VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Haoxiang Peng

DEVELOPMENTS OF VISION SYSTEM AND GAME STRATEGY WITH NAO ROBOT

Technology and Communication

2015

FOREWORD

This is my final undergraduate thesis in the Degree Programme in Information Technology, at Vaasan Ammattikorkeakoulu, Vaasa University of Applied Sciences.

I would like to sincerely thank my thesis supervisor, Dr. Yang Liu, for his instructions for my thesis as well as his patient guidance in my academic research. During the time when I working on my thesis, he always helps me by providing valuable ideas, inspired me and gave me confidence. While benefitting from his profound knowledge and effective advice, I have learned much about scientific research approaches.

Moreover, in the past three years, as an Embedded System lecturer, he not only has taught me a lot in this field but also provided me lots of chances to take part in practical projects as well. I have benefitted tremendously from doing these projects. I can update my knowledge with current technology and gain many experiences in carrying out independent research and analysis. Therefore, I have to admit that I could never reach this far without his kind help.

Furthermore, I am very thankful to other professors and staff in Vaasan Ammattikorkeakoulu, including Dr. Menani Smail, Dr. Ghodrat Moghadampour, Dr. Chao Gao, Mr. Jukka Matila, Mr. Santiago Chavez, Mr. Antti Virtanen, Mr. Jani Ahvonen and all other staff who kindly helped and trained me with their wisdom and knowledge during my undergraduate study. Thanks to their help and guidance, I have learned both theoretical knowledge and precious life experiences.

Finally, I want to express my great appreciation to my family for their support and encouragement during my study and life during all these years. Likewise, my fellows, including Lu Chunqiu, Li Xiaotian, Zhou Yang in Botnia RoboCup laboratory should also be appreciated for their encouragement to my thesis. Hope they can obtain great achievements in the future.

Peng Haoxiang

Vaasa, Finland

21/12/2014

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Haoxiang Peng
Title	Developments of Vision System and Game Strategy with Nao Robot
Year	2015
Language	English
Pages	80 + 2 Appendices
Name of Supervisor	Yang Liu

This thesis introduces the development of vision system and game strategy with Nao robot. In this thesis, the Candy Crush Saga game is discussed for testing the algorithm. Discussions of usage of the algorithm applied to other computer games are also included.

This thesis mainly contributes to explore the possibility to apply vision system of the Nao robot to PC game area. For a long time, the usage of vision system of the robot was mainly focused on real world object. And a lot of attention has been paid to the communication between human to robot or robot to robot, when a computer is used as a tool for the human user. This thesis discusses the possibility for a robot to use the computer as a tool themselves.

This thesis was mainly developed with the Python language under Windows platform. The demo applications were compiled both for the Windows platform in PC and NAOqi system in the Nao robot. The 5th generation Nao robot belongs to the Vaasa University of Applied Sciences and NAOqi is version 2.1. In this thesis, the implementation methods were typical software engineering approaches, which include planning, developing, debugging and testing.

It can be concluded that the Nao robot is very functional and have immense potential to be developed with more functions. Likewise, mathematical theory support is very important and necessary when designing an algorithm.

Keyword: Nao robot, vision system, game strategy

CONTENTS

LIST OF ABBREVIATIONS

RGB	Red, Green, Blue
PC	Personal Computer
LoL	League of Legends
CCS	Candy Crush Saga
SPL	Standard Platform League
SSL	Small Size League
LPC	Local Procedure Call
RPC	Remote Procedure Call
PIL	Python Imaging Library
HSV	Hue Saturation Value
AI	Artificial Intelligence
HD	High Definition
RGBA	Red, Green, Blue, Alpha

LIST OF FIGURES AND TABLES

LIST OF APPENDICES

1 INTRODUCTION

1.1 Purpose

This thesis introduces the development of vision system of the Nao robot applied in PC game area with game strategy in detail. The main thesis focuses on three parts. The first part is about the algorithm of target recognition and identification, and the second part introduces the game strategy. In the third part, further discussion of usage in other games will be included.

1.2 Overview Structure

The whole thesis is divided into eight chapters. The first chapter introduces the background information of this thesis, including basic information about the Nao robot, and the Candy Crush Saga game, and motivation of this thesis. The second chapter illustrates the overall structure of the thesis and gives a simply introduction to each main module. The third chapter introduces the communication part. The fourth chapter focuses on the introduction of the algorithm of vision module and practical results with details. The fifth chapter gives the introduction of algorithm of game strategy module. The practical results with details are also included. The sixth chapter focuses on the algorithm of control module and its practical results. The seventh chapter discusses the possibility of the algorithm applied to other games. In this chapter, the League of Legend game is mainly discussed as an example. The eighth chapter suggests some expectation of research direction in the future. Finally, the ninth chapter summarizes the whole thesis and the appendix lists the mentioned source code of the thesis.

The entire application contains three modules: vision module, strategy module and control module.

- Vision Module

The whole vision module contains two parts. The first part aims to acquire the image of what the robot currently is watching. Identical video cameras that locate on the fore-

head of the Nao robot provide the vision function. They provide the resolution up to 1280x960 at 30 frames per second. They can be used to identify objects in the visual field, such as a box or a ball, and the bottom camera can watch the Nao robot's dribbles, which are not used in this thesis. The second part is mainly used to process the image, including the progress of elimination and reduction of noises and unnecessary information. After this module, a list contains recognition result is outputted.

- Strategy Module

The strategy module aims to realize algorithm part for the Candy Crush Saga game. This module receives the recognition result of vision module and follows programmed strategy to make the best decision of the next move in the game to get the highest scores. The decision made by the strategy module is based on calculation and simulation result. This final decision information will be output as a list to control module.

- Control Module

This module first receives the output list from the strategy module. Then by simulating the mouse event, the right candies will be swapped with each other based on the final decision. It also counts the steps to control the whole thread, for the reason that there are only six moves given to finish level 1 in the game.

1.3 Introduction to Nao Robot

- History

The Nao robot is an autonomous, programmable humanoid robot developed by a French robotics company Aldebaran Robotics headquartered in Paris. It has a body with 25 degrees of freedom and key elements are electric motors and actuators. Its sensor network includes two cameras, four directional microphones, sonar rangefinder, two IR emitters and receivers, one inertial board, nine tactile sensors and eight pressure sensors. /1/

The development of the Nao robot was launched in 2004 and lots of versions of the Nao robot have been released since 2008. The Nao Academics Edition was developed for research and education purposes and mainly serves universities and laboratories. It was released to institutions in 2008 and was made publicly available by 2011. Currently, the Nao robot has been used in many academic institutions worldwide. /2/ /3/

On 15 August 2007, Nao replaced Sony's robot dog Aibo as the robot used in the RoboCup Standard Platform League (SPL), an international robot soccer competition. The Nao was used in RoboCup 2008 and 2009, and the NaoV3R was chosen as the platform for the SPL at RoboCup 2010. The Botnia robot team of the VAMK, University of Applied Sciences took part in the small size league (SSL) years ago and became the only RoboCup SSL team in the Nordic countries. Now the Botnia robot team pays much attention to the Nao robot and the aim is to regain the glory in the competition. /4/

Figure 1. Developers with Nao robot/6/

- Specifications of Nao V5 Evolution

Nao V5 Evolution was used to build this thesis and all the practical testing of application were taken with Nao V5 Evolution. The basic information of it can be found below in **Table 1**.

Table 1. Specifications of Nao V5 Evolution

Nao V5 Evolution (2014)[3][5]	
Height	58 centimeters (23 in)
Weight	4.3 kilograms (9.5 lb)
Power supply	Rechargeable lithium-ion battery providing 48.6 Wh
Autonomy	90 minutes (active use)
Degrees of freedom	25
CPU	Intel Atom @ 1.6 GHz
Built-in OS	NAOqi 2.0 (Linux-based)
Compatible OS	Windows, Mac OS, Linux
Programming languages	C++, Python, Java, MATLAB, Urbi, C, .Net
Sensors	Two HD cameras, four microphones, sonar rangefinder, two infrared emitters and receivers, inertial board, nine tactile sensors, eight pressure sensors
Connectivity	Ethernet, Wi-Fi

1.4 Background of Botnia Humanoid Robot Team

Before the Botnia Nao Robot Team organized, our Botnia SSL Robot Team, as the only qualified RoboCup SSL team in the Nordic area, has participated in the RoboCup world competition for several times and achieved high grades. In 2014, VAMK, University of Applied Sciences made the decision to buy two Nao robots V5 for educational purposes. Currently, Nao robot V5 is the most advanced version among other laboratories of Finnish universities. Later on, the Botnia Nao robot team was built and led by Dr. Yang Liu. Some applications of the Nao robot have already made in the laboratory.

1.5 Software in and out of Nao Robot

- Embedded Software

The embedded Software of the Nao robot is software that is running on the mother-board located in the head, allowing autonomous behavior to be executed. There are mainly two kinds of embedded software.

a. OpenNAO

OpenNAO is the embedded system for the Nao robot. It is a GNU/Linux distribution based on Gentoo and developed to fit various needs of the Nao robot. The system provides programs and libraries required by NAOqi.

b. NAOqi

NAOqi is the main software that runs on the Nao robot which is used to control its behavior. When the user creates new behavior for Nao, the relative modules and methods will be called and managed by NAOqi. The NAOqi Framework is used to program the Nao robot. It is able to answer to multiple robotics needs including parallelism, resources, synchronization and other events. With the help of NAOqi Framework, homogeneous communication between different modules (motion, audio, and video), homogeneous programming and homogeneous information sharing are allowed.

- Desktop Software

The desktop software is mainly used to create new behavior and control the Nao robot remotely. As the name suggests, they are running on the user's personal computer. There are two kind of desktop software that is used frequently.

a. Choregraphe

Choregraphe is mainly used to create animations and behavior, and users can test them on a simulated robot before trying them with the real robot. This software also provides functions of monitoring and controlling the robot.

b. Monitor

Monitor is used to give the user an elementary feedback from the Nao robot and to provide a simple access to camera settings.

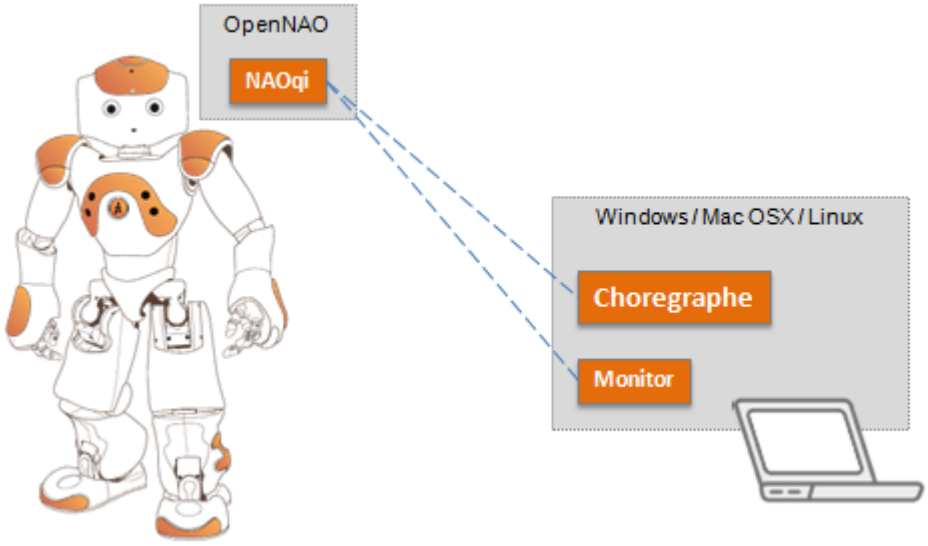


Figure 2. Desktop Software/7/

1.6 Introduction of Programming with Nao Robot

In this thesis, the latest version of Nao V5 robot is used. It is more capable and stable compared with older generation. Currently, the Nao robot offers a multi-platform and multi-level programming environment for users to develop new applications.

The easiest way to program the Nao robot is to use Choregraphe, which is mainly used by beginners. Choregraphe is a user-friendly behavior editor and also the name of a visual programming language. This software provides a drag-and-drop method that allows users to simply pick-up boxes of pre-programmed behavior and connects them to create complete sequences of behavior. They can also program their own behavior in the Python language, and then save it into the library. Besides, before sending a command to the real robot, users can modify the environment, insert and edit objects and simulate the behavior of the Nao robot by using the NAOSim 3D simulator. This kind of simulator is interfaced with Choregraphe to let users test the behavior of Nao robot in a virtual environment.

Moreover, experienced users can program the Nao robot with other languages like C++ and C# languages. In this way, applications can directly access to NAOqi (NAO

framework) APIs, which give the user low-level access to the robot's sensors and actuators.

Users can also program NAO with other software including Webots, MATLAB and Microsoft Robotics Studio. This thesis was built with the Python language on the Windows platform.

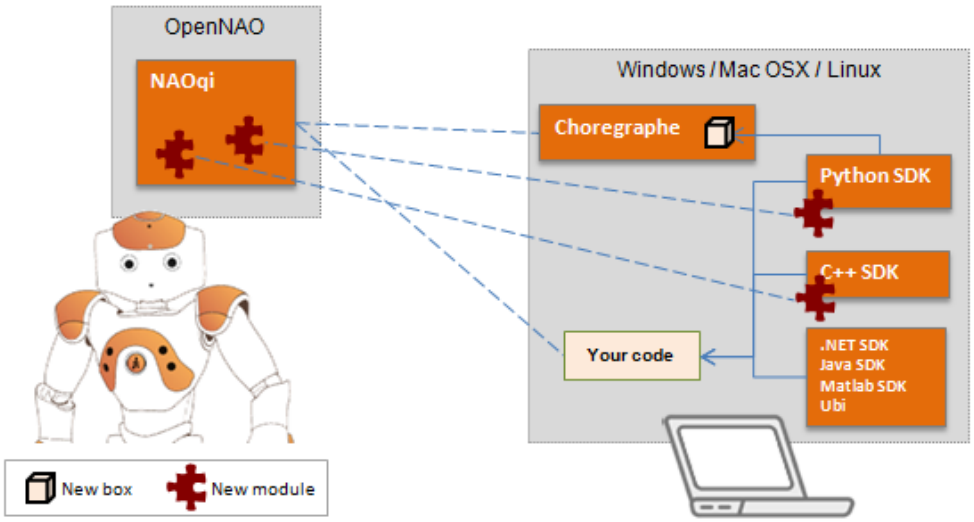


Figure 3. Programming with Nao robot/7/

1.7 Programming with Python in this Thesis

The Nao robot can be programmed by multiple languages and in this thesis the application was programmed with the Python language.

