safe place to test behaviours before playing them on a real robot./7/

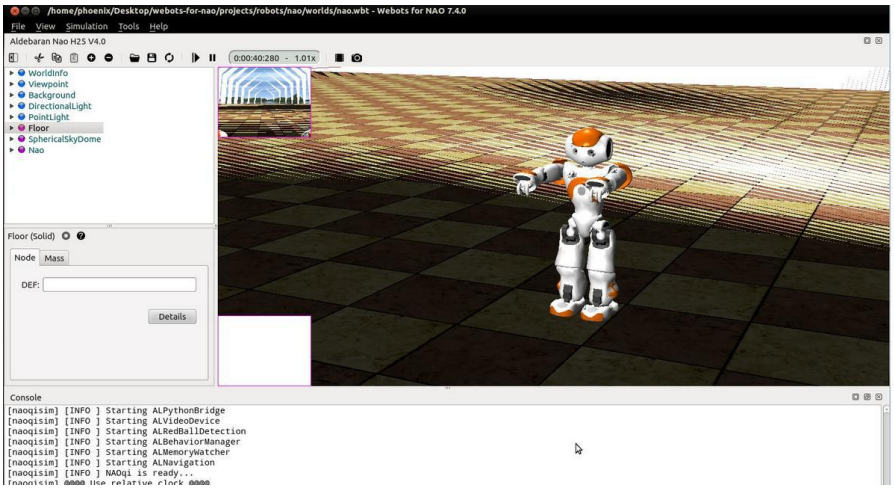

Figure 5. Webots Software 8.0.3 desktop interface

1.5.3 Monitor

Monitor is used to give a direct access to its camera settings and give an elementary feedback from the robot. There are three plug-ins available, camera viewer, laser monitor, and memory viewer./8/

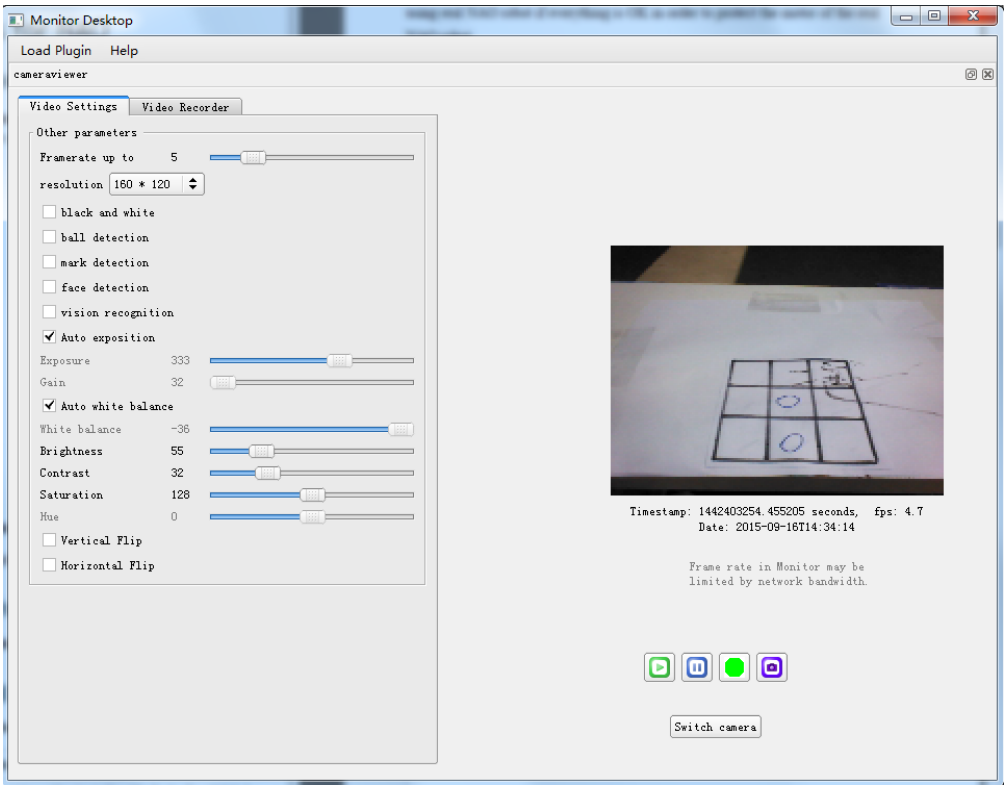

Figure 6. Monitor Software 2.1.3 desktop interface

1.5.4 NAOqi

NAOqi is an operating system on the NAO robot based on natural interaction and emotion. NAOqi is the main embedded software running on the robot and controls it under OpenNAO distribution. It can also run on the computer in order to test the code on a simulated robot. /9/

1.6 Programming the NAO Robot

The NAOqi API is currently available in at least eight languages. Apart from some minor language-specific differences, the API is mostly the same across all languages, allowing to bring knowledge form one language to another. In this thesis the application was programmed with the Python language. Python is the second most complete framework, this framework also allows the running of the embedded code. NAOqi modules can be created in Python, and the notification from other modules used. Here is a sample Python code, shown in Figure 7. In this thesis Python was used inside the Choregraphe boxes.

```
from naoqi import ALProxy

tts = ALProxy("ALTextToSpeech", "<IP of your robot>", 9559)
tts.say("Hello world from python")
```

Figure 7. Sample code in Python language

Figure 8 shows the panels display of Choregraphe software. Area A is the project content panel. The Project content panel displays the project properties and all the files attached to the current project. Area B is the box libraries panel, in this panel any of the boxes contained in a box library can be dragged and dropped onto the flow diagram panel and a behaviour or a box enriched. Area C is the flow diagram panel where NAO's behaviours can be composed. This panel is a group of boxes linked with each other and with at least an input. Area D is the pose library panel and video monitor panel, for the video monitor panel cannot be tested on a simulated robot, unless in a virtual world. The pose library panel displays specific timeline boxes containing the NAO preset positions. Area E is the robot view and robot applications panel. The robot view displays in the 3D view the robot

Choregraphe is connected to, it allows the checking and modifying the joint values (and then move the limbs) using limb properties. The robot applications panel displays the applications available on the connected robot. /11/

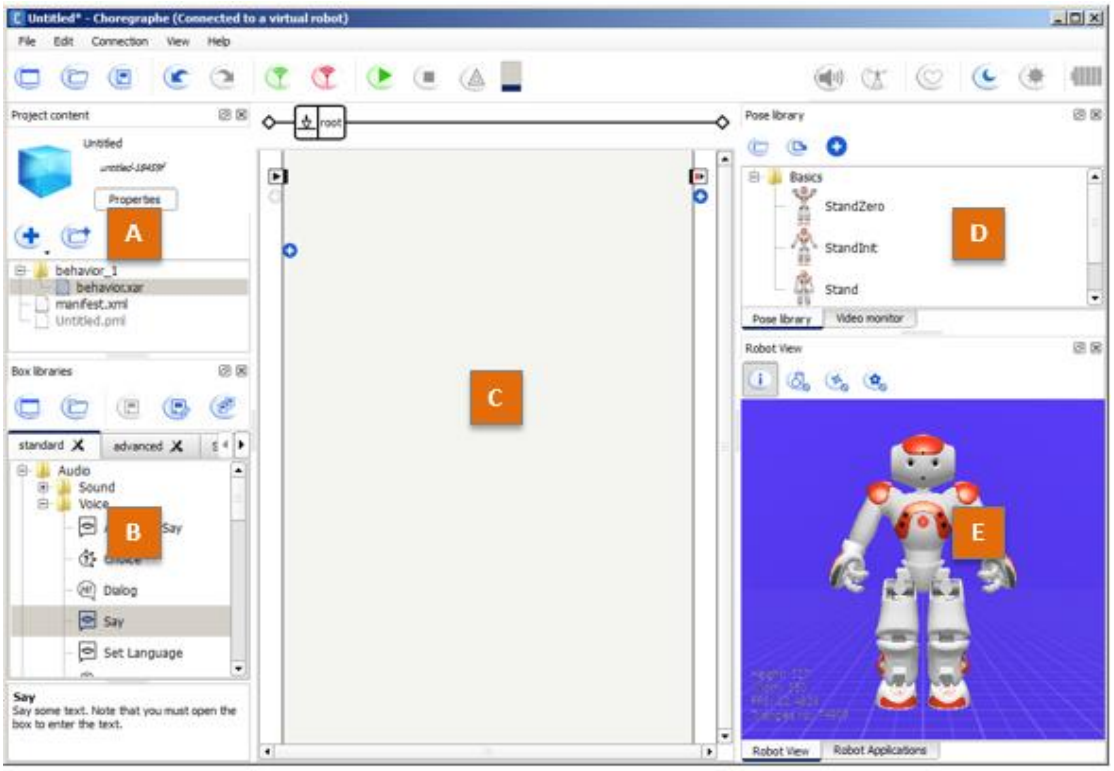

Figure 8. Programming using Choregraphe /10/

1.7 Background of Tic Tac Toe game

The Tic Tac Toe game is a paper-and-pencil game for two players, X and O , who take turns marking the spaces in a 3x3 grid. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game.

An early variant of Tic Tac Toe was played in the Roman Empire, around the first century BC. It was called Terni Lapilli and instead of having any number of pieces, each player only had three, thus they had to move them around to empty spaces to keep playing. The game's grid markings have been found chalked all over Rome.

Nowadays, many Tic Tac Toe games can be found online, so that people can play with a PC. In this application, a human player can play the Tic Tac Toe game with the NAO

Robot. The human player using a circle as his chess, and instead of writing a cross, NAO will only write a line here to represent his chess. Because this is only a mark for NAO, it does not matter if it is a line or a cross. In future works NAO can write an 'X' when playing this game on a board.

1.8 Motivation

A humanoid robot is becoming more and more popular, the robot is not only for research purpose any more, in the future, the robot can plays a significant role at homes. The NAO robot is designed for use by academics and schools wanting to work on robotics projects. Besides it is not difficult to start working with NAO, so in this case it was a great chance to get start on working with the robot. For a long term purpose the Bothnia robot team can also participate in the world RoboCup using NAO robot.

As for the Tic Tac Toe game, it is a simple paper game. The rules of this game are easy and no other materials are required. This game is also popular worldwide, almost every kids has played it before. For algorithm purposes, the code for NAO's movement was done in the project module, so in this case it helped me a lot.

2 OVERALL STRUCTURE

This chapter discusses the structure of the whole project, as well as gives a brief introduction to each module and flow chart of the whole project.