- Advantages and Disadvantages

Python is a general-purpose, high-level programming language and usually considered to be cleaner and more direct, with emphasis of code readability. Even though it may run slower than other languages, such as C++, considering the timing requirement of the Candy Crush Saga game is relatively low, Python is more convenient for programming. The Python API for the Nao robot is allowed to use all of the C++ API from a remote machine, and create Python modules that can run remotely on local computer or on the robot. /14/

- Input and Output

Unlike in C/C++, a list is usually used to carry the information as input or output of a function. The list is the most versatile data type available in Python which can be written as a list of values (items) separated by comma between square brackets. All items in a list are not required to be the same type.

```
sample_list = ['a','b',0,1,2]
```

1.8 Background of Candy Crush Saga Game

Candy Crush Saga is a ‘match-three’ type of computer game and currently also a mobile game. It was developed on April 12, 2012 for Facebook, and on November 14, 2012 for smart phones. In this game, each level has a game board filled with differently colored candies (sometimes might contain obstacles). These different colored candies include the red jelly bean, the orange lozenge, the yellow lemon drop, green chiclets, the blue lollipop head, and the purple cluster.

The basic move in this game is horizontally or vertically swapping the positions of two adjacent candies on game board, to create sets of three (or more) candies of the same color. Each level contains one or even more certain requirements that have to be completed in a given number of moves (or on a time limit); some levels require clearing "jelly" off the board by making matches on top of them, reaching a certain score, getting ingredient items to the bottom of the board, or having to clear certain amounts.

On level 1 of this game, a candy matrix formed with 6 different kinds of candies is shown with 8 columns and 5 rows. The requirement is to get as high scores as possible within 6 moves.

1.9 Motivation

There is no denying that robotic vision is of vital importance for robotic and embedded researches. Our Botnia Nao robot team has already made some applications about robot vision, such as face tracking. Besides, the long-term goal of the team is to compete

with other teams and even win the title in the RoboCup competition. So research on robotic vision about the Nao robot seems quite appealing and important.

Currently, there are plenty of robotic applications designed for robotic vision aiming at object recognition, tracking or navigation, however, there are only a few projects focusing on the recognition and identification of target shown on the screen. It is a promising area for robotic and embedded system to explore the possibility of teaching robot using the computer as a tool to complete certain task. Based on the considerations above, this thesis performs research on the recognition and identification of target shows on a PC screen and teaches the Nao robot to play a simple computer game.

The Candy Crush Saga game is a very famous game. This game relies on the recognition ability and calculation ability but does not have high requirements of the reacting speed of players at most of the levels. So it is a suitable game for the Nao robot to play. In this thesis, the level 1 of the Candy Crush Saga game will be introduced as a sample.

1.10 Further Developments

Besides the Candy Crush Saga game, other examples are referred in this thesis. The League of Legend game is also discussed as another example for the use of the algorithm.

2 COMMUNICATION MODULE

2.1 Distributed Tree and Communication

A real time application is a standalone executable or a tree of robot, or a tree of processes, or a tree of modules. However, the call methods are almost the same whatever the choice is. To connect an executable to another robot, the IP address of the robot and port are both needed. After the connection, all the API methods are available in the same way as with a local method. Embedded software NAOqi makes the choice between fast direct call (LPC) and remote call (RPC).

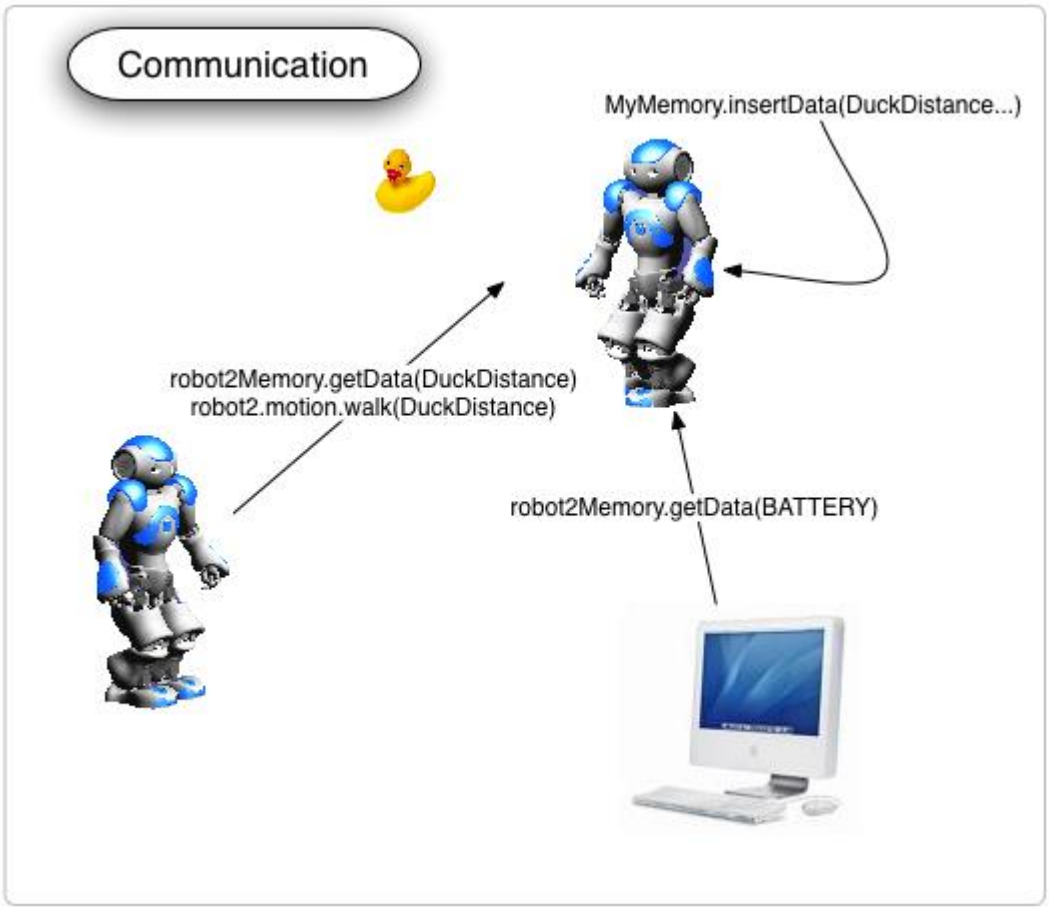


Figure 4. Communication with Nao robot/2/

2.2 The NAOqi Process

The embedded NAOqi executable which runs on the robot is a broker. When it starts to work, a preferences file called “autoload.ini” t will be opened, which defines the libraries it should be loaded. Each library contains one or more modules that use the broker to advertise their methods.

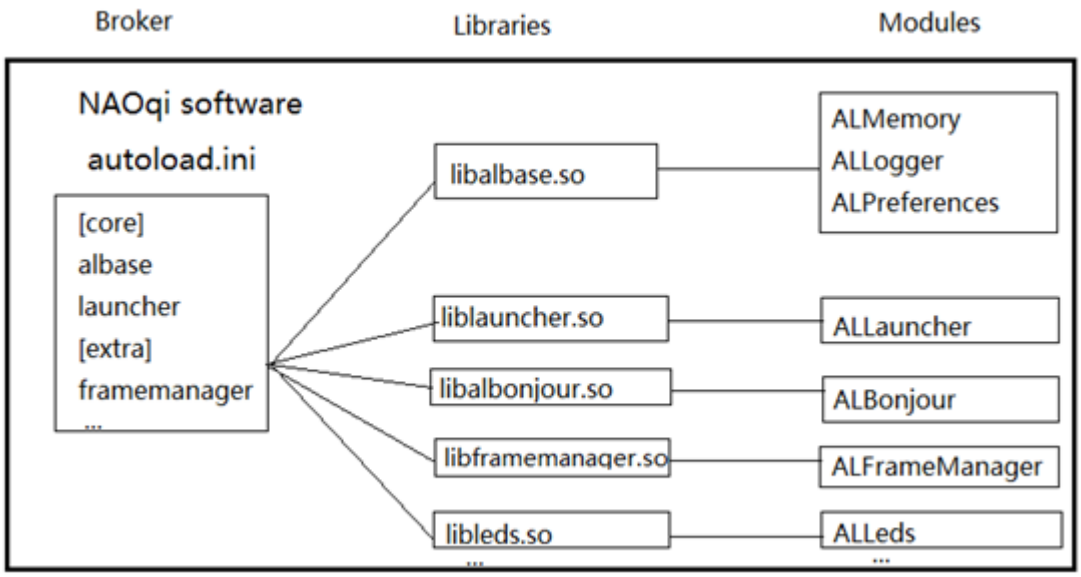


Figure 5. The NAOqi process

2.3 **Broker**

A broker is an object that provides two main roles:

a. It provides directory services, which allow the user to find modules and methods.

b. It provides network accesses, which allow the methods of attached modules to be called from outside the process.

Most of the time, brokers work in the back end, allowing users to program like calling to a “local module” (in the same process) or a “remote module” (in another process or on another machine).

The broker provides look-up services so that every module in the tree or across the network is able to find a counterpart method that has been advertised. Loading modules forms a tree of methods attached to modules, and modules attached to a broker.

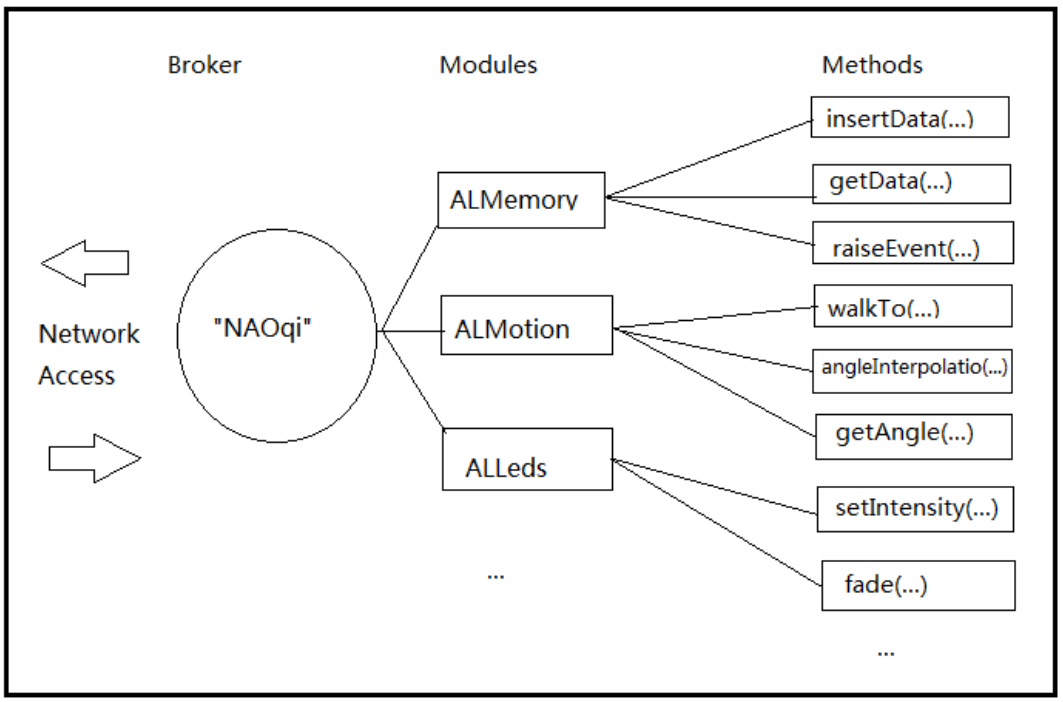


Figure 6. The Broker

3 VISION MODULE

The vision module is very important for the whole system. The algorithm for this module is based on color recognition. Although candies in the game have many different features, such as forms, color is still one of the most obvious features. The vision module contains two parts. The first part is used to acquire an image and raw data (RGB stream). This part is compiled and running on OpenNAO embedded system and stores the image result in the local work space. The second part is used to identify the candy, includes colors and positions. This part requires the progress of elimination and reduction of noises and unneeded information. The second part compiles in a local computer and output recognition result as a list for the next game strategy module.

3.1 Hardware Part for Vision System

3.1.1 Range of Vision Field

The Nao robot sees things using two 1280*960 cameras, which can capture up to 30 images per second. The first camera that is located on the forehead of the robot is used to scan the horizon; while the second camera is located at the mouth level scans the immediate surroundings. In this thesis, the first camera is used for vision services.

As mentioned above, the identical video camera of the Nao robot is responsible for collecting raw data from the vision field. For the reason that this camera is embedded inside the head of the Nao robot, the vision is quite different from a normal laptop camera. One of the biggest differences is uneven light distribution in the image seen by the robot, usually the upper part of the image appears darker than the bottom because of the position of the camera and light source.

Figure 7 and **Figure 8** generally illustrate the range of vision field of view. The vision field of Nao is relatively narrow, compared with human beings. Normally, the vertical range of the field of view with a human being is around 135 degrees and almost

180-degree forward-facing horizontally./15/ Therefore, the position of robot, which located in front of the monitor, need to be adjusted carefully according to the different size of monitor.

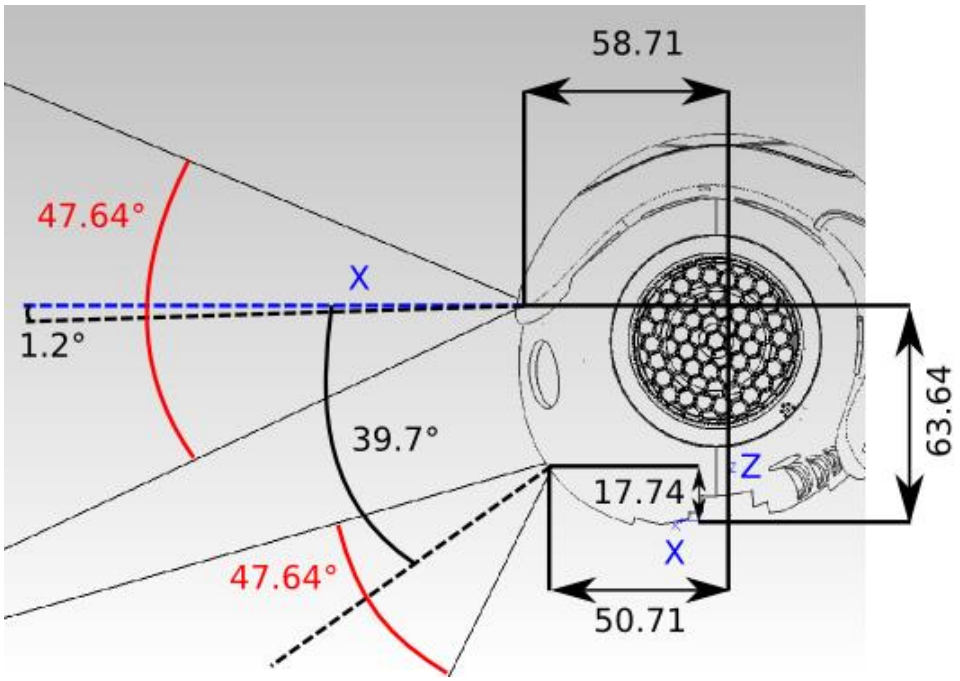


Figure 7. Vision range of the Nao robot/2/

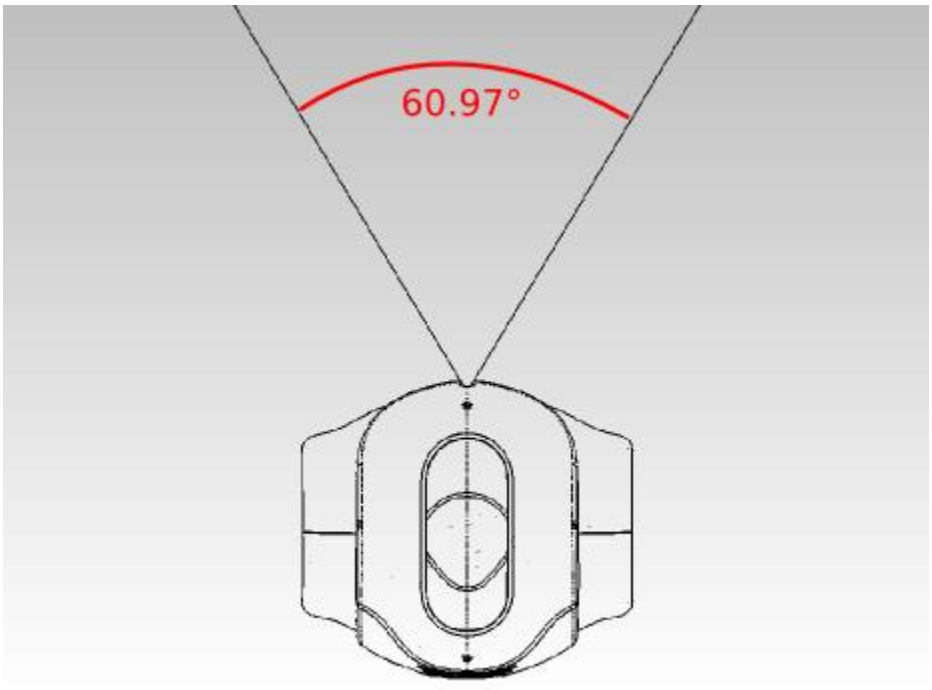


Figure 8. Vision range of Nao robot/2/

3.1.2 Data Sheet of Camera

The data sheet of the camera from tested the Nao robot is listed below in Table 2, which illustrates the basic information of the robotic camera that used to in this thesis.

If the table does not fit here, move some text from under the table here to avoid this blank space.

Camera	Model	MT9M114
	Type	SOC Image Sensor
Imaging Array	Resolution	1.22 Mp
	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

Table 2. Data Sheet of Camera/2/

3.2 Image Acquisition

Although real-time images can be watched by using Choregraphe 2.1 (shows in **Figure 9**), however in order to process necessary vision information, an image containing enough target information is necessary to be acquired. Since during the Candy Crush Sage game, the candy matrix will change itself until one candy is clicked and swapped with the other. A photo acquired from the robot's camera is taken when the candy matrix is stable enough for further processing. Choregraphe 2.1 is used during general adjustment progress of the robot’s position before running the application. When the robot is planted in a suitable position, the completed candy matrix is showed in the center of the screen. Otherwise, the position of the robot needs to be changed according to the real-time images provided by Choregraphe 2.1.

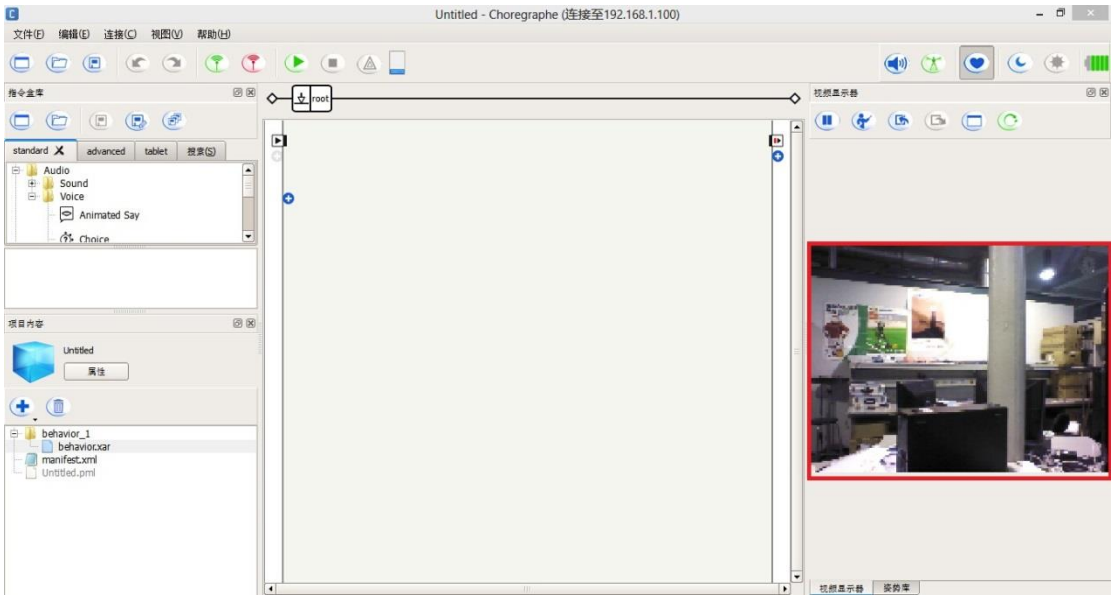


Figure 9. Video monitor in Choregraphe

3.2.1 Position

To make sure the image contains enough qualified information, the relative position of the Nao position and the position of monitor is important. The position should face the candy matrix (zoom to 100%) and have distance around 240mm. The candy matrix should shows in the center of the monitor and only the content shown on the PC screen should be seen by the robot. The monitor used in this thesis for testing purpose is a 27-inch Samsung SyncMaster P2770HD.

Some specific information of the position in this thesis are listed below in **Table 3**, the practical position can be adjusted.

Table 3. Robot's position.

Distance between center of screen to the ground	46 cm
Distance between center of screen to the camera	25 cm

And the practical position in testing progress of this thesis is illustrated in **Figure 10.**

Figure 10. The suitable position for recognition

3.2.2 Proxy

A proxy is an object that will behave as a module it represents. To use it during programming with Python, necessary proxy needs to be imported like below:

```
from naoqi import ALProxy
```

In this thesis, camProxy is created for the image acquiring. By using the name of the module (ALVideoDevice), the IP address of the Nao robot and the port of a broker, OpenNAO system of the Nao robot can be connected and its camera can be used by the user. One thing that needs to be paid attention to is that the module must be in the corresponding broker. After this progress, an image can be acquired by ALVideoDevice module. A piece of demo code is shown below. The resolution of the image acquired in this way is 1280*960 and stored in the workspace. This image will be opened in further processing.

```
resolution = 3    # VGA
```

```
colorSpace = 11   # RGB

camProxy = ALProxy("ALVideoDevice", IP, PORT) #make con-
nection with robot and use camera

videoClient = camProxy.subscribe("python_client", resolu-
tion, colorSpace, 5)

naoImage = camProxy.getImageRemote(videoClient)

camProxy.unsubscribe(videoClient)

  # The image returned and save it as a PNG using ImageDraw

  # package.

  # Get the image size and pixel array.

 imageWidth = naoImage[0]

 imageHeight = naoImage[1]

 array = naoImage[6]

  # Create a PIL Image from our pixel array.

 im = Image.fromstring("RGB", (imageWidth, imageHeight),
array)

  # Save the image.

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

3.3 RGB Color Model

The Python Imaging Library (PIL) provides image processing and graphics capabilities to the Python interpreter. The mode defines the type and depth of every pixel of the image that need to be selected before processing. There are several standard modes listed below:

- 1 (1-bit pixels, black and white, stored with one pixel per byte)
- L (8-bit pixels, black and white)

- P (8-bit pixels, mapped to any other mode using a color palette)
- RGB (3x8-bit pixels, true color)
- RGBA (4x8-bit pixels, true color with transparency mask)
- CMYK (4x8-bit pixels, color separation)
- YCbCr (3x8-bit pixels, color video format)
- I (32-bit signed integer pixels)
- F (32-bit floating point pixels)

To make it more convenient for further mathematical calculation, the RGB color model is used to convert vision information into mathematical form (shown in **Figure 11**). This mode is a kind of color model in which three primary colors (red, green and blue) are mixed together in different ways to reproduce a large range of colors. The name of this model results from the initials of the three additive primary colors (red, green, blue). In regular RGB mode, there are 3 bytes RGB data per pixel, which ranges from 0-254. For example, orange can be RGB (255, 200, 69).

To process the image, this thesis used Python Imaging Library to handle raster images, which is considered as rectangles of pixel data. The mode of the image defines the type and depth of a pixel in the image. In this module, RGB color mode (3x8-bit pixels, true color) is used. A piece of demo code is shown below.

```
img = Image.open("example.png") #open the image
rgb_im = img.convert('RGB') #get RGB stream
rgb_im.size # get size of the image
for j in xrange(rgb_im.size[1]):
    i=0
```