2.1 Structure of the Whole Project

In this project, the whole system has five main modules, they are vision module, behaviour module which belong to the NAO robot part., and also communication module, as well as computer part which contains controller PC and strategy module.

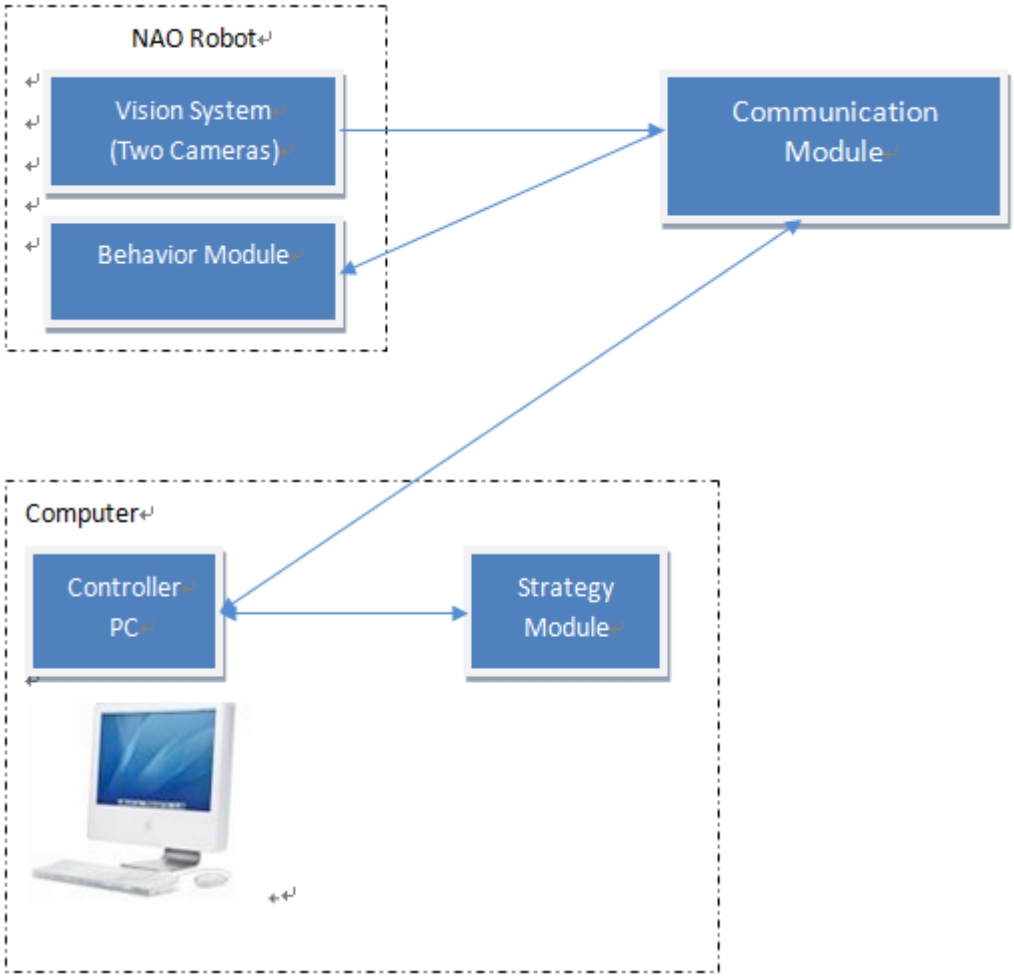

Figure 9. Structure of the whole project

2.2 Introduction to Modules

- Vision system

  Vision system contains two cameras as a hardware part, and as for software part it has OpenCV. In this system to complete image acquisition and return it to NAOqi as well as image processing are needed. To detect lines and circles are needed by using Hough line transform and Hough circle transform functions

- Communication system

  The communication system is an important part between NAO humanoid robot and PC. Distributed tree and communication and NAOqi process including broker and proxy are discussed in this part.

- Behaviour

  In this thesis project, NAO's behaviour design is only about his arms. When the game starts, NAO is in a squat position, and the arm movement including close and open hand are all made by timeline. How to achieve it, will be explained later in chapter implementation module.

- PC controller

PC controller is used to call all the strategies and methods which is designed in the project.

- Strategy

  Algorithm and decision making part are included in strategy. More about this part will be described in detail later in chapter strategy module.

2.3 Flowchart of the Project

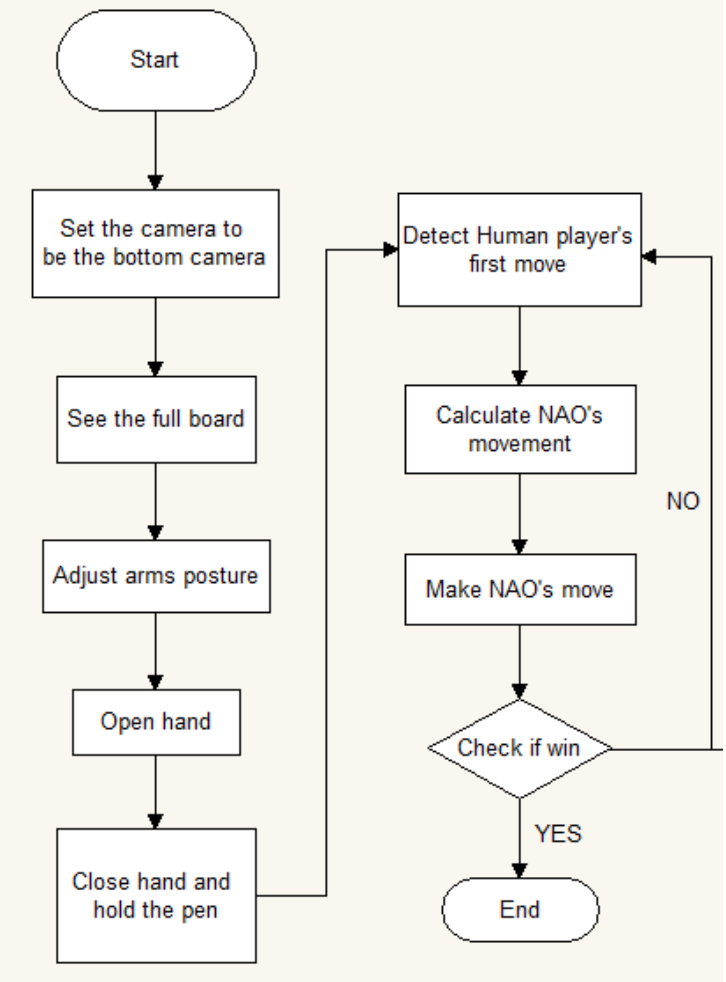

Figure 10. Flowchart of the whole project

Figure 10 shows the flowchart of the whole project. First of all the camera was set to be the bottom one, because using this camera NAO can see the area closer to itself, so that is easy for NAO to draw in the next part. And then timeline was used to set the prepare position for NAO, it will move its arms to the pre-set position as well as open its hand and say ”please give me a pen” at the same time. After NAO got the pen , the human player will draw a circle as his first move, then NAO will get a new image and detect the human player’s move. After this processing NAO will make a decision. The it is NAO’s turn to make a move. After making a move, NAO will get another new image, then it will check if it wins or not., If it wins then the game ends, if not, then it will play again by detecting human player’s move procedure.

3 COMMUNICATION MODULE

The communication module is used for communicating between the controller PC and the NAO humanoid robot. There are two ways that NAO can connect to PCs, one is using an Ethernet cable, and the other one is wireless connection using Wi-Fi. The router needed to be set which supports the DHCP function to make it possible to use the wireless network.

By using the IP address which NAO connected, the NAO robot and the remote PC can send files between each other. This services are called FTP, File Transfer Support.

3.1 Distributed Tree and Communication

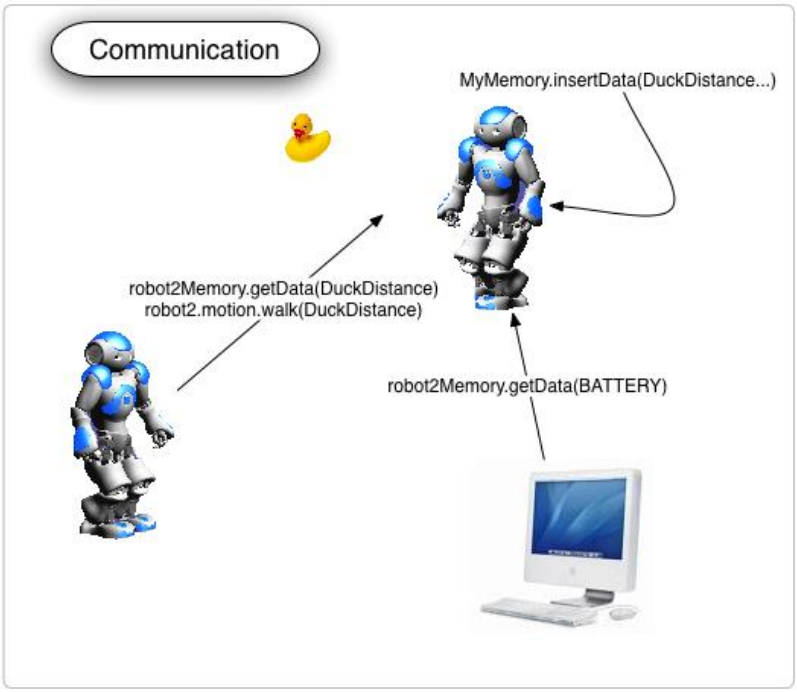

Figure 11. Communication between NAO robot and PC

A standalone executable on the tree of robot, tree of processes, tree of modules could be a real time application. An executable is connected to another robot with an IP address and port, it is available in almost the same way like a local method that all the API methods from other executables. The choice is made by NAOqi between the fast direct call (LPC) and remote call (RPC)./12/

3.2 NAOqi Process

It is known to all that Choregraphe software can connect to the real robot by the use of IP address and the port. For Python, C++ and other programming languages, there is another way for them connect to the robot, by the use of embedded software NAOqi, it works as the user and communication of NAO system.

NAOqi is the programming framework used to program NAO, and it was designed to satisfy the requirements needed in robotics. Main features include parallelization, resource management, synchronization, and event, and it allows communication between other modules like motion, audio, and video. NAOqi also enables information sharing and programming through ALMemory and communication between Homogeneous Modules like motion, audio, and video that serve other roles.

NAO's overall operation is managed by the NAOqi Framework as the user and system communicate. DCM (Device Communication Manager) manages the communication between NAO devices like the actuator and sensors. DCM is a part of the NAOqi system. It is a NAO software module and manages the communication of all the devices (board, sensors, actuator, etc) excluding the cameras and sound.

First the components of distribution and the role of the module are defined and some of the ways they interact with one another are explained. Figure 12 below shows NAOqi's framework structure.

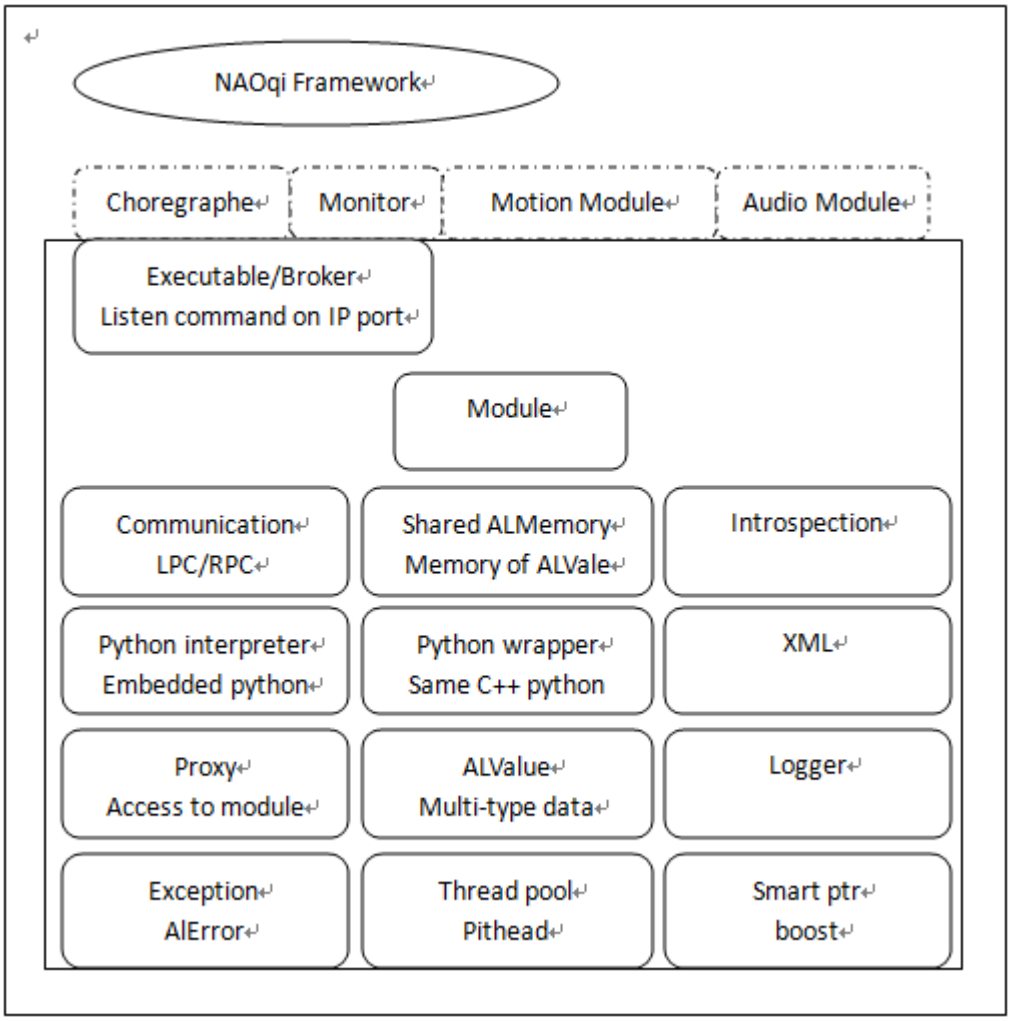

Figure 12. NAOqi's framework structure

NAOqi framework works by having Choregraphe, Monitor, Motion module, and Audio module pass information to each other. NAOqi is executed by having Broker deliver information and commands. The following explains the different elements that configure the NAOqi framework.

- Module: Module is both a class and library that uses the function and API defined in ALModule to obtain information or control regarding each module.

- Communication: Communication uses Local Procedure Call or Remote Procedure Call to connect to NAO and exchange information.

- ALMemory: ALMemory is the robot's memory. Any module can use or read this data and can monitor events. It can be called when an event occurs. ALMemory is an array of ALVaule.

- Proxy: All Aldebaran module have been modularized. Rather than directly referencing other module files, the user can request the Proxy to find the corresponding module. If the module doesn't exist, and exception occurs. The user can tell the corresponding function or module through the Proxy from two independent brokers, mainBroker(local call) and myBroker(remote call).

- ALValue: In order to be compatible, some NAOqi modules or methods are saved as a specific data type in ALValue.

3.2.1 Broker

Broker is running on the robot for the NAOqi executable. When the broker starts, it loads a preferences file called autoload.ini and that file defines which libraries it should load. Each library contains one or more than one modules that advertise their methods by using the broker.

The broker is an object also the broker provides lookup services so that any module in the tree or across the network can find any advertised method. Loading modules form the modules attached to a broker and a tree of methods to the attached to modules.

The broker also provides two main roles:

- The broker provides a directly accessing services: which allows the user to find methods and modules.
- The broker provides network access: which allows outside the process calling the methods of attached modules.

The brokers do their work transparently, which allows the code to be written that will be the same for calls to "local modules" (in the same process) or "remote modules" (in another process or on another machine).

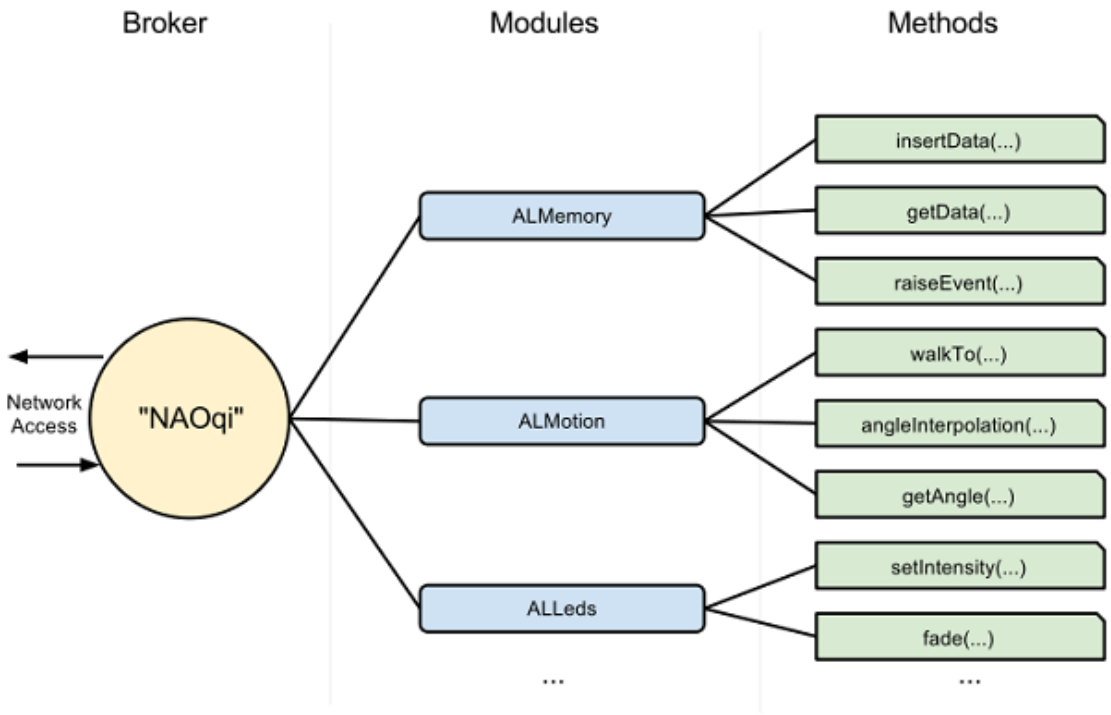

Figure 13. The Broker/12/

3.2.2 Proxy

A proxy is an object that will behave as the module it represents.

For instance, a proxy is created to the ALMotion module, an object will be obtained containing all the ALMotion methods.

To create a proxy to a module, (and thus calling the methods of a module) there are two choices:

- Using the name of the module. In this case, the code run and the module to connect to must be in the *same* broker. This is called a *local* call.
- Using the name of the module, and the IP and port of a broker. In this case, the module must be in the corresponding broker./12/

4 VISION MODULE

In order to make the NAO robot see the board fully and clearly and know each movement during the game, the vision module is very important in the whole project. The vision module needs to detect lines and circles so that NAO can get the return value and make a decision to write a line at a proper place.

The algorithm for this module is based on the recognition of lines and circles. Because the game board is drawn on a paper by lines and make it a 3x3 grids. Besides the human player's chess is to draw a circle, so NAO only needs to detect these two symbols. The vision module contains two parts. The first part is about image acquisition and raw data acquisition. This part is compiled and executed on OpenCV which is an open source library that contains a large number of computer vision algorithm. And the image results are stored in a local workspace. The second part is about identifying board especially the chess position. This part compiles in a local PC and the result of output recognition is used as a list for the next game strategy module.

4.1 Flowchart of Vision Module

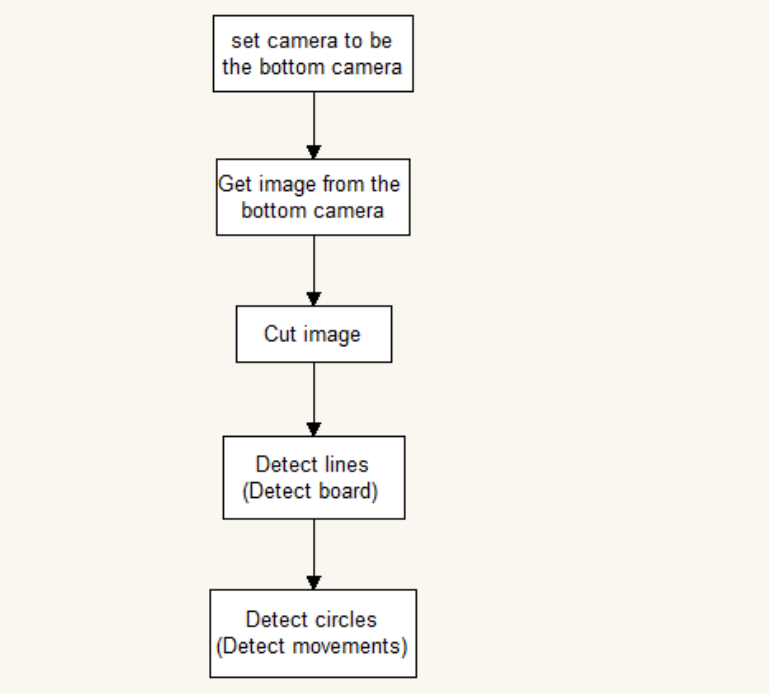

Figure 14. Flowchart of Vision module

4.2 Hardware Part of Vision Module

In this section the definition of the 3x3 game board and the specification of NAO robot's camera are discussed. All information will be presented in details below.