```
    first_row.extend([rgb_im.getpixel((i,j)) for i in
range(rgb_im.size[0])]) #scan one pixel by pixel
```

In the RGB color model, the function is able to convert a raw image into a stream containing RGB information. An example is shown in **Figure 11**. In this thesis, the converted stream is stored in a list for the convenient of further processing. The length of the list is: 1280*960= 1228800 (shown in **Figure 12**), which is quite a large amount. Therefore, useful information has to be dug out through data mining according to certain rules. All the noises and unneeded information have to be eliminated or reduced as much as possible until the useful information can be processed correctly.

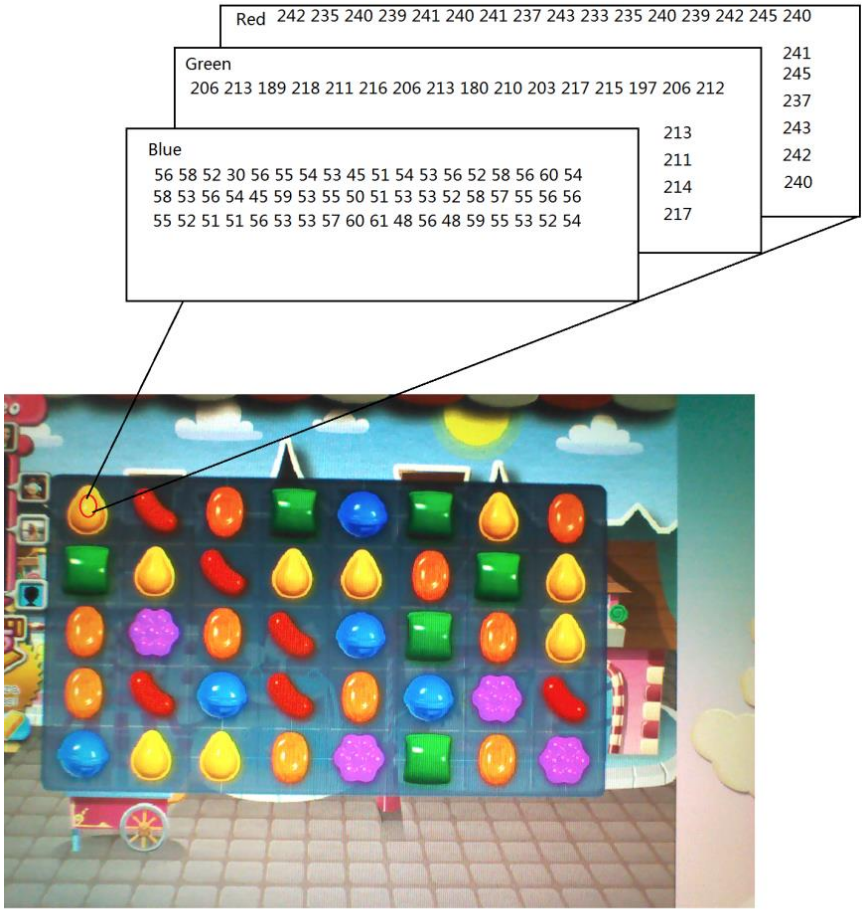


Figure 11. RGB stream in the image

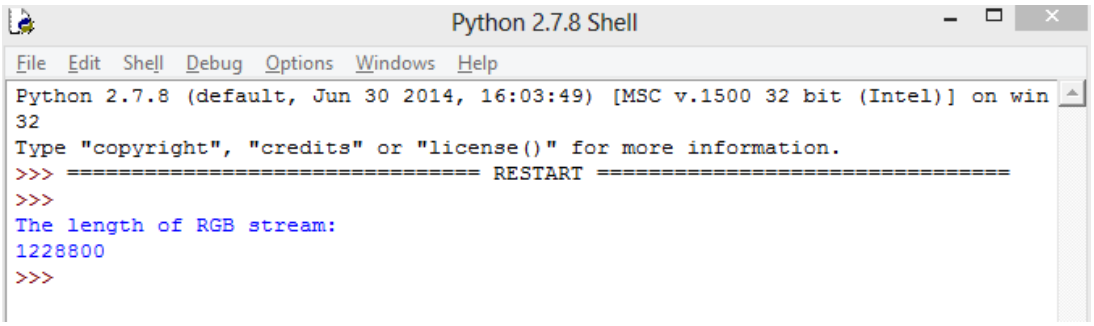


Figure 12. Length of RGB stream in the image

3.4 Comparison between different Color Spaces

There are many different kinds of color spaces that people may use for image processing. In this thesis, other three kinds of color space that often used are discussed. They are Lab color space, YUV color space and HSV color space.

The lab color space is a color opponent space based on nonlinearly compressed coordinates. The dimension L for lightness, while a and b stand for the color opponent dimensions. This model is based on digitized way to describe the human visual sensor, device-independent, so it makes up for the lack of RGB and CMYK mode must be dependent on the characteristics of the device color. The advantage of this color space provides wider range of secondary colors. /10/

The YUV color space is also very commonly used in image processing. In modern color television system, normally a three-tube color video camera or a color CCD (Coupled Device) camera is used. One advantage of using this color space is that color difference signals are encoded separately, using the same channel to send out with less bandwidth, compared with RGB color space. The other advantage of using YUV color space is that the luminance signal Y and chrominance signals U, V are separated. This feature can be used for fixing the light problem in an image, for example, illumination compensation is a usage of RGB to YUV color space. The relationship of RGB color space and YUV color space are: /8/ /11/

$Y=0.299R+0.587G+0.114B$	(1)

U = -0.147R- 0.289G +0.436B	(2)
V = 0.615R - 0.515G - 0.100B	(3)
R = Y + 1.14V	(4)
G = Y - 0.39U - 0.58V	(5)
B = Y + 2.03U	(6)

The picture (**Figure 13**) shown below is a practical example of illumination compensation progress. One piece of code wrote in Matlab can be found in Appendix 2. /12/ /13/

Figure 13. Illumination Compensation progress

HSV (hue-saturation-value) is one of the most common cylindrical coordinate representations of points in an RGB color model. Its angular dimension, starting at the primary color red at 0 °, passing through the primary color green at 120 ° and the primary color blue at 240 °, and then wrapping back to red at 360 °. The hue is referred to the proportion of the distance around the edge of the hexagon that passes through the projected point measured on the range [0, 1) or in degrees [0 °, 360 °). Saturation is used to help scale the chroma to fit into the range [0, 1] for every combination of hue and lightness or value, and can be acquired by dividing the chroma by the maximum chroma for that value or lightness. /9/ These two attributes can be illustrated

in mathematical form (R, G, B referred to the RGB data of three primary colors), for hue:

$$C = \max(R,G,B) - \min(R,G,B) \tag{7}$$

$$H' = \begin{cases} undefined, if C = 0 \\ \dfrac{G-B}{C} \bmod 6, if M = R \\ \dfrac{B-R}{C} + 2, if M = G \\ \dfrac{R-G}{C} + 4, if M = B \end{cases} \tag{8}$$

$$H = 60° * H' \text{ (if in degrees } [0°, 360°]) \tag{9}$$

For Saturation:

$$S_{HSV} = \begin{cases} 0, if \ \max(R,G,B) = 0 \\ \dfrac{C}{\max(R,G,B)}, otherwise \end{cases} \tag{10}$$

Currently, HSV is more often used in color pickers and some image editing software, and less commonly in image analysis and computer vision. Moreover, despite of the fact that both Lab color space and YUV color space have their own advantages, their influences are not significant when they are applied in the applications. The Lab color space provides far more secondary colors that may be required for the recognition of colored candies of a candy matrix. The color recognition of one color does not necessarily have to be so specific. The YUV color space truly puts positive effects on image acquisition; however the light affection can also be solved with a suitable RGB relationship. To avoid calculation progress which may slow down the recognition progress, the RGB color space is used in this thesis. However, the YUV color space is also suitable and can be considered to be applied in further researches.

3.5 Noises

The most important things players care about in the game is to get correct information of candy matrix. The entire strategy module is relied on the correct recognition result of the candy matrix. Therefore, both the color and position are needed to be recognized exactly correct. To reach this goal, information that is not related to the recognition process need to be reduced and eliminated as much as possible. They can mainly be divided into two parts: noises of the image and unneeded information.

a. Noise of the Image:

Digital images are prone to a variety of types of noise; however there are two main kinds of noises have negative effects on the image. They are random noise and banding noise.

- Random Noise

The noise may take place in the process of image acquisition randomly. This could happen because of multiple reasons, such as the feature of screen or the condition of light source; however, these affect the process very slightly.

- Banding (moiré patterns)

Photographs of a TV or PC screen taken with a digital camera can exhibit moiré patterns. Because both the screen and the digital camera use a scanning technique to produce or to capture pictures with horizontal scan lines, the conflicting sets of lines cause the moiré patterns. The digital camera can be aimed at an angle of 30 degrees to the screen to avoid the effect.

It is very obvious that several bands appear inside the candy, as it is shown in **Figure 14**. The pixel inside of the bands will show the darker color than its original color. It will affect the accuracy of color recognition.

Figure 14. Banding noise in the image

b. Unwanted Information:

This kind of information appearances are not by accident or by mistake, however, they still can affect the result by mixing unrelated information into useful information. So this kind of information also has to be reduced and eliminated as much as possible.

- Advertisements and Decorations of Web Page:

The useful information can be found in the candy matrix, which in the form of a small rectangle containing candies in the center. The candy matrix shows the red rectangle in **Figure 15**. Therefore, other information shown outside of the candy matrix is useless for the vision module but in some case may confuse the recognition progress. For example, an advertisement containing similar color like a candy may be considered to be a candy.

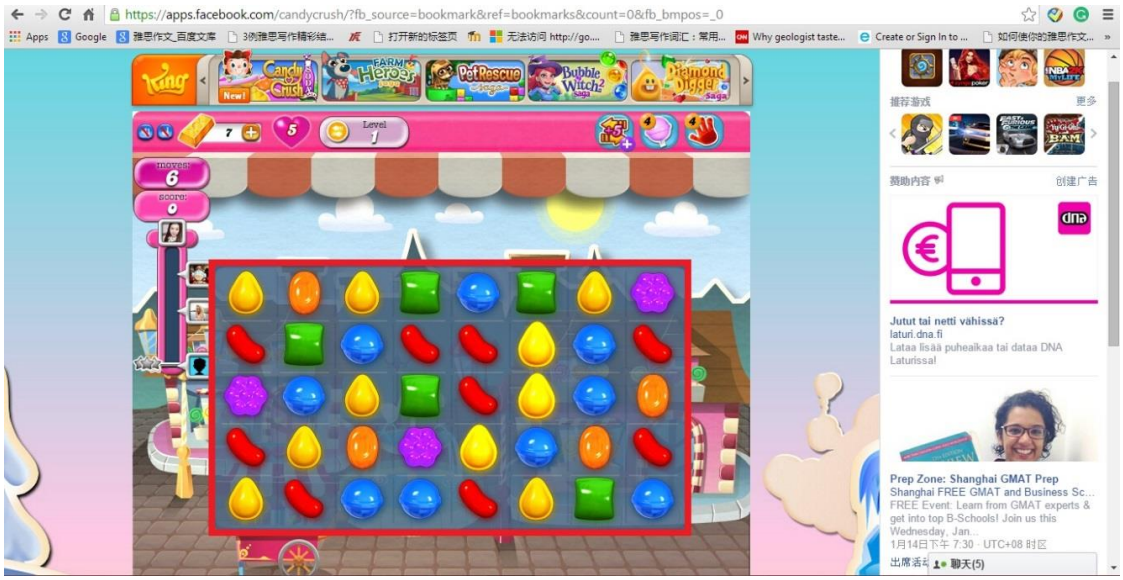


Figure 15. Candy matrix

- Unwanted Color inside of Candies

The candies are usually decorated with animations, such as flashing (shown inside the red circle), the color of flashing is near to white and may confuse recognition progress.

Figure 16. Green candy

Figure 17. Blue candy

- Easily misunderstand color because of the light condition

Some colors, such as yellow and orange, blue and transparent blue could be partly very similar to each other under certain light conditions.

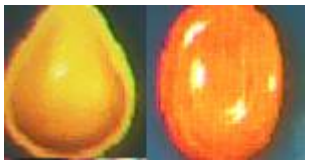


Figure 18. Easily misunderstood colors

3.6 Other Affections

Besides the noise appearance on the image, there are some other influences that may have negative effects on the recognition.

- Distance between the Nao Robot and Monitor

The distance between the Nao robot and the target (the monitor) will greatly affect the quality of the acquired image by affecting the proportion of the necessary information.

- The Brightness of Monitor

The brightness of the monitor will affect the result of color recognition, but normally have limited influences.

- Possible Shadow

Because of the position of the light source, a shadow may show on the monitor sometimes, which will affect the accuracy of the color recognition.

- Degree of Slant (either monitor or Nao robot or both)

As it is shown in **Figure 19**, unlike the game interface shown on the monitor, the image in the eyes of the Nao robot is not seriously parallel, due to the slant of the monitor and the Nao robot. It is not always possible to place the Nao robot in a proper place that the game interface shows in the center and vertical.

Figure 19. Slant of candy matrix

Figure 20. Scanning candy matrix

In order to collect enough information, one column of candies is needed to be crossed by enough horizontal lines, like scanning (shown in **Figure 20**).

The scanning process will check every pixel from the right side of the image to left. Therefore, because of the slant of the candy matrix in the image, surely scanning cannot go through the whole row or column and acquire completed information every time, but if most scanned rows contain all the information, then it will be enough for further processing.

3.7 Color Identification

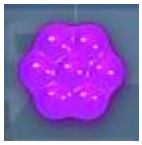
Despite the fact that there are many effects that may confuse color recognition, human beings are still able to identify different candies, therefore, the Nao robot should have the same ability with the help of a good algorithm.

3.7.1 Color Aliases

There are six kinds of colored candies on level 1 of the Candy Crush Saga game: yellow, red, green, blue, orange, purple, transparent blue. To make it more convenient for calculation and processing, the aliases for different colors are used to represent them respectively. A form contains information about aliases are showed below.

Table 4. Color Aliases

Color	**Aliases in letter**	**Aliases in number**
 yellow	y	1
 red	r	2
 green	g	3

 blue	b	4
 orange	o	5
 purple	p	6
 transparent blue	t	NA

For example, a row of candies in **Figure 21** can be represented as: “tytytgtrtptptptgtb”, or “11326634” (during simulation progress).

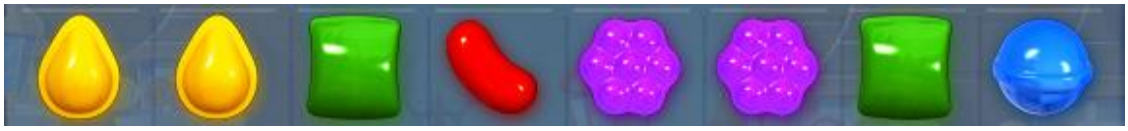


Figure 21. A row of candies

Moreover, a completed candy matrix of level 1 in **Figure 22** can be represented in a list like:

```
[[tgty-
tbtptptgtotpt][tptptbtbtgtbtptgt][totgtptbtgtytrtrt][tbtrtbtgtytytbtpt][tytbtbtotytgtbtbt]]
```

Figure 22. Candy matrix

3.7.2 Background Color

Transparent blue is the background color of the candy matrix and it plays a very important role in the module. The red lines identify the flags that are shown in the picture below. A row and a column of transparent blue pixels separate the adjacent two candies just like the red lines shown in the **Figure 23**. Therefore, the right recognition of transparent blue pixels can identify where the starter is and where the end of the whole recognition results for one candy is. Otherwise, a mistake will be made because information of adjacent candies may be processed and treated like the result of one candy. This kind of mistake will cause a fatal failure of the recognition because of uncompleted candy matrix.

Figure 23. Background color server as flag

For a single candy, the identification result is stored between two flags (’t’ list).

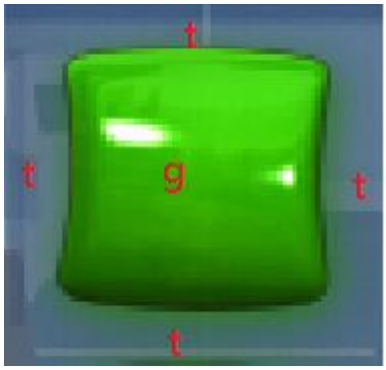


Figure 24. Background color server as flag

3.7.3 RGB Range

Although one candy shows with one color, in practice, the RGB information of every pixel in the candy is not same. Their differences are within a certain range. In this thesis, the RGB range of a color refers to the range of three primary colors (red, green, blue) of

every target color. To identify one color based on its RGB information, the RGB range firstly has to be measured.

Due to the influences of different light conditions, the practical RGB range of one color can be larger compared with the ideal range. Therefore, the practical RGB information for different colors has to be measured. There are several figures in different light condition and specific color in the figure can be measured with the tool to get the RGB information. For example, to collect information of red candy, the image to be measured should be taken in different light conditions. Then several points from red candy are chosen randomly and the RGB information is measured for the point, which is also the RGB information for one pixel. Finally the RGB information for those points is needed to be collected and analyzed. Through finding the range of the collected RGB information for three primary colors, the RGB range for red candy can be found.

Figure 25. Tool of measurement for RGB information

In this thesis, the practical RGB range of every color is found based on the measurement of the RGB information for each color. The measurements are captured under different light conditions and the images are acquired in the view of different directions. The final range of seven different colors is listed in the form below:

Table 5. RGB Range

Color	**Primary colors**	**Maximum**	**Minimum**
Orange	red	255	170
	green	228	103
	blue	92	36
Yellow	red	255	175
	green	255	178
	blue	103	53
Blue	red	58	20
	green	218	72
	blue	255	186
Red	red	249	178
	green	89	24
	blue	67	24
Purple	red	254	172
	green	232	81
	blue	255	190
Green	red	91	20
	green	251	81
	blue	144	33
Transparent blue	red	162	38
	green	197	48
	blue	220	44

3.7.4 RGB relationship

Through analyzing the RGB information for colors, the maximum and minimum value for primary colors (red, green, blue) can be found. However, ranges of some colors are

duplicated with the other in some certain range. So, to identify one color based on the RGB information, it is not enough to get only a RGB range. The RGB relationship has to be found as well.

The RGB relationship refers to the mathematical relationship between three primary colors data of one color, for the reason that three primary colors are formed to be a new color by a certain proportion. In practice, one or two general rules are enough to identify the color correctly with the help of the RGB range.

The RGB relationship of each color is found through observing the measured results. Normally the rules can be the average between two primary colors and the biggest number of one of primary color. Through analysis the recorded results, the average differences can be calculated, which is:

Dn refers to the difference of number n sample point's two primary RGB data.

N refers to the recorded quantity of sample points.

Equation of calculating average difference:

$$\text{Average difference}=\text{Sum}(D1+D2+\ldots+Dn)/N \quad (11)$$

Since the colors are significantly different with each other, the RGB range and RGB relationship cannot be duplicated at the same time.

In this thesis, the RGB relationship between primary colors is listed below in **Table 6** (red, green, blue means the respective RGB data):

Table 6. RGB relationship

Color	**Rule 1**	**Rule 2**
transparent blue	red<=green	abs(blue-green)<50
yellow	(red-green)<=45	(green-blue)>130
red	(red-green)>=150	(red-blue)>=150
green	(green>red)	(green-blue)>50
blue	(blue-red)>=135	N/A
orange	(red-green)>45	(green-blue)<=130
purple	(red-green)>80	(blue-green)>80

3.8 Noise Elimination and Reduction

3.8.1 General Noise Elimination and Reduction

The first step is general noise elimination. All the unnecessary colors in the whole image are eliminated after this step. In chapter 4.6, the proper RGB range and RGB relationship are acquired.

Based on the RGB range of color of each candy, the RGB information of every pixel will be compared with the range to find out if it is a useful color, otherwise this pixel will be set to black (0,0,0). Moreover, since the candy matrix is normally located in the center of the image, the scan range will set to 90 to 1190 in X direction while 125 to 800 in Y directions. The parts outside the scan area will set to black (0,0,0). A processed image and the original image are shown in **Figure 27** and **Figure 26**.

Figure 26. Practical image capture from Nao robot's camera

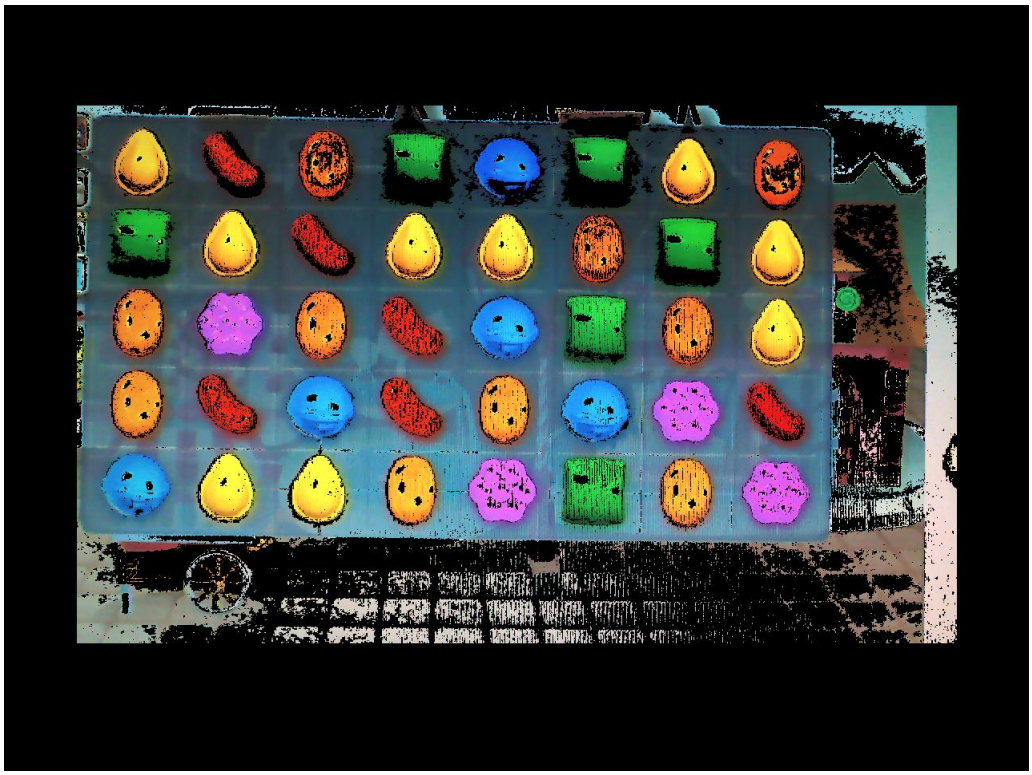


Figure 27. Image after first noise elimination and reduction

After this step, the list will contain 960 sub-lists. The RGB information of each item in the sub-list will be checked with the RGB relationship. If the information of one pixel fits all the rules, then a color will be considered to be recognized and the list marked with alias of this color will be added. The items of the sub-list after this step will be the aliases of recognized color.

However, the information contains both useful information and wrong recognition results. These lists are needed to be further processed and useful information need to be further dug out.

- Statistics and Calculation for information of one Candy

Because of the light influences and uneven distribution of color in one candy, the color recognition result, after comparing with RGB one pixel by one pixel, is not the same. For example, a recognition result for a yellow candy can be many ‘y’ lists but also contains some ’o’ lists among them.

However, it is obvious that most of the color recognition results of one candy are correct and that is one of the main reasons why human beings can identify the color of the candy even with many noises. So it is necessary to calculate the possibility for different color results of one candy. The function will choose the most possible recognition result that shows in one candy as the final recognition result for this candy.

An example of the recognition results of one row of candies (“tytrtotgtbtgtytrt”) after this step shows as below:

't', 't', 't', 't', 't', 't', 'tgtot', 'tgtgtot', 't', 'tgt', 'totgt', 't', 'tgt', 'tot', 'tgtgtot', 'tgtgt', 't', 't', 'tbtbtbtbtbtbtbt', 'tbtbtbtbt', 'tbtbtbtgt', 't', 't', 't', 't', 't', 'tbt', 'tbtbtbt', 't', 't', 't', 't', 'tot', 'tyt', 'tytrt', 'tytrtgt', 'tytrtgt', 'tytrtrtgt', 'tytrtrtgtbt', 'tytrtrtgtgt', 'tytrtrtgtgt', 'tytrtgtbtgtot', 'tytrtrtgtbtgtot', 'tgtytrtrtgtbtgtyt', 'tytrtrtgtbtgtyt', 'tytrtrtgtbtgtyt', 'tytrtrtgtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'totytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtgtytrt', 'tytrtrtgtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrtbtgtytrt', 'tytrtrtbtgtytrt', 'tytrtrtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtotgtbtgtytrt', 'totrtotgtbtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgty-

trt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotbtgtytrt', 'totrtotbtgtytrt', 'totrtotbtgtytrt', 'totrtotbtgtytrt', 'totrtotbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrtgt', 'totrtotbtgtytot', 'totrtotbtgtytot', 'totrtotbtgtytot', 'totrtotgtbtgtotot', 'totrtotbtgtotot', 'totrtotbtgtototgt', 'totrtotgtbtgtotot', 'totrtotgtbtgtotot', 'totrtrtgtbtotrt', 'totrtrtbtotot', 'totrtrtbtotot', 'totrtrtbtotot', 'totrtbtototgt', 'totrtbtotot', 'totrtbtotrt', 'totrtbtotrt', 'totrtbtotrt', 'totrtbtotrtgt', 'totrtbtotrt', 'totrtbtotrt', 'totrtbtotrt', 'totrtbtotrtgtgt', 'totrtotrt', 'trtotrt', 'trtotrt', 'tbtotrt', 't', 't', 't', 't', 't', 't', 't', 'totrt', 't', 'trt', 't', 't',

3.8.2 Second Stage of Noise Elimination and Reduction

- Reset possible ‘t’ List

The recognition results of transparent blue will contain several lists, however they do not purely contain only pixels of transparent blue. In most cases, they will also contain some pixels of other colors. These rows will shows like 'totrt', 'trt'. They usually have 4 features：

a. The length is shorter compared with other list of recognition result.

b. They show in a form of ‘t(color)t(color)t’ or ‘t(color)t’

c. The recognition results appear between two flags (‘t’ list) are part of the next row because of the slant of candy matrix in the image

d. They appears among other ‘t’ list

So these list will set to‘t’ list to avoid misunderstanding of flag, which may be mistreated as a row of candies.

- Check Length

Moreover, the length of the list will be checked. If the length of the list is 0, which means it contains no useful information, and then this list will be removed. In this progress, ‘t’ list will be used as the flag, all items between two ’t’ lists are treated as recognition results for one candy, as shown below.

't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 'tgtytrtrtgtbtgtyt', 'tytrtrtgtbtgty-
trt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrt-
gtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt',
'tytrtrtgtbtgtytrt', 'tytrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtot-
gtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'ty-
trtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt',
'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtotot', 'totrtotbtgtotot-
gt', 'totrtotgtbtgtotot', 'totrtotgtbtgtotot', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't', 't',

- Check duplicated ‘t’ List

After the identification progress, the amount of 960 lists will be generated which each of them contains information of a row of pixels. Black (0,0,0) will not be included in the list and consecutive of appearance of item ‘t’ list (transparent blue space) will be written with only one ‘t’ list in the list instead of several duplicated ‘t’ list.

't', 'tgtytrtrtgtbtgtyt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgty-
trt', 'tytrtrtgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtot-
gtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtrtgtbtgtytrt', 'tytrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'to-
trtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt',
'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'tytrtotgtbtgtytrt', 'totrtotgtbtgty-
trt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtotgtbtgtytrt', 'totrtot-
gtbtgtotot', 'totrtotbtgtototgt', 'totrtotgtbtgtotot', 'totrtotgtbtgtotot', 't',

- Statistics and Calculation for a row of Candies

Due to the slant of the candy matrix in the image, the list does not always contain an entire row of candies. So it is important to eliminate these uncompleted lists.

A function will check the length of the result list. If the length is equal to 17, which means it contains 8 colors and 9 transparent blue spaces (‘t’ flag) or the length is only one, which means a row of transparent blue as the flag of next row of candies, then this list will be kept.

Moreover, all the sub-lists between two flags are checked and the possibility of the right recognition result of this row is calculated. The most possible sub-lists will be treated as the recognition results for this row.

For the same example above, 'tytrtrtgtbtgtytrt' appears for 7 times, 'tgtytrtrtgtbtgtyt' appears for 1 time, 'totrtotgtbtgtytrt' appears for 8 times, 'tytrtotgtbtgtytrt' appears for 12 times, 'totrtotgtbtgtotot' appears for 3 times, 'totrtotbtgtototgt' appears for 1 time.

Therefore, in order to calculate the possibility of the recognition result, it can be assumed that 'tytrtrtgtbtgtytrt' is referred to event A, 'tgtytrtrtgtbtgtyt' is referred to event B, 'totrtotgtbtgtytrt' is referred to event C, 'tytrtotgtbtgtytrt' is referred to event D, 'totrtotgtbtgtotot' is referred to event E, 'totrtotbtgtototgt' is referred to event F.

The possibility of the recognition result is:

$$P(A) = \frac{7}{32} \tag{12}$$

$$P(B) = \frac{1}{32} \tag{13}$$

$$P(C) = \frac{8}{32} \tag{14}$$

$$P(D) = \frac{12}{32} \tag{15}$$

$$P(E) = \frac{3}{32} \tag{16}$$

$$P(F) = \frac{1}{32} \tag{17}$$

Based on the calculation result above, the most possible result is event D, which is 'totrtotgtbtgtotot'. The practical result is shown as below:

```
['y', 'r', 'o', 'g', 'b', 'g', 'y', 'r']
```

3.8.3 Flow Chart

Based on the discussions above, the flow chart below illustrates the progress of recognition for one row of candies.

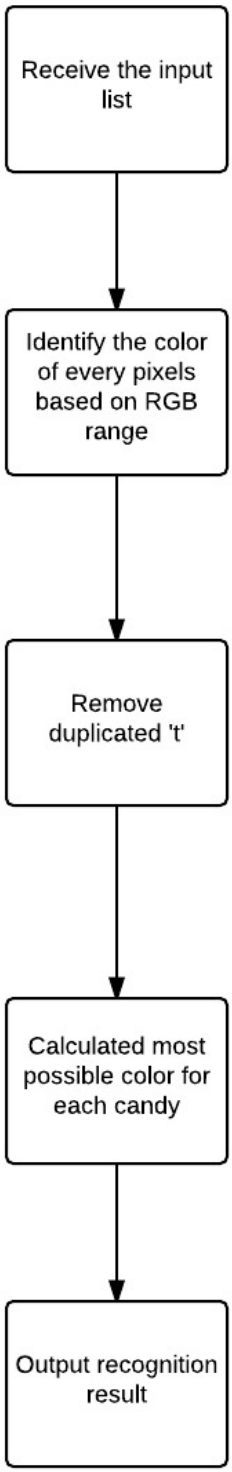


Figure 28. Flow chart for recognition a row of candies

4 STRATEGY MODULE

In this module, the game strategy is introduced. This module will be used to help the robot make the right decision of the next move based on the simulation result of all possible moves. After comparing the simulation results, the best decision will be made after the calculation.

4.1 Simulation Part

Because in the real game, whenever candies fit the match-three (or even more) situation, these candies will be crushed and leave an empty place. Then the candies above will fall down and fill the empty place until there are no more candies to be crushed. To figure out the best next move, this progress will be simulated in the application through the simulation part.

The simulation part (simulation function) is an important part of the whole strategy module. It receives the current candy matrix and crushes those candies that fit the match-three (or even more) situation.

For example, when the first candy of the first row and first column swap with the candy right next to it on the same row, then if any candies fit the match-three (or even more) situation, these same colored candies will all be crushed (set to 0). Candies on the top of crushed candies (number 0) will fall down. The match-three (or even more) situation will be checked again. This progress will continue until a candy matrix has no candies fit the match-three (or even more) situation and no candies left on the top of the empty space (number 0).

This progress can be illustrated in the flow chart shown in **Figure 29**.

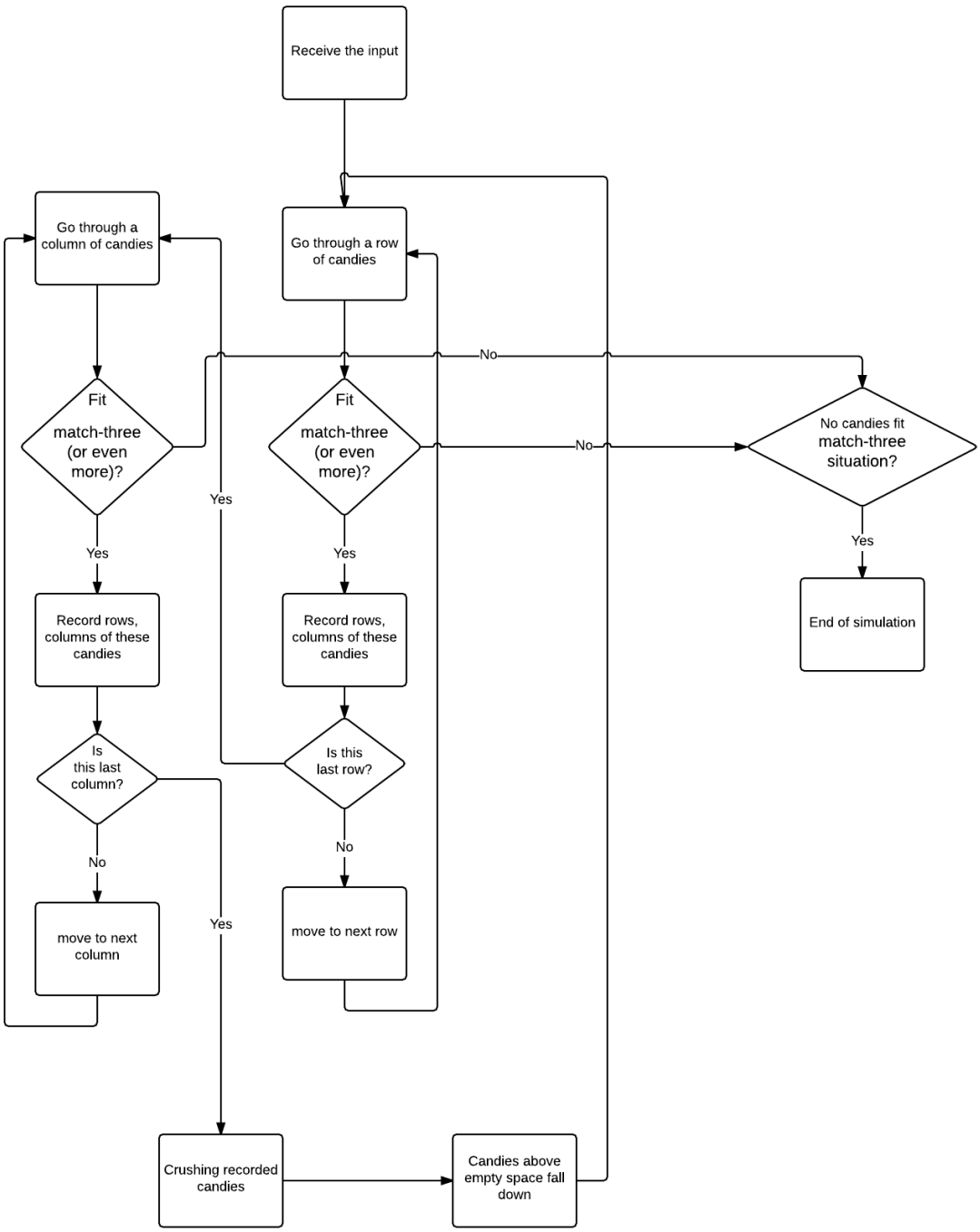


Figure 29. Flow chart of simulation progress

4.2 Decision-Making part

The Decision-Making part is also a very important part of this program. In this part, the main usage is to put information about a candy matrix into the simulation process and then the function will check how many candies are crushed and record the result. The

computer will try every possible move with the help of the simulation part and take advantage of the simulation result. Finally, according to the biggest amount of crushed candies, the move will be considered as the best decision.

To be more specific, after receiving the recognition result from the vision module, the simulation function will read this list and record the candy matrix in number alias. Then the amount of crushed candies, the row number, and the column number will be recorded. This is the simulation progress of one move.

Furthermore, every possible move for every two adjacent candies will be tried by the program. For a row of candies, every candy will be swapped with the right one next to it. And for a column of candies, every candy will be swapped with the one under it. If the amount of crushed candies (the amount of '0' items) of the current move is larger than the previous move, then the amount of crushed candies, the row number, the column number will be recorded and the previous move record will be deleted. Otherwise it will not be recorded even if the amount of crushed candies equals to the current best decision. Therefore, after the completion of the simulation progress, the record left will contain the information of the largest amount of crushed candies, the swapped mode, the row number and the column number of this candy.

This progress continues until all two adjacent candies are swapped with each other. Candies that made the largest amount of candies crushed are recorded. Finally , these two certain adjacent candies and this move (changed vertically or horizontal) will be chosen as the best move and simulated again to show the final simulation result , which is exactly the same with what will happen in the real game (before new candies in game fall and fill in the empty space created by crushed candies).

The swapped times will be: $C_5^1 * C_7^1 + C_8^1 * C_4^1 = 5 * 7 + 8 * 4 = 67$ (18)

If the changed candy matrixes of every move as lists are printed out, the results will be like shown in **Figure 30**.

```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
[[3, 2, 5, 3, 4, 3, 1, 2], [0, 1, 2, 1, 1, 5, 3, 1], [0, 6, 5, 2, 4, 3, 5, 1], [0, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 1, 5, 3, 4, 3, 1, 2], [3, 2, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 2, 3, 4, 3, 1, 2], [3, 1, 5, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 1, 4, 3, 1, 2], [3, 1, 2, 3, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 1, 3, 1, 2], [3, 1, 2, 1, 4, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 5, 1, 2], [3, 1, 2, 1, 1, 3, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 3, 2], [3, 1, 2, 1, 1, 5, 1, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [5, 1, 2, 1, 1, 5, 3, 1], [3, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 6, 2, 1, 1, 5, 3, 1], [5, 1, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 5, 1, 1, 5, 3, 1], [5, 6, 2, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 2, 1, 5, 3, 1], [5, 6, 5, 1, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 4, 5, 3, 1], [5, 6, 5, 2, 1, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 3, 3, 1], [5, 6, 5, 2, 4, 5, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 5, 1], [5, 6, 5, 2, 4, 3, 3, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 2, 5, 2, 4, 3, 5, 1], [5, 6, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 4, 2, 4, 3, 5, 1], [5, 2, 5, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 5, 3, 5, 1], [5, 2, 4, 2, 4, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 4, 5, 1], [5, 2, 4, 2, 5, 3, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 6, 1], [5, 2, 4, 2, 5, 4, 5, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [4, 2, 4, 2, 5, 4, 6, 5], [5, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 1, 4, 2, 5, 4, 6, 5], [4, 2, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 1, 2, 5, 4, 6, 5], [4, 1, 4, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 5, 5, 4, 6, 5], [4, 1, 1, 2, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 6, 4, 6, 5], [4, 1, 1, 5, 5, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 3, 6, 5], [4, 1, 1, 5, 6, 4, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 5, 5], [4, 1, 1, 5, 6, 3, 6, 6]]
[[2, 5, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 5, 2, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 3, 5, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 4, 3, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 3, 4, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 1, 3, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 2, 0], [3, 1, 2, 1, 1, 5, 3, 0], [5, 6, 5, 2, 4, 3, 5, 0], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [1, 3, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 2, 0, 0, 0, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 1, 0, 1, 5, 3, 1], [5, 6, 5, 0, 4, 3, 5, 1], [5, 2, 4, 0, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 5, 1, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 0, 1, 2], [3, 1, 2, 1, 1, 0, 5, 1], [5, 6, 5, 2, 4, 0, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 1, 3], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [6, 5, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 5, 6, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 2, 5, 4, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 4, 2, 3, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 3, 4, 5, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 5, 3, 1], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 1, 5], [5, 2, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [2, 5, 4, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 4, 2, 2, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 2, 4, 5, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 5, 2, 4, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 4, 5, 6, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 5, 1], [5, 2, 4, 2, 5, 6, 4, 5], [4, 1, 1, 5, 6, 3, 5, 6]]
[[5, 2, 5, 3, 4, 3, 1, 2], [3, 1, 2, 1, 1, 5, 3, 1], [5, 6, 5, 2, 4, 3, 0, 1], [5, 2, 4, 2, 5, 4, 0, 6], [4, 1, 1, 5, 6, 3, 0, 6]]
>>>
```

Figure 30. Changed candy matrix during decision-making process

This progress can also be illustrated with flow chart in **Figure 31**:

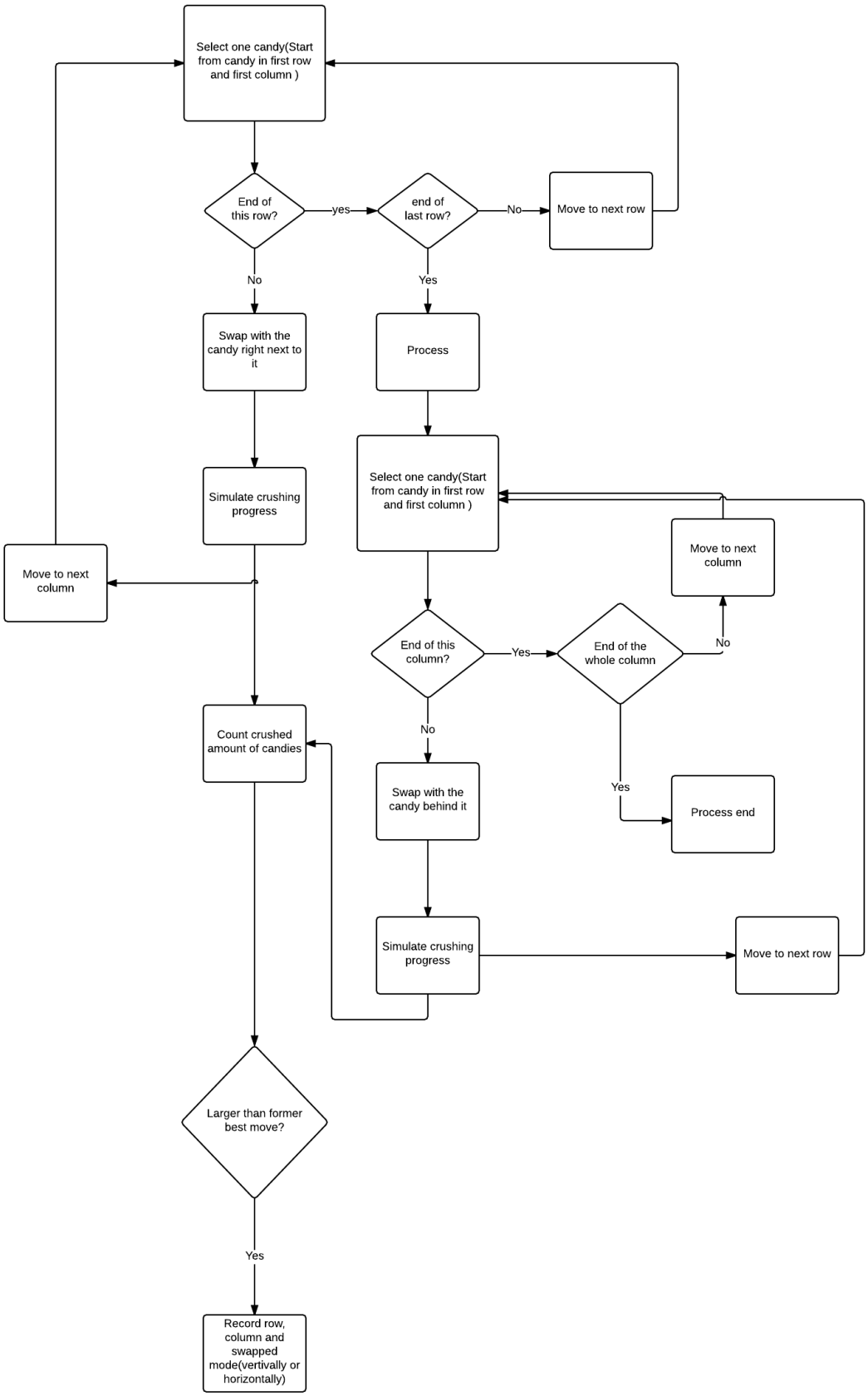


Figure 31. Flow chart of decision-making part

4.3 Output

The output will be a list containing the row number, column number and swapping mode (‘1’ means swap the candy with the one below it while ‘2’ means swap the candy with the one right next to it). For example, if the list [2,1,1] is generated, then the candy in row 3, column 2 will be swapped with the candy below it, which is the candy in row 4, column 2.

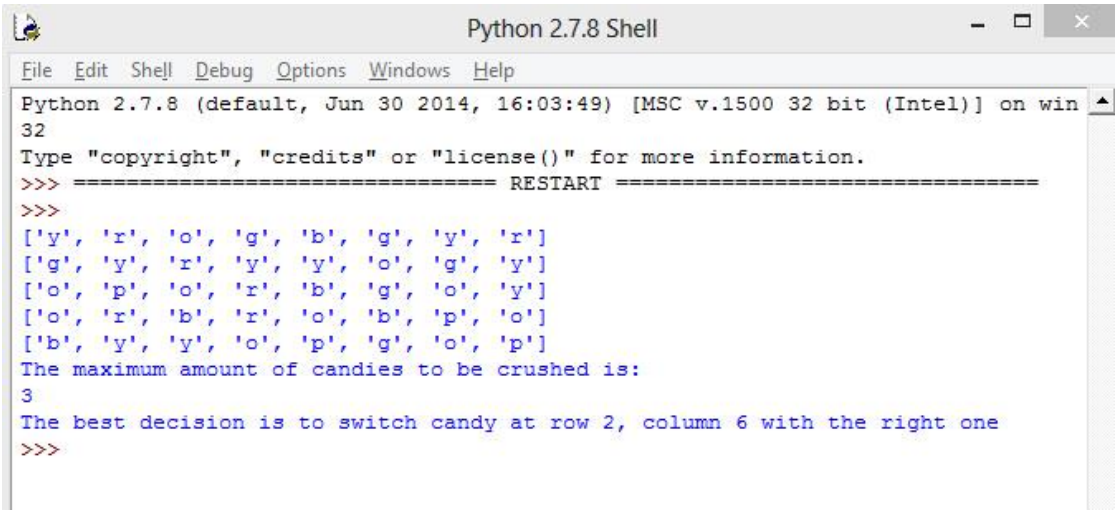


Figure 32. Recognition result

5 CONTROL MODULE

5.1 Overall Control

In the game level 1, the requirement is to crush as many candies as possible in six steps. So the whole progress will run and stop for six times to win the game. However, because of the animation after every crushing, a short pause is important. After tests, 4 seconds is enough and can satisfy the requirements. For the details of script example, please refer to the appendix.

5.2 Mouse Signal Control

- Input

In this module, the function mouse receives the list generated by the function decision. This list contains three variables: the line number, column number and swapping mode.

- Algorithm

The initial setting for the game interface is zoomed to 100%, and then absolute coordinates of every candy in the screen have to be measured. In this thesis, the absolute coordinate of the center of the first candy (first row and first column) is round (1903, 466). And the distance between the centers of two candies in the same row is about 80 pixels while the distance between the centers of two candies in the same column is about 70 (shown in **Figure 33**).

Figure 33. Measurement of distance between candies

Then the function will simulate the mouse events. The mouse arrow will click the candy that needs to be swapped, and then it will move to the other candy and click. After that, the mouse arrow will move to a certain coordinate near the edge of screen and stay away of candy matrix to avoid the mouse arrow affecting the recognition.

- Delay

There are mainly two kinds of delay in the application. One delay appears at the end of mouse arrow click the first candy. This delay aims to make it more clearly for users to observe the mouse events. It will last for 1 second. The other delay appears at the end of mouse events to wait for the completion of crushing animation. This delay will last for at least 3 seconds.

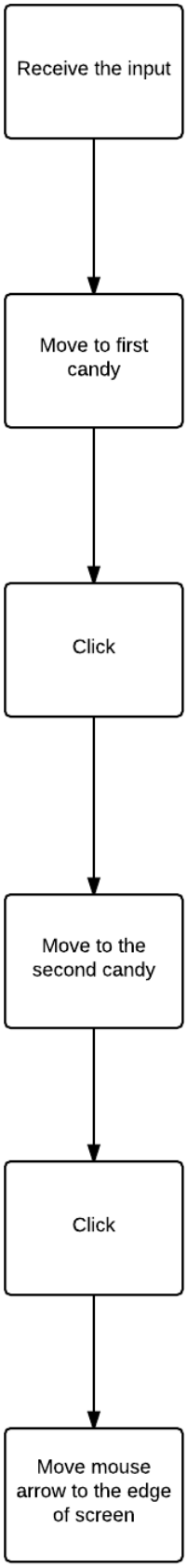


Figure 34. Flow chart of control module (mouse event)

- win32api

Since the hands of the Nao robot are too small and its three fingers are too short to control a normal sized mouse, a simulated mouse event is applied in the thesis. The mouse event simulation function can be used by importing win32api for Python. The following functions are used in this thesis:

a. SetCursorPos()

The function receives the coordinate then moves the mouse arrow to the spot. The co-ordinate should be the absolute coordinate on the screen, instead of the coordinate in the image.

b. mouse_event()

The function is used to simulate the click event. Three variables are needed as input for the function. They are settings of the left mouse button or the right mouse button and the absolute coordinate. For example, MOUSEEVENTF_LEFTDOWN means the left mouse button click down. To choose one candy, the event contains the left button down and up.

A piece of sample code for the implementation of the algorithm is written below:

```
def mouse(in_list):

    a=in_list[0]

    b=in_list[1]

    t=in_list[2]

    windll.user32.SetCursorPos(1903+80*b, 466+70*a) #Set
initial coordinate of the mouse arrow

    win32api.mouse_event(win32con.MOUSEEVENTF_LEFTDOWN,
1903+80*b, 466+70*a)  #mouse arrow move to target and left
mouse button down
```

```
    win32api.mouse_event(win32con.MOUSEEVENTF_LEFTUP,
1903+80*b, 466+70*a)  #mouse arrow move to target and left
mouse button up

    time.sleep(1) #delay for users see more clearly

    if(t==2): # if swap

        windll.user32.SetCursorPos(1903+80*(b+1), 466+70*a)
#mouse arrow move to target

        win32api.mouse_event(win32con.MOUSEEVENTF_LEFTDOWN,
1903+80*(b+1), 466+70*a)  #mouse arrow move to target and left
mouse button down

        win32api.mouse_event(win32con.MOUSEEVENTF_LEFTUP,
1903+80*(b+1), 466+70*a)  #mouse arrow move to target and left
mouse button up

        windll.user32.SetCursorPos(1451, 408)  #mouse arrow
move to target

    elif(t==1): # if swap vertical

        windll.user32.SetCursorPos(1903+80*b, 466+70*(a+1))

        win32api.mouse_event(win32con.MOUSEEVENTF_LEFTDOWN,
1903+80*b, 466+70*(a+1))  #mouse arrow move to target and
left mouse button down

        win32api.mouse_event(win32con.MOUSEEVENTF_LEFTUP,
1903+80*b, 466+70*(a+1))  #mouse arrow move to target and left
mouse button up

        windll.user32.SetCursorPos(1451, 408)  #mouse arrow
move to target
```

6 ALGORITHM APPLIED IN OTHER GAMES

In addition to the Candy Crush Saga game, the same algorithm can also be applied to other games. However, those games need to have the following features to ensure the accuracy of recognition:

- The targets can be identified by colors
- Background color should be different with the color of target
- The affections of noises is limited

6.1 Example of League of Legend Game

6.1.1 Simple Introduction of League of Legend Game

League of Legends (LoL) is a multiplayer online battle arena video game developed by Riot Games for Microsoft Windows and Mac OS X. It is a free-to-play game, supported by micro-transactions that was inspired by the mod Defense of the Ancients for the video game Warcraft III: The Frozen Throne.

In League of Legends, players assume the role of a character "Champion" with unique abilities, battling with a team against other players or computer-controlled champions. In the most popular game modes, each team's goal is to destroy the opposing team's nexus, a building which lies at the heart of a base protected by defensive structures. Each League of Legends game is discrete, with all champions starting in each game fairly weak and progressing by accumulating gold and experience over the course of the game.

Since this game is far more complicated, the discussion of the game in this thesis focuses more on the vision system instead of the game strategy. This application was tested under some simple occasion. In this thesis, an occasion is discussed to use the algorithm to identify the location of our Champion, opponent’s Champion.

6.1.2 RGB Range Measurement

Figure 35 shows a normal occasion when our champion fighting with the opponent's champion in the battlefield. Our Champion (Jax) is the character named "WEoKg" with a yellow health bar while the opponent's Champion is "Garen Bot" with a red health bar. Meanwhile, there are many decorations like trees, stones, fire, etc. Likewise, there are some Minions fights for each side with blue and red health bar. The necessary information for the Champions, such as spells, health, mana and a mini map are all shown around the edge of the screen.

The game is played in the colorblind mode to avoid unclear color recognition.

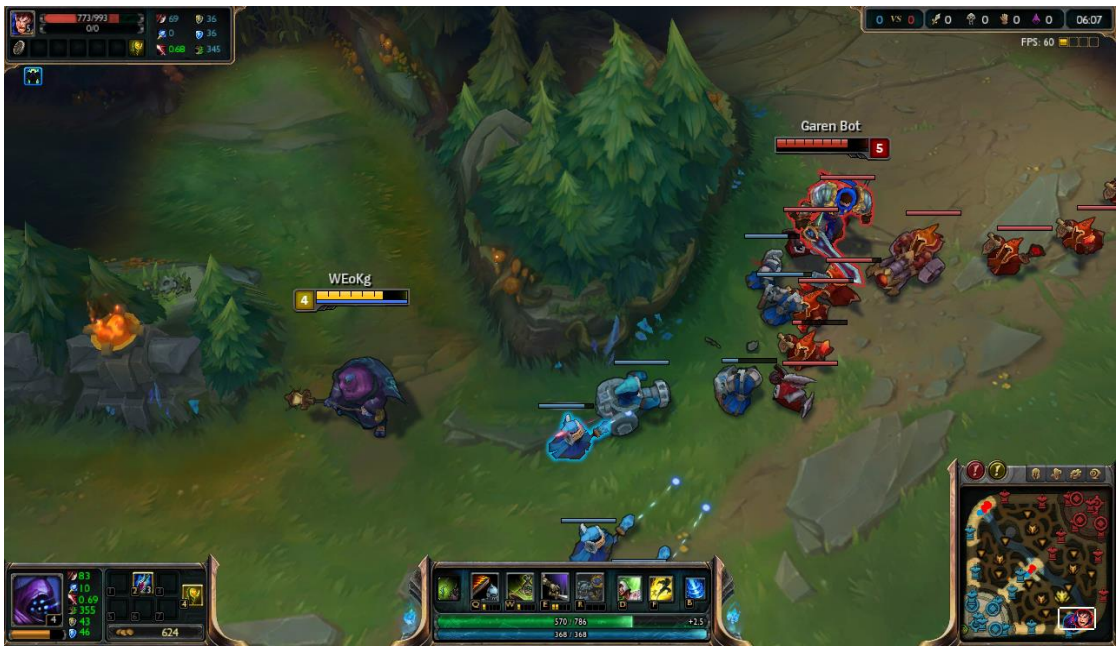


Figure 35. An occasion happened in League of Legends game

Therefore, based on the same algorithm discussed in the former chapter, the RGB information is measured first. The 10 points can be randomly chosen from the health bar of both sides. The RGB information is recorded below.

For our Champion (yellow health bar):

Number	Red(RGB data)	Green(RGB data)	Blue(RGB data)
1	174	137	28
2	167	132	27
3	172	136	27
4	255	255	33
5	170	134	27
6	205	163	43
7	255	204	54
8	156	124	26
9	211	164	26
10	226	186	70

Table 7. RGB data of our Champion

Therefore, the RGB data range can be concluded as below:

```
yellow_red_low=156
yellow_red_high=255
yellow_green_low=124
yellow_green_high=204
yellow_blue_low=26
yellow_blue_high=70
```

Then the opponent's Champion (red health bar) can also be measured. A total of 10 points are randomly chosen. The record are shown as below:

Number	Red(RGB data)	Green(RGB data)	Blue(RGB data)
1	172	45	25
2	112	28	14

3	114	28	15
4	113	28	15
5	168	44	24
6	170	45	25
7	113	28	15
8	177	59	41
9	165	43	24
10	188	86	70

Table 8. RGB data of opponent's Champion

Therefore, the RGB range for red health bar can be concluded below:

```
red_red_low=112
red_red_high=188
red_green_low=28
red_green_high=86
red_blue_low=14
red_blue_high=70
```

Since this game does not aim to identify different colors, the RGB relationship is not necessary to be acquired. Based on the RGB range of each side above, the image created after the elimination and reduction of noises is shown in **Figure 36**. It is obvious that most unnecessary information and noises are removed and the health bar still left in the image. Further processing can base on this image.

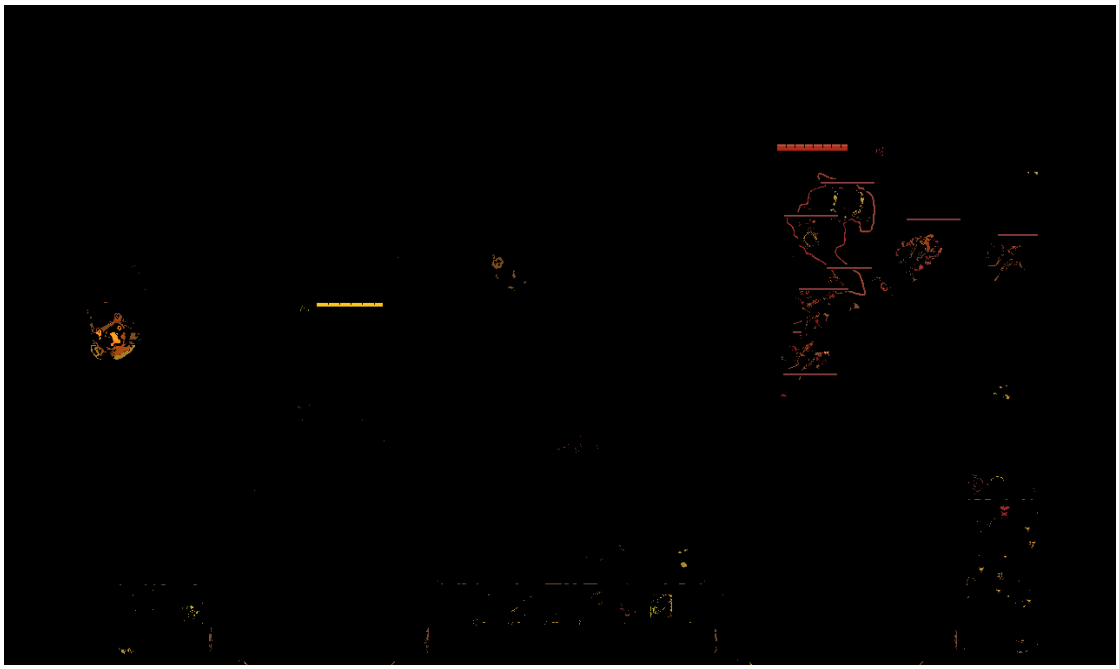


Figure 36. Image after first noise elimination and reduction

6.1.3 Role and Position Recognition

To play the game, the first thing is to know how to control our character. Therefore, it is important to know the location of the Champion forms each side. To take my Champion in the game as an example, the algorithm will be used to find the coordinate of the Champions.

As shown in **Figure 36**, the health bar appears as a small and length-changeable rectangle. It does not only show how many lives are left but it also identifies the position for the Champion. Although this colored bar has a fixed width and changeable length before the champion dies, the rectangle is always parallel with the bottom edge of screen, which makes it a reliable flag showing the position. If the coordinate of the first pot of this bar is recognized, then the coordinate of the character is recognized.

Based on the shape of the health bar, which is 5 pixels in width and 10 pixels in length, three rules can be used to eliminate and reduce the unnecessary information in the image:

- There should be 5 consecutive pixels in a column of the same color (yellow or red)
- There should be at least the amount of 10 consecutive pixels of the same color (yellow or red) in a row.
- The length should have 10 consecutive pixels in a row; every 10 consecutive pixels should have black pixels between them.

A piece of demo code is shown as below:

```
def first_renoise(in_list): # remove other colors
    row=[]
    j_old=0
    flag=0
    row=copy.deepcopy(in_list) #copy the list from input
    for j in xrange(715):
        for i in xrange(1280):
            flag=0
            for n in xrange (5):
                cx=row[j+n][i][0]  # RGB data(red)
                cy=row[j+n][i][1]  # RGB data(green)
                cz=row[j+n][i][2]  # RGB data(blue)
                #find out color or noise
                if((yellow_red_low<= cx <=yellow_red_high) &
(yellow_green_low<= cy <=yellow_green_high) & (yel-
low_blue_low<= cz <=yellow_blue_high) ):#| (red_red_low<= cx
<=red_red_high) & (red_green_low<= cy <=red_green_high) &
(red_blue_low<= cz <=red_blue_high)
                    flag=flag+1
```

```
            if(flag==4): # the width of health bar is 4 pixels
                print i,j
                flag=0
            else:
                row[j][i]=(0,0,0)
    return row
```

Then the required RGB stream can be acquired and the row number and the column number of the first item will be the coordinate of the Champions. To make the result show more clearly in the image, this thesis uses Python to draw two black lines across the health bar of Champion as markers.

Figure 37. Marker of health bar

The role and position for the Champions can be recognized. The yellow health bar is for our Champion while the red health bar is for the opponent's Champion. The position is the row and column number for the first pixel in whole list. The output is printed out in **Figure 38**.