4.2.1 Definition of the Game Board

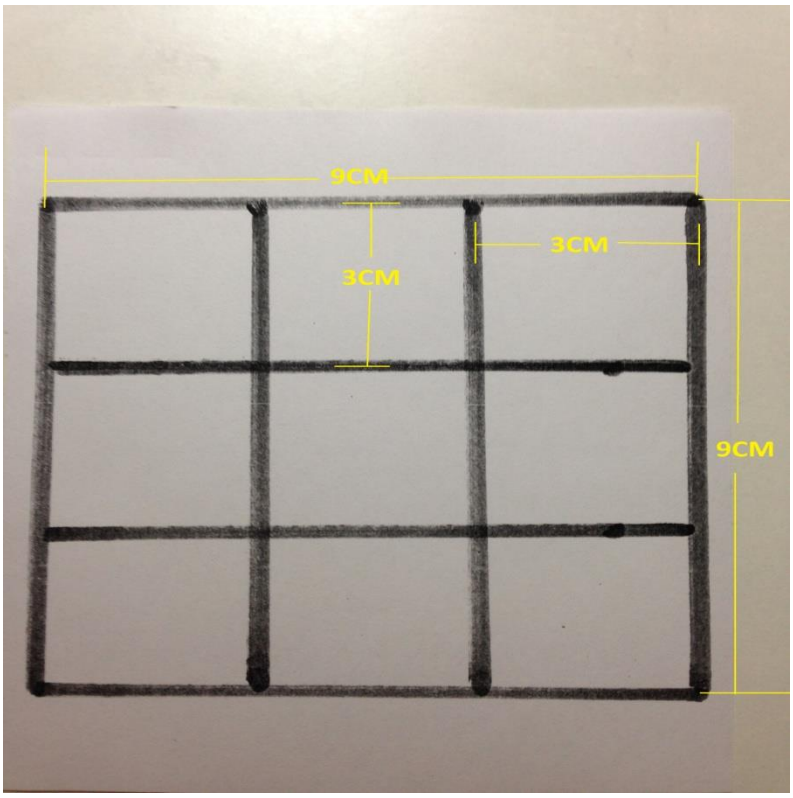

Figure 15. 3x3 Game Board

The 3x3 game board shown in Figure 14 was used in this project. The whole board is a 9cm x 9cm square, and each grid is a 3cm x 3cm square drawn in black pen.

4.2.2 Technical Overview of Cameras

Two identical video cameras are located in the forehead of NAO robot. These two cameras provide a up to 1280x960 resolution at 30 frames per second. They can be used to identify objects in the visual field such as recognizing objects, walking followed NAO marks, and the bottom camera can be used to ease NAO's dribbles. The top camera is

located in the centre of NAO's two eyes in his forehead and the location of the bottom camera is as his mouth.

Figure 16 and 17 generally introduce the range of vision field of view from both side view and top view. In this project the camera was chosen to be the bottom one, in order to make NAO draw on the game board easily.

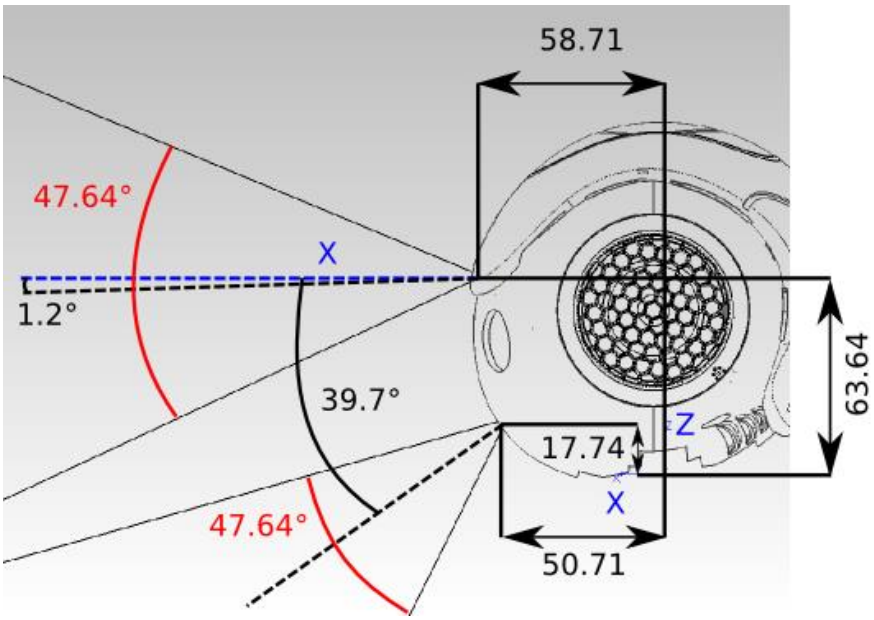

Figure 16. Side view of cameras/13/

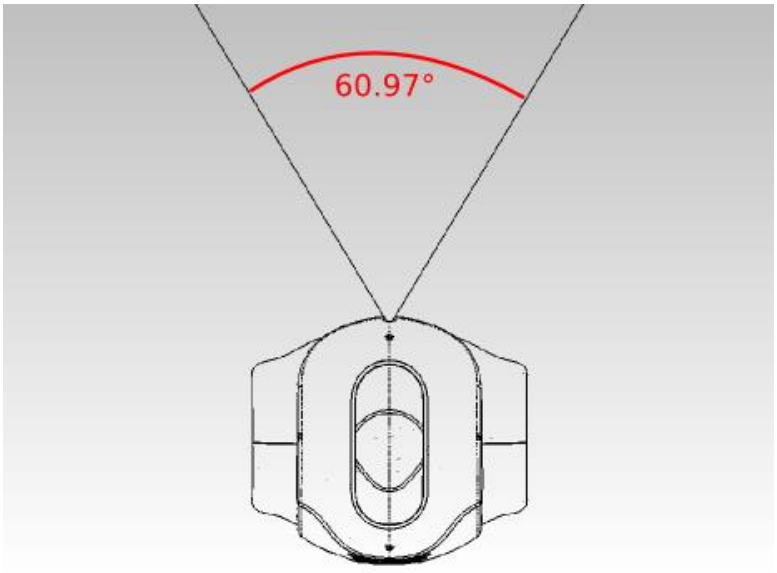

Figure 17. Top view of cameras/14/

4.2.3 Data Sheet of Camera

Here is the table for the data sheet of the NAO robot video cameras, which illustrates the basic information of the robotic camera that was used to in this thesis.

Table 2. Data Sheet of Cameras/15/

Camera	Model	MT9M114
	Type	SOC Image Sensor
Imaging Array	Resolution	1.22Mp
	Optical format	1/6 inch
	Active Pixels(HxV)	1288x968
Sensitivity	Pixel size	1.9µm*1.9µm
	Dynamic range	70 dB
	Signal/Noise ratio (max)	37dB
	Responsivity	2.24V/Lux-sec (550 nm)
Output	Camera output	1280*960@30fps
	Data Format	(YUV422 color space)
	Shutter type	Electronic Rolling shutter (ERS)
View	Field of view	72.6°DFOV (60.9°HFOV,47.6°VFOV)
	Focus range	30cm ~ infinity
	Focus type	Fixed focus

4.3 Brief introduction of OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision, originally developed by Intel research centre in Nizhny Novgorod (Russia), later supported by Willow Garage and now maintained by

Itseez. In addition, OpenCV stresses on computational efficiency and pay close attention to on real time applications. In this project, Python language and OpenCV library were used for the image processing.

4.4 Image Acquisition

Choregraphe 2.1 was used in this project to monitor the real-time image of NAO's camera shown in Figure 4 above. In order to process necessary vision information, an image which includes enough needed information must to be acquired. Here is a piece of code which shows the process of image acquisition, use Figure 18.

```
def saveImage(IP, PORT)
    camProxy = ALProxy("ALVideoDevice", IP, PORT)
    resolution = 2  #VGA
    videoClient = camProxy.subscribe(python_client, resolution, 5)

    t0 = time.time()

#get the camera image

    camProxy.getImageRemote(videoClient)
    t1 = time.time()

#time the image transfer
    print "acquisition delay", t1 - t0
    camProxy.unsubscribe(videlClient)

#image return and save it as PNG using ImageDrew package
#get the size and pixel array of image
    imageWidth = naoImage[0]
    imageHeight = naoImage[1]
    array = naoImage[6]

#save image
im.save("camImage.png", "PNG")
```

Figure 18. Code for image acquisition

As shown in Figure 18, two objects were created first, they are called as ”camProxy” and ”ALProxy”. And also modules named as ”ALVideoDevice” was specified, by using this module image from video source can be received. The resolution of this image was set to be VGA which means the number of pixel is 640*480, for this project higher

resolution value is not needed. With a higher resolution it takes more time to process the image. By using a method called ”getImageRemote” an array can be get which contains all the image data. And after getting the image this image was saved as a PNG image on local computer.

4.4.1 Position

The position of NAO in this project is shown in Figure 19. The NAO robot is in the rest position before the project starts, and there is a desk in front of it which holds the game board. For the whole project NAO is squatting and it only moves its arms. In this position Nao can rest its legs and is also more stable than standing up. Also it can detect the full board and see each move clearly in this position.