```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
The coordinate of opponents champion is (151,438)
The coordinate of our champion is (324,360)
>>>
```

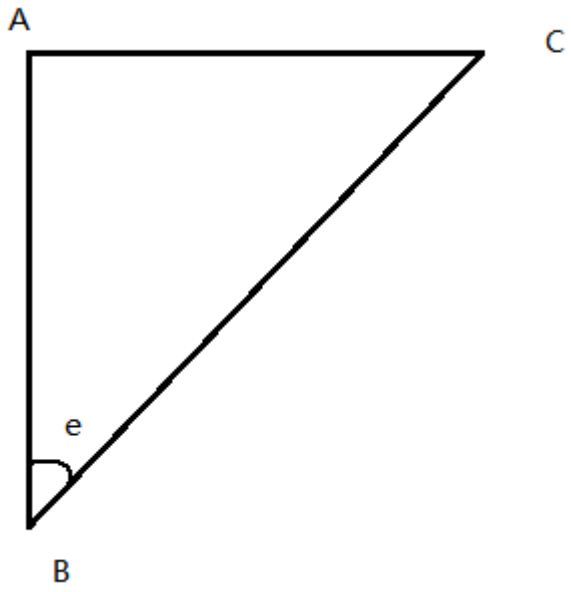
Figure 38. Role and position recognition result

Unlike the Candy Crush Saga game, the coordinate of the champion is flexible, to control the movement of our Champion, more information will be needed. In the next chapter, distance between Champions and elevation are calculated as an example.

6.2 Calculation of Elevation and Distance

The elevation and distance are normally important information to control the Champion. Although far more information can be acquired, this thesis only discusses the calculation of elevation and distance between the Champions as examples.

6.2.1 Calculation of Elevation



Suppose our Champion stands in spot B (324,260) while the opponent's champion stands in spot C (151,438). AB is perpendicular to AC

|AB|=324-151=173 (19)

|AC|=438-260=178 (20)

$$e = \arctan\left(\frac{|AC|}{|AB|}\right) \quad (21)$$

Therefore, based on function (19), (20), (21):

e=45.82 degree (22)

(The elevation is 45.82 degree.)

6.2.2 Calculation of Distance

Suppose our Champion stands in spot B (324,260) while the opponent's champion stands in spot C (151,438)

$$\text{Distance} = \sqrt{(X_B - X_C)^2 + (Y_B - Y_C)^2} \quad (23)$$

= 248.23

7 OVERVIEW OF FUTURE RESEARCH

7.1 Time Complexity of Vision Algorithm

The time complexity of the game strategy greatly depends on the game itself. The more complicated the game is, the strategy will also be more complicated. AI can always improve with a better algorithm.

The entire strategy module running on the computer, the time is affected obviously by the processing speed of the computer.

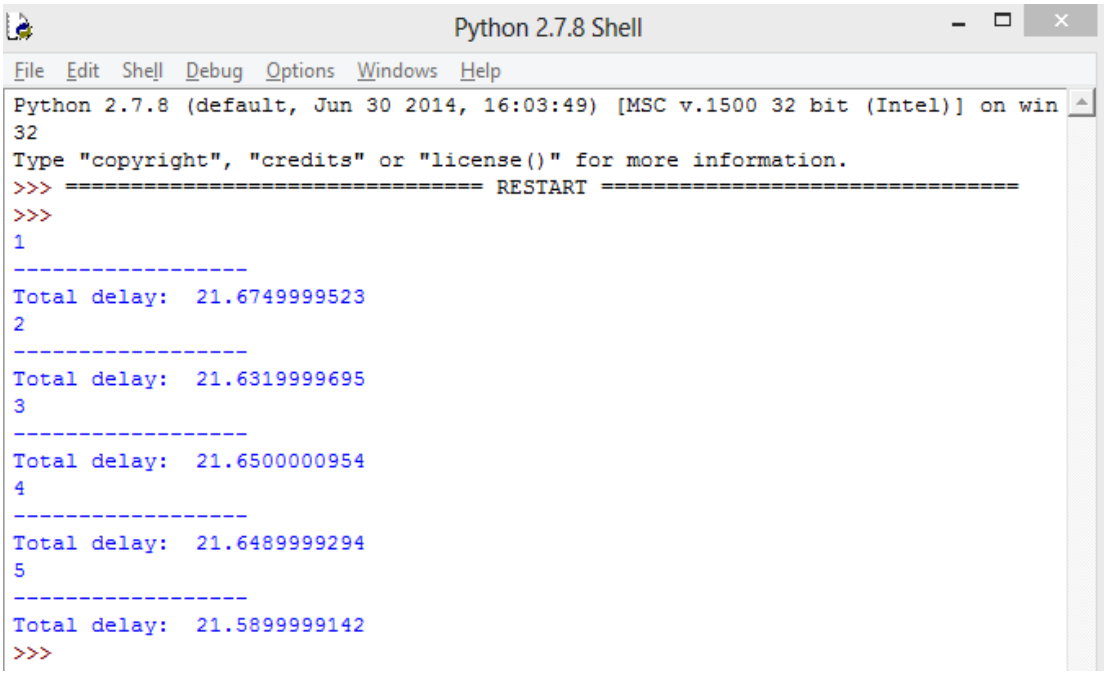


Figure 39. The delay for the whole progress

The vision module is partly compiled in the NAOqi system of the robot, which aims to get a proper image according to the settings of the user. The time cost depends on the robot itself and normally not changed. The other parts running on the computer are processing the picture and through analyzing the image makes the correct recognition. This part depends on the time complexity of the calculation and the processing speed of the local computer.

Leaving the hardware influences, the time complexity still can be reduced to adjust to more games that may have time of reaction requirements. A relatively long processing speed will be a limit for further research.

There are some general suggestions for further research:

- A better algorithm can be used to reduce the time complexity or space complexity even both.
- A more suitable color space can be applied to the vision module.
- The modules containing complicated calculation should be compiled on the local computer, which is faster than running in the robot.
- For modules running on the computer, a better computer to be used as a server can directly reduce the processing delay.
- Programming with a more efficient language, for example, C++. Python is much slower than C++ (May up to about 400 times), Python is more of a memory hog. For some games, like the Candy Crush Saga game has almost no requirements of timing, however, C++ is more suitable for those games that have requirements of timing.

7.2 Vision Algorithm applied in the Practical World

For the reason that our long-term goal is to compete with other teams in the RoboCup soccer game, it is important to develop our own vision algorithm to fit the high requirements in the real robot soccer game. To reach this goal, several demands of vision system need to be satisfied:

- Relativity accurate recognition

For most vision systems, accurate recognition is the basic requirement. In the RoboCup competition, the requirement will be higher because any inaccuracy even false recognition will negatively affect the team.

- Low processing time

In the robot soccer game, teammates, opponents and a red ball are all moving all the time, rapid reaction will gain big advantage.

- Stay reliable in a different environment

The robot soccer field is a complicated environment, the size is larger than testing field in the laboratory and light conditions may not be guaranteed.

- Have the ability to calculate the approximate distance between target and robot

Although an object recognition function has already been included in Choregraphe 2.1, however, the tested result is not good enough, especially when the target appears smaller or larger in the sight of the robot then in its memory. Therefore, a more flexible algorithm needs to be designed and it needs to be able to recognize the target in a different distance, moreover it needs to calculate the distance between the robot and the target based on the size appears in the sight of robot.

- Recognize multiple targets in the same field of vision

In the robot soccer game, normally more than one target will appear in the field of the vision of the robot at the same time. So it is important to identify possible teammates, opponents and ball correctly at the same time.

8 SUMMARY

This thesis introduces the development of Vision system applied in a PC game and game strategy with the Nao robot in detail for Botnia RoboCup Humanoid Robot Team.

By the developments of the vision system algorithm and the design of game strategy with the Nao robot, the possibility for the research of the Nao robot applied in PC game area is discussed. Although there are some limitations, there are still a lot of video games that can be used as further topic of the Nao robot research. In this thesis, level 1 of the Candy Crush Saga game is mainly used as an example and a far more complicated game, the Legal of Legend game, is also discussed with the algorithm. The main algorithm is based on color recognition and noise elimination and reduction. The mathematical model and statistics are utilized, which helps to dig the useful information from a large quantity of RGB data.

As the old saying goes, “What hurts more: the pain of hard work or the pain of regret?” When people focus on a huge project, it is understandable that they will face a lot of difficulties, in the project itself or daily life. Especially when encountering with a new things like the Nao robot, self-studying should never be stopped. Lots of ways can be chosen to fight against the problem, like reading academic articles, taking advanced courses, consulting experts, etc. For any students who are interested in this project, this thesis recommends they first thoroughly figure out the basic function of the Nao robot and the whole structure of the system. For the new student, it is strongly recommended to study the basic knowledge by following the tuition to avoid any damage to the Nao robot. Based on that, the students should then decide the research direction and start work. Whatever direction is chosen, do not underestimate yourself and do not quit easily. After the project is finally finished, when people review this process, they will feel the ground and satisfaction, and have surely learned a lot during the process.

We wish all the students working on this project the best for their future study and career.

9 REFERENCES

/1/ UK robots prepare for world cup. BBC News. 25 October 2010.

/2/ Aldebaran Robotics announces Nao Next Gen humanoid robot. Engadget.

/3/ Unveiling of NAO Evolution: a stronger robot and a more comprehensive operating system. Aldebaran Robotics. 2014.

/4/ RoboCup Standard Platform League. Tzi.de. Accessed 1 February 2015. http://www.robocup2014.org/?page_id=943

/5/ Aldebaran Nao Humanoid Robot. Active8 Robots UK. 2014. Accessed 1 February 2015.http://www.shopage.fr/?q=NAO+Robot+NAO+Evo+Humanoid+Robot+Active+Robots#

/6/ More about NAO. Accessed 1 February 2015. https://www.aldebaran.com/en/more-about

/7/ Aldebaran Robots and NAOqi OS documentation. Accessed 1 February 2015.http://doc.aldebaran.com/

/8/ Poynton, Charles A. 2003. Digital Video and HDTV: Algorithms and Interfaces. Morgan Kaufmann. ISBN 1-55860-792-7.

/9/ Joblove, George H., Greenberg, Donald. August, 1978. Color spaces for computer graphics. Computer Graphics 12 (3), 20–25.

/10/ Hunter, Richard Sewall. December, 1948. Accuracy, Precision, and Stability of New Photo-electric Color-Difference Meter. JOSA 38 (12): 1094.

/11/ Anderson, Dean Anderson. Color Spaces in Frame Grabbers: RGB vs. YUV. Accessed 1 February 2015. http://www.sensoray.com/support/frame_grabber_capture_modes.htm

/12/ YUui Ting Pai. 2006. Simple and Accurate Color Face Detection Algorithm in Complex Background. Multimedia and Expo. IEEE International Conference.

/13/ L Chen, C Grecos. 2005. A fast skin region detector for colour images. IEE In-ternational Conference on Visual Information Engineering (VIE 2005)

/14/ TIOBE Programming Community Index Python. 2012. Accessed 1 February 2015.TIOBE Software Index

/15/ Howard, Ian P.; Rogers, Brian J. 1995. Binocular vision and stereopsis. New York: Oxford University Press. Accessed 1 February 2015

APPENDIX 1

MAIN FUNCTION OF CANDY CRUSH SAGA APPLICATION:

```
if __name__ == '__main__':
  IP = "192.168.1.100"  #  IP address.
  PORT = 9559
  step=6
  n=0

  #Read IP address from first argument if any.# left inside line
up shadow
  if len(sys.argv) > 1:
     IP = sys.argv[1]
  for n in xrange(step):
     print 'this is step %r'%(n+1)
     naoImage = showNaoImage(IP, PORT)
     time.sleep(4)
```

APPENDIX 2

A PIECE OF CODE OF ILLUMINATION COMPENSATION PROGRESS

```
function[]= illumination_compensationpart(RGB_VALUE)
YCbCr=rgb2ycbcr(RGB_VALUE);
Y=YCbCr(:,:,1);
%normalize Y
minY=min(min(Y));
maxY=max(max(Y));
Y=255.0*(Y-minY)./(maxY-minY);
YEye=Y;
Yavg=sum(sum(Y))/(W*H);

T=1;
if (Yavg<64)
    T=1.4;
elseif (Yavg>192)
    T=0.6;
end

if (T~=1)
    RI=R.^T;
    GI=G.^T;
else
    RI=R;
    GI=G;
end
```