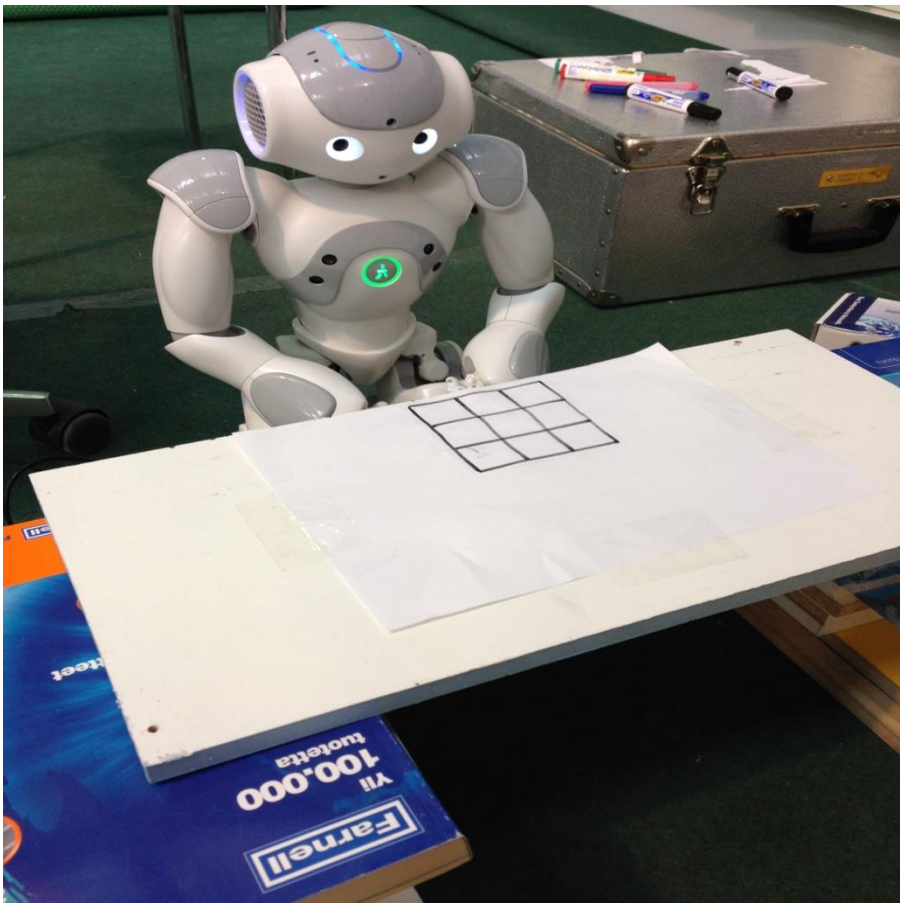

Figure 19. NAO’s working position

4.5 Cut Image

When the camera was set to be the bottom one, and the NAO humanoid robot looks down to the game board, it will see a part of floor. Thus, it will influence the NAO robot to detect the game board somehow. At the same time, when playing Tic Tac Toe with NAO, it will always use the bottom camera. So cutting the image will be the best solution for avoiding the noise of picture from the bottom camera.

Figure 20 is the image before cutting and this image captured from the NAO video monitor.

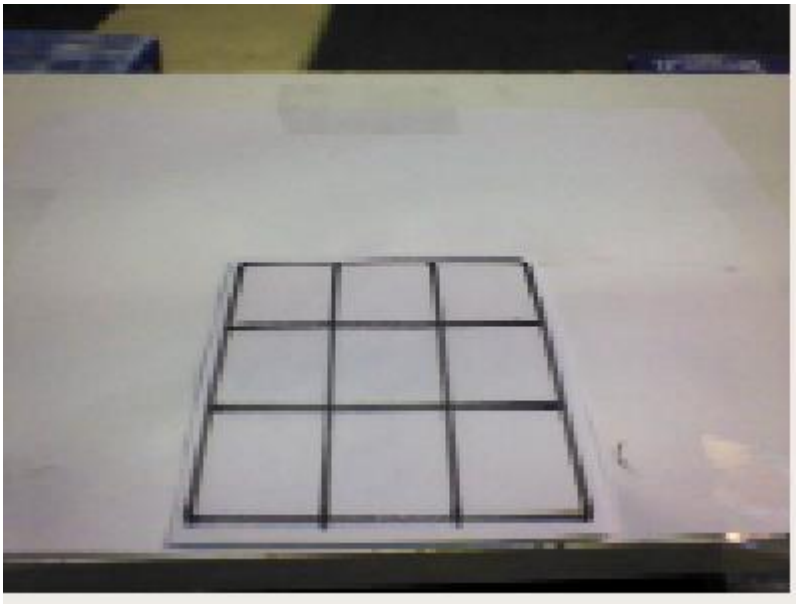

Figure 20. Image before cutting

Here is the code for image cutting, with running the code cut image is obtained .

```
def cutImage(picture_name, y1, y2, x1, x2):
#read image by using opencv
img = cv2.imread(picture_name)
#cut image
img = img[y1:y2, x1:x2]
#save the cutting image
cv2.imwrite(picture_name, img)
```

Figure 21. Code for image cutting

A picture of cut image is shown below, where the unneeded part is cut from previous picture, and the new image is easier for NAO humanoid robot to detect.

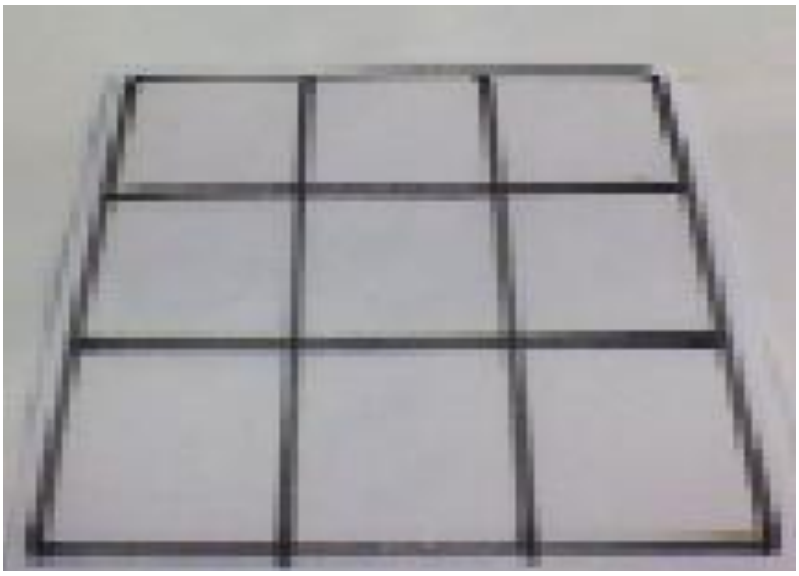

Figure 22. Image after cutting

4.6 Detection of Lines and Circles

In the project an important part is that NAO can detect lines and circles because based on that NAO can play Tic Tac Toe game. To detect lines is to know the game board, and to detect circles is to find out the movement of the human players so that NAO can make his own decision and next move.

The Hough transform is a feature extraction technique that used in analyzing image, vision of the computer, and processing digital image. The purpose of the technique is to find imperfect instances of objects which within a certain class of shapes by voting. /18/

4.6.1 Hough Line Transform

The Hough line transform is an algorithm that can be used to detect straight lines in an image. It is recommended to pre-process the image, such as converting this image to a binary image first in order to applying this transform,.

A line in the image space can be expressed by two variables. For example if the user put the line in the Cartesian coordinate system, then the parameters are (m, b). If the user put the line in polar coordinate system, then the parameters are (r, θ). For Hough Transforms, the lines are expressed in the *Polar system*. Hence, a line equation can be written as:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right) \quad (1)$$

By deforming the formula a new line equation can be get:

$$r = x\cos\theta + y\sin\theta \quad (2)$$

Where r is the perpendicular distance from the origin to the line, and θ is the angle formed by this horizontal axis and perpendicular line (This representation is used in OpenCV). Check below image:

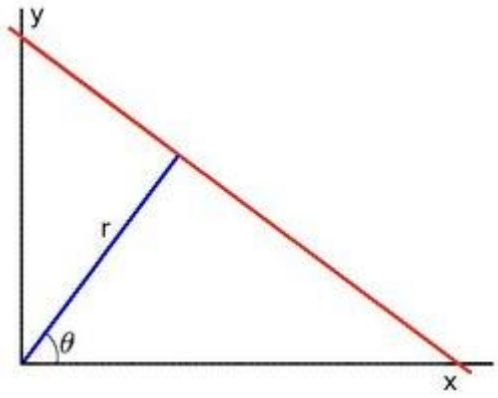

Figure 23. Polar coordinate system

The family of lines going through a point (x0; y0) can be found by substituting x_0 and y_0 in equation below:

$$r_\theta = x_0\cos\theta + y_0\sin\theta \quad (3)$$

Each pair (r_θ, θ) on behalf of each line that passes by (x_0, y_0). If for a given (x_0, y_0) plot the family of lines that goes through it, a sine wave is get. For instance, for $x_0 = 8$ and $y_0 = 6$ the following plot (in a plane θ - r) is get. Only points such that r > 0 and $0 < \theta < 2\pi$ are considered.

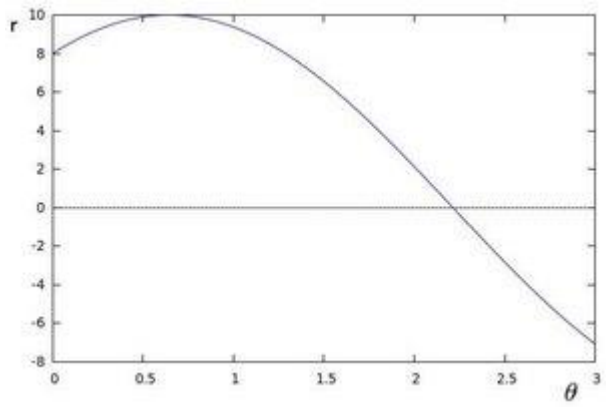

Figure 24. Sinusoid of lines goes through (x_0, y_0)

The same operation can be done above for all the points in an image. If the curves of two different points intersect in the plane θ - r, that means that both points belong to a same line. In general, by finding the number of intersections between curves it can define a line. The more curves intersecting means that the line have more points that represented by that intersection.

Here is the code for line detection by using Hough line transfer:

```
import cv2
import numpy as np

img = cv2.imread('camImage.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray,50,150,apertureSize = 3)
minLineLength = 30
maxLineGap = 10
lines = cv2.HoughLinesP(edges,1,np.pi/180,100,minLineLength,maxLineGap)
for x1,y1,x2,y2 in lines[0]:
    cv2.line(img,(x1,y1),(x2,y2),(0,255,0),2)

cv2.imwrite('houghlines5.jpg',img)
```

Figure 25. Code for line detect

"minLineLength" means the minimum length of the line. When detecting lines, line segments shorter than this are rejected.

4.6.2 Hough Circle Transform

The Hough circle transform works similar to Hough line transform discussed above. The Hough circle transform is the algorithm used to detect a circle in an image, and in the polar coordinate system any circle can be expressed as :

$$x = x0 + r\cos\theta \quad (4)$$

$$y = y0 + r\sin\theta \quad (5)$$

In the formula above, x and y are the coordinates of any point at the circle and the circle formula is shown above. x0 and y0 are the centre of the circle. “cv2.HoughCircles()” function is been used in this code.

Here is the code for circle detection:

```
import cv2
import numpy as np

img = cv2.imread('camImage.png',0)
img = cv2.medianBlur(img,5)
cimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)

circles = cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,1,20,
                           param1=50,param2=30,minRadius=0,maxRadius=
                           0)

circles = np.uint16(np.around(circles))
for i in circles[0,:]:
    # draw the outer circle
    cv2.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)
    # draw the center of the circle
    cv2.circle(cimg,(i[0],i[1]),2,(0,0,255),3)

cv2.imshow('detected circles',cimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 26. Hough circle transform

5 STRATEGY MODULE

In this module it introducing the game strategy, including achieve the robot to make a decision about the next move position based on a fixed priority. After comparing the simulation results, after the calculation it will make the best decision.

5.1 Game Strategy

The strategy of Tic Tac Toe game is easy, in order to win in the game, the row of three chess has to be made. On paper, the Tic Tac Toe board is drawn as a pair of horizontal lines and a pair of vertical lines, with either an X, O, or empty space in each of the nine spaces. In the program, the Tic Tac Toe board is represented as a list of strings. Each string will represent one of the nine spaces on the board. To make it easier to remember which index in the list is for which space, they will mirror the numbers on a keyboard's number keypad. Because it is not easy to describe the .position in a 3x3 board, so the board is numbered like a keyboard's number pad as shown in Figure 27. The board[7] would be the top-left space on the board. board[5] would be the centre. board[4] would be the left side space, and so on.

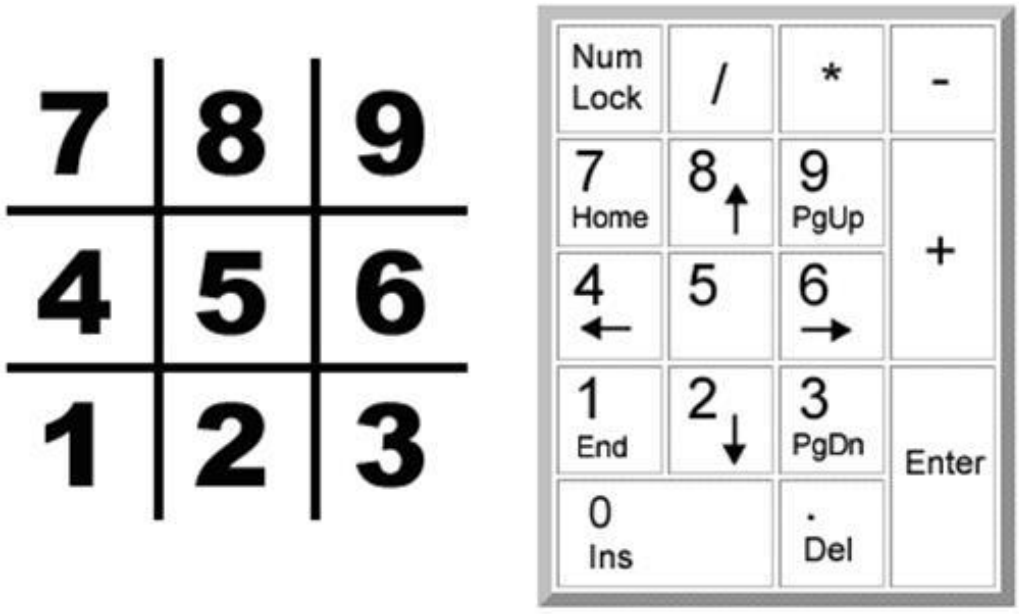

Figure 27. The board is numbered like a keyboard's number pad/16/

At the same time this game board was separated in another way into three different types of space, such as centre, corner and side. Figure 28 is a chart of each space.

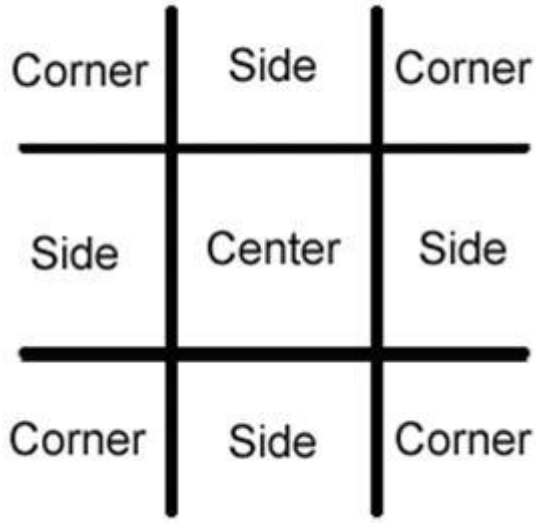

Figure 28. Location of centre, corner and side space/17/

After defining the location of game board, it is time to set the priority of this game. First of all a brief explanation about the possibility of success when putting the chess at each types of space.

When putting the chess at the centre, there are four ways to win this game, one is board[2] board[5] board[8] and the other is board[4] board[5] board[6], as well as board[1] board[5] board[9] and board[3] board[5] board[7] as shown in Figure 29.

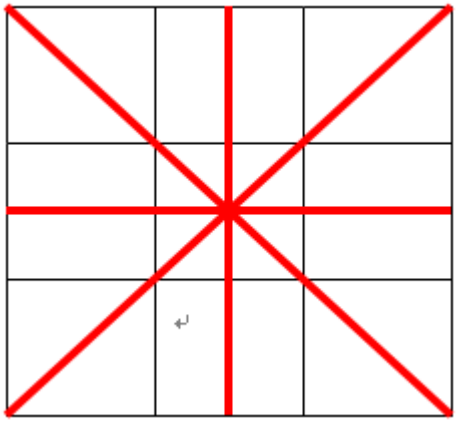

Figure 29. Winning situation when put chess at centre

If the chess is put at corner then there are three ways to win in the game. Even though there are four different corner places, but the result is the same, so board[1] is used as an example to explain it. If a chess is put at place of board[1] then the only way to win is to achieve broad[1] board[2] board[3] in a line and also board[1] board [5] board [9] in a line or board[1] board[4] board[7] in a line. The result are shown in Figure 30.

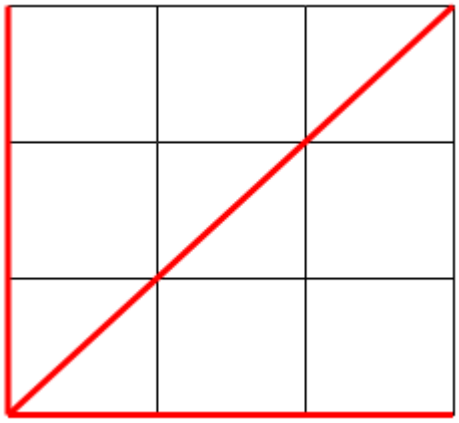

Figure 30. Winning situation when put chess at corner

If the chess is put at the side. Then there are only two situations that to win in the game. To choose board[2] as an example the winning situation is board[2] board[5] board[8] in a line or board[1] board[2] board [3] in a line just like as shown in Figure 31.

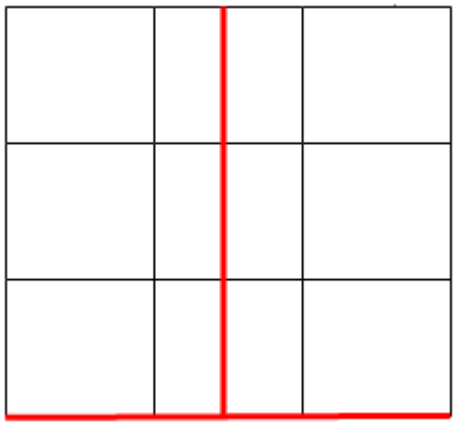

Figure 31. Winning situation when put chess at side

From what has been discussed above, it is easy to detect that the centre place is better than the corner place and the corner place is better than the side place. And as for four

different corner place, set the priority by ourselves is like board[9] > board[7] > board[3] > board[1]. As well as the four different side place, set it like board[8] > board[6] > board[4] > board[2].

5.2 Game AI

The AI needs to be able to look at the board and decide which types of spaces it will move on. The types of spaces have been discussed above.

The AI's algorithm for NAO's movement here is described in the flowchart shown in Figure 32.

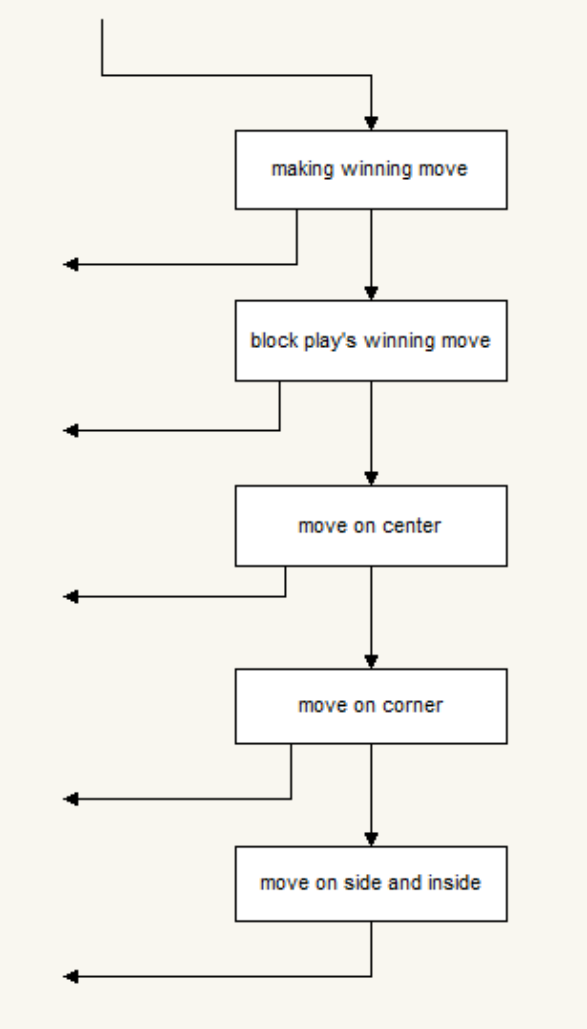

Figure 32. Five steps of the "Get NAO's move" algorithm

1. First, see if there's a move NAO robot can make so that it will win the game. If there is, then take that move. Otherwise, go to step 2.

2. See if there's a move the player can make so that the human player will win the game meanwhile the NAO robot player will lose in the game . If there is, move there to block the player. Otherwise, go to step 3.

3. Check if the centre place is free. If so, then move there. If it is not, then go to step 4.

4. Check if any of the corner spaces are free. If so, move there. If no corner space is free, then go to step 5.

5. Move on any of the side places. There are no more other steps, because if the execution reaches step 5 the side spaces are the only spaces empty.

A piece of code written in the Python language out of the algorithm for Tic Tac Toe AI is given below.

```
# Here is our algorithm for our Tic Tac Toe AI:
# First, check if we can win in the next move
for i in range(1, 10):
    copy = getBoardCopy(board)
    if isSpaceFree(copy, i):
        makeMove(copy, computerLetter, i)
        if isWinner(copy, computerLetter):
            return i

# Check if the player could win on their next move, and block them.
    for i in range(1, 10):
        copy = getBoardCopy(board)
        if isSpaceFree(copy, i):
            makeMove(copy, playerLetter, i)
            if isWinner(copy, playerLetter):
                return i

# Try to take one of the corners, if they are free.
    move = chooseRandomMoveFromList(board, [1, 3, 7, 9])
    if move != None:
        return move

# Try to take the center, if it is free.
    if isSpaceFree(board, 5):
        return 5

# Move on one of the sides.
    return chooseRandomMoveFromList(board, [2, 4, 6, 8])
```

Figure 33. Algorithm for Tic Tac Toe AI

5.3 Algorithm for bigger game board

Previous discussion is based on 3x3 game board which is simple and only for beginners. The robot can also play much difficult level. For example, playing on a bigger game board, which means it will have more space to place the chess and it will takes more time to end the game.

Take an example to expand the game board, such as play the game on a 9x9 board. The whole board can also be divided into three types of spaces, like what is shown in figure 34. Blue area is the centre area, like the centre space in 3x3 game board, in this kind of space there are four different ways to win in the game. Yellow area is the corner area, in this kind of space there are three approaches to win in the game. White area is the side location, in this kind of space, there are only two ways to win. Like explained before in 3x3 board, this 9x9 game board is numbered.

73	74	75	76	77	78	79	80	81
64	65	66	67	68	69	70	71	72
55	56	57	58	59	60	61	62	63
46	47	48	49	50	51	52	53	54
37	38	39	40	41	42	43	44	45
28	29	30	31	32	33	34	35	36
19	20	21	22	23	24	25	26	27
10	11	12	13	14	15	16	17	18
1	2	3	4	5	6	7	8	9

Figure 34. Location of centre area, corner and side space

There is the algorithm to bigger game board written in python:

```
# Here is our algorithm for our Tic Tac Toe AI:
# First, check if we can win in the next move
for i in range(1, 82):
    copy = getBoardCopy(board)
    if isSpaceFree(copy, i):
        makeMove(copy, computerLetter, i)
        if isWinner(copy, computerLetter):
            return i

# Check if the player could win on their next move, and block them.
    for i in range(1, 82):
        copy = getBoardCopy(board)
        if isSpaceFree(copy, i):
            makeMove(copy, playerLetter, i)
            if isWinner(copy, playerLetter):
                return i

# Try to take one of the centre area place, if they are free.
    move = chooseRandomMoveFromList(board, [11, 12, 13, 14, 15, 16, 17,
    20, 21, 22, 23, 24, 25, 26, 29, 30, 31, 32, 33, 34, 35, 38, 39, 40,
    41, 42, 43, 44, 47, 48, 49, 50, 51, 52, 53, 56, 57, 58, 59, 60, 61,
    62, 65, 66, 67, 68, 69, 70, 71])
    if move != None:
        return move

# Try to take one of the corners, if they are free.
    move = chooseRandomMoveFromList(board, [1, 9, 73, 81])
    if move != None:
        return move

# Move on one of the sides.
    return chooseRandomMoveFromList(board, [2, 3, 4, 5, 6, 7, 8, 10, 18,
     19, 27, 28, 36, 37, 45, 46, 54, 55, 63, 64, 72, 74, 75, 76, 77, 78,
     79, 80])
```

Figure 35. Algorithm for 9x9 game board

6 IMPLEMENTATION DETAILS

6.1 Environment Configuration

6.1.1 OpenCV Configuration

OpenCV (Open source computer vision library) is used for image processing. Open CV 2.7.3 was used in this thesis. OpenCV can be found and downloaded from their official website. When the installation package is run, all the files are extracted to the directory.

Except OpenCV, numpy-1.9.2 installation package and scipy-0.15.1 installation package were also needed. Both of them were plug-in which will be used in OpenCV.

OpenCV is configured by following steps:

- Open control panel and then find the system and security, then click the system, A window will open below and also for the whole path on the top of the window.
- Choose the advanced system setting then a window named system properties will appear, In the new window click environment variables.
- In environment variable settings the click the path, and the path of Python and OpenCV are saved into the variable value. Moreover the path is separated by using a semicolon.

The path is: “%PATH%;E:\doenloads\openCV\opencv\build\x86\vc11\bin

6.2 Timeline

Timeline is a tool which is located in Choregraphe software, It enables synchronization of the box with movements and also box with box and even support movements with movements. Timeline is always used to create animation for NAO. The key frames of NAO’s animation can be saved on time ruler and by connecting all the key frames together. NAO robot moves. Figure 36 shows the timeline panel and in Table 3 it shows the function of each part of the timeline panel.

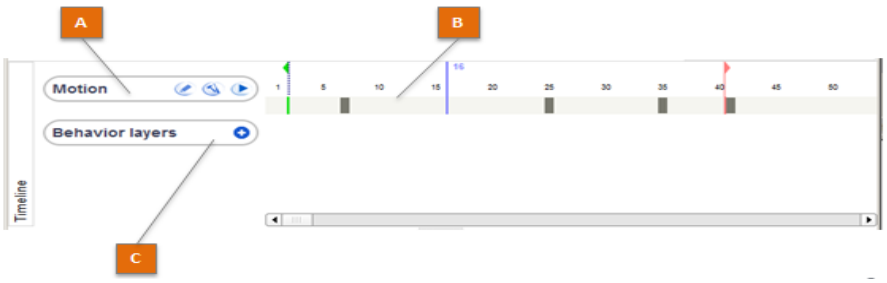

Figure 36. Timeline panel

Table 3. Description about each part of timeline tool /19/

Part.	Name.	Description.
A.	Motion.	It can be defined the motion key frames. Timeline editor bottom can be used to edit the motion of each joints in more details. Timeline properties can be used to define the value of frame per second. And in this case, it is set to be 10. Since in one second, the robot will move 10 frames, which will make the movements stable. Besides it can also be used to set the mode of resource acquisition. And normally this mode is set tube passive mode. Play motion will play the motion layer of timeline.
B.	Time Ruler.	The posture of each key frame can be created in the time ruler by click the one frame on the time ruler. And when creating the movement by using creating the several key frames instead of creating all frames of the movement, the robot will supplement the frames between each two key frames automatically.
C.	Behaviour Layers.	It can be used to define behaviour layers to be executed in parallel with motion key frame. Add button can be used to add one or more behaviour layers.

6.3 NAO Write Animation

For the NAO write animation, there are total of ten key frames. These frames can be separated into two animations, one of them is to prepare animation and the other one is to write animation.

6.3.1 Prepare Animation

For to prepare animation, there are total of five key frames. First, the robot is squatting and in the rest position as well as getting his motor on. Then the robot puts both of its arms down for the preparation. After this, robot raises its right arms to the start position. And then the NAO robot will open its right hand. After it has got the pen it will close its right hand to hold the pen. These five images from Figure 37 to Figure 41 show every single key frame of this prepare animation.

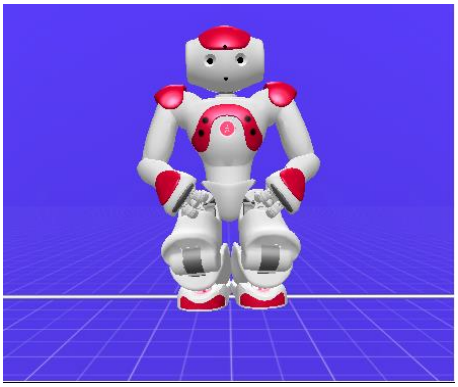

Figure 37. Rest position

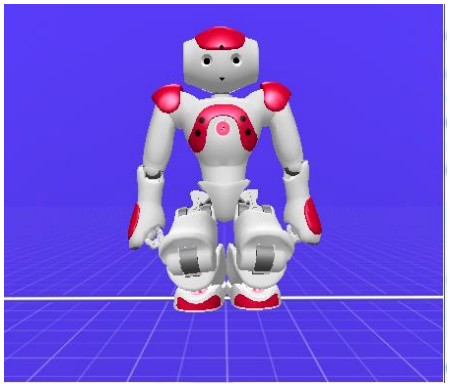

Figure 38. Put both arms dow

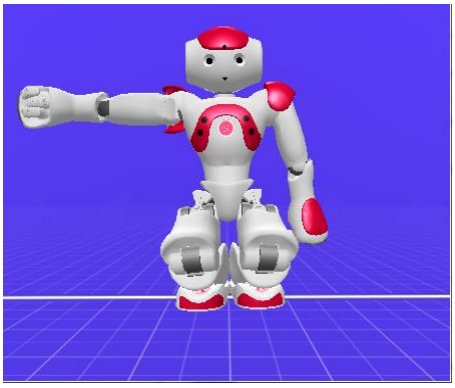

Figure 39. Raise Right Arm

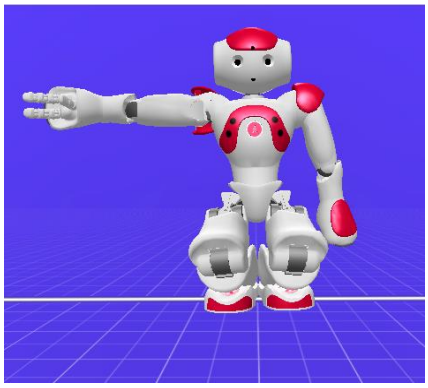

Figure 40. Open right hand

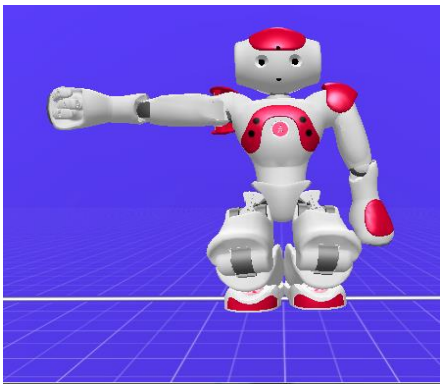

Figure 41. Close right hand

6.3.2 Write Animation

While writing, the robot will first move its right arms to the destination position, and then lower its right arms until the pen touch the paper. Moreover the robot will move its arms horizontally to make a small line as its chess. Then the robot will lift its right arms and move back to the start position waiting for the next movement. These five images from Figure 42 to Figure 46 show every single key frame of this animation.

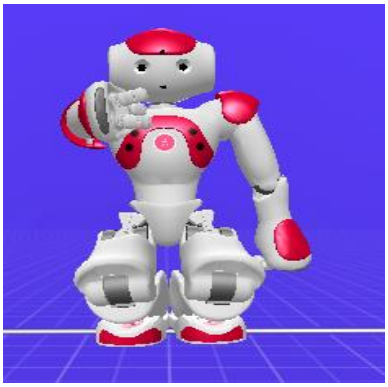

Figure 42. Destination position

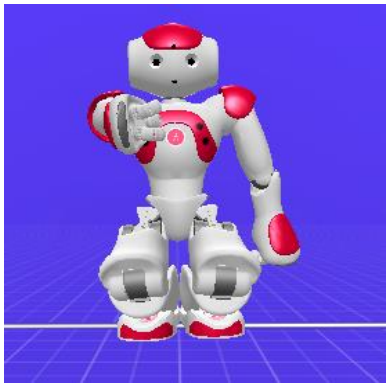

Figure 44. Lower right arm

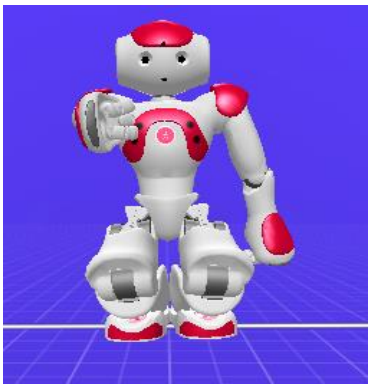

Figure 46. Move right arm horizontal

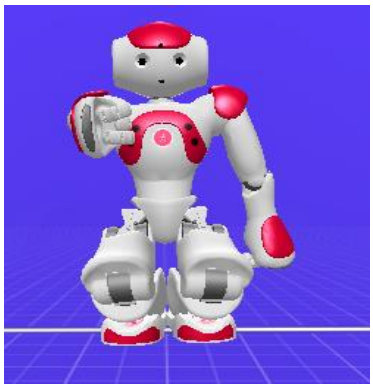

Figure 45. Lift right arm

Figure43. Move back to start position

7 REVIEW OF FUTURE RESEARCH

7.1 Improving NAO's Animation

In this project the animation of NAO robot was set by using Timeline. While it is easy to get started by using Timeline the problem is it cannot make NAO's move accurately. In this case NAO was only allowed to write a line in the project instead of drawing a cross. That is because of NAO's sensor on its hand is too weak to handle a pen tightly, so every time when NAO draws something it will change the position of the pen on its hand, therefore NAO cannot draw a line every time the same.

So in the future NAO's Animation should be improved so that NAO can hold the pen tightly and draw a more complicated symbol, such as a cross. There is a way for improving the animation called Forward And Backward Reaching Inverse Kinematics.

FABRIK(short for Forward And Backward Reaching Inverse Kinematics) is a heuristic methods. Unlike traditional methods, FABRIK does not make the use of calculations involving rotational angles or matrices. Instead, by finding the joint coordinates as being points on a line, the Inverse Kinematics problem is solved. these points are interactively adjusted one at a time, until the end effectors has reached the target position, or the error is sufficiently small. /20/

Here is the picture about NAO's animation part in the future:

Figure 47. NAO's animation in the future

A piece of code online about the FABRIK algorithm is given below.

Algorithm 1 The FABRIK algorithm.

Input: The joint positions $\boldsymbol{p}_i$ for $i = 1, ..., n$, the target position $\boldsymbol{t}$ and the distances between each joint $d_i = |\boldsymbol{p}_{i+1} - \boldsymbol{p}_i|$ for $i = 1, ..., n-1$.
Output: The new joint positions p_i for $i = 1, ..., n$.

```
% The distance between root and target
dist = |p_1 − t|
% Check whether the target is within reach
if dist >= d_1 + d_2 + ... + d_{n−1} then
    % The target is unreachable
    for i = 1, ..., n − 1 do
        % Find the distance r_i between the target t and the joint position p_i
        r_i = |t − p_i|
        λ_i = d_i / r_i
        % Find the new joint positions p_i
        p_{i+1} = (1 − λ_i) p_i + λ_i t
    end for
else
    % The target is reachable; thus, set b as the initial position of the joint p_1
    b = p_1
    % Check whether the distance between the end effector p_n and the target t is greater than a tolerance
    dif_A = |p_n − t|
    while dif_A > tol do
        % STAGE 1: FORWARD REACHING
        % Set the end effector p_n as target t
        p_n = t
        for i = n-1, ..., 1 do
            % Find the distance r_i between the new joint position p_{i+1} and the joint p_i
            r_i = |p_{i+1} − p_i|
            λ_i = d_i / r_i
            % Find the new joint positions p_i
            p_i = (1 − λ_i) p_{i+1} + λ_i p_i
        end for
        % STAGE 2: BACKWARD REACHING
        % Set the root p_1 at its initial position
        p_1 = b
        for i = 1, ..., n-1 do
            % Find the distance r_i between the new joint position p_i and the joint p_{i+1}
            λ_i = d_i / r_i
            % Find the new joint positions p_i
            p_{i+1} = (1 − λ_i) p_i + λ_i p_{i+1}
        end for
        dif_A = |p_n − t|
    end while
end if
```

Figure 48. The FABRIK Algorithm/20/

7.2 Improving TIC TAC TOE Algorithm

In this project the algorithm of Tic Tac Toe is not the best solution, so it be improved it in the future, so that NAO robot will never lose in the game. This can be achieved by using the MIN-MAX algorithm. In addition different game algorithm can be added in the project , such as connecting four or five in a row game, using a different game algorithm so that NAO can play a different game.

8 SUMMARY

This thesis introduces the behaviour design of the NAO humanoid robot: playing the Tic Tac Toe game.

For the vision system, the main algorithm is based on the Hough line transfer and Hough circle transfer to detect the game board and players movements. For the hardware part, NAO has two cameras one located in its forehead and the other located in its mouse, which provides up to 1280*960 resolution at 30 frames per second. By using the camera with image processing algorithm the vision system was completed.

The behaviour design was divided into two part, one was knowing how to play the game, and the other was drawing animation. As for knowing how to play the game, the game strategy needed to be known, This part is been done before in the project module studies. The drawing animation, it was made by using timeline, the key frames of the robot were set and saved it to the time ruler. By connecting all the key frames together NAO was made to move.

When the project started a lot of information and documentation about NAO robot are read and that inspired me a lot. The whole project was divided into several subproject. In this case this project was completed. Python language is also for the first time use in this project, fortunately Python is not difficult to get started, and there are also many source code online so that it could be learned by myself.

Finally this final project as well as the Bachelor’s thesis is finished, wish all the student working on this project the best for their future study and career.

REFERENCES

/1/Actives8 Robots official webpage. Accessed 1 September 2015.
http://www.active8robots.com/wp-content/uploads/nao-robot1.jpg

/2/History of NAO Robot. Accessed 1 September 2015.
https://www.aldebaran.com/en/humanoid-robot/nao-robot

/3/Hardware Platform of NAO Robot. Accessed 1 September 2015.
https://www.aldebaran.com/en/more-about

/4/NAO Evolution Datasheet pdf. Accessed 1 September 2015.
http://www.active8robots.com/wp-content/uploads/File-1400771269.pdf

/5/Dimension of NAO V5 png. Accessed 4 September 2015.
http://doc.aldebaran.com/2-1/_images/nao_v5_dimension.png

/6/Aldebaran Documentation What is Choregraphe. Accessed 2 September 2015
http://doc.aldebaran.com/2-1/software/choregraphe/choregraphe_overview.html

/7/Webots Overview. Accessed 2 September 2015.
http://www.cyberbotics.com/overview

/8/NAO software 1.14.5 documentation Monitor. Accessed 2 September 2015.
http://doc.aldebaran.com/1-14/software/monitor/index.html

/9/Software and OS. Accessed 2 September 2015.
https://www.aldebaran.com/en/robotics-solutions/robot-software/nao

/10/Choregraphe Panels. Accessed 2 September 2015.
http://doc.aldebaran.com/2-1/_images/chore_interface.png

/11/Panels in a Glance. Accessed 3 September 2015.
http://doc.aldebaran.com/2-1/software/choregraphe/interface.html

/12/NAOqi Framework. Accessed 23 September 2015.
http://doc.aldebaran.com/1-14/dev/naoqi/index.html

/13/Side View of Cameras Picture. Accessed 5 September 2015.
http://doc.aldebaran.com/2-1/_images/hardware_camera_lateral_4.0.png

/14/Top View of Cameras Picture. Accessed 5 September 2015.
http://doc.aldebaran.com/2-1/_images/hardware_camera_top_4.0.png

/15/NAO-Video Camera. Accessed 6 September 2015.
http://doc.aldebaran.com/2-1/family/robots/video_robot.html

/16/Numbered 3x3 board. Accessed 26 September 2015.
https://inventwithpython.com/chapter10_files/image003.jpg

/17/Location of Centre Corner and Side. Accessed 26 September 2015.
https://inventwithpython.com/chapter10_files/image004.jpg

/18/Wikipedia Hough transform. Accessed 28 September 2015.
https://en.wikipedia.org/wiki/Hough_transform

/19/NAO Software Documentation: Timeline. Accessed 30 September 2015.
http://doc.aldebaran.com/1-14/software/choregraphe/panels/timeline_panel.html

/20/Playing Tic Tac Toe with the NAO Humanoid Robot pdf. Author: Renzo Poddighe. Date: July 1, 2013. Accessed 30 September 2015